

# Configuration Manual

MSc Research Project Cloud Computing

Ghiridhar Iyer Student ID: X18183468

School of Computing National College of Ireland

Supervisor: Dr. Manuel Tova-Izquierdo

#### National College of Ireland Project Submission Sheet School of Computing



Student Name:	Ghiridhar Iyer
Student ID:	X18183468
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Manuel Tova-Izquierdo
Submission Due Date:	17/8/2020
Project Title:	Configuration Manual
Word Count:	4283
Page Count:	61

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	28th September 2020

#### PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).					
Attach a Moodle submission receipt of the online project submission, to					
each project (including multiple copies).					
You must ensure that you retain a HARD COPY of the project, both for					
your own reference and in case a project is lost or mislaid. It is not sufficient to keep					
a copy on computer.					

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only						
Signature:						
Date:						
Penalty Applied (if applicable):						

# Configuration Manual

### Ghiridhar Iyer X18183468

### 1 Introduction

The implementation of the artifact was done using the Amazon Web Services (AWS) platform. This configuration manual guides in executing the artifact. The following services were utilised during the implementation:

- Amazon S3
- AWS Athena
- AWS IAM
- AWS Sagemaker
- AWS QuickSight

The following sections are divided based on the implementation process.

### 2 Data Acquisition

Log in to the AWS Console at https://console.aws.amazon.com/



Figure 1: AWS Console Login

😒 WhatsApp	🗙 🥚 AWS Managem	ent Console × +			-	٥	×
$\leftrightarrow$ $\rightarrow$ C $\triangle$	Console.aws.amazon.com/cons	ole/home?region=us-east-1		ф	₩.	* 💿	:
🗰 Apps 🛛 G Googl	e 📕 My Account 💩 School of Skill	📔 Rediff.com: Online 🔇	NSE - National Stoc 💿 Account Acces	s - L 🖮 Drivers and Downlo 😁 Product Support   D 💶 Solve Pro	ogramming	j	>>
	Q Example: Relational Databas	e Service, database, RDS		or Android mobile device. Learn more 🗹			*
	Recently visited services			Explore AWS			1
	▼ All services	Die skehe in	The Committee Indonetitee 8	Amazon Redshift			
	EC2 Lightsail [2]	Amazon Managed Blockchain	Compliance	Fast, simple, cost-effective data warehouse that can extend queries to your data lake. Learn more 🖸			
	Batch Elastic Beanstalk Serverless Application	Satellite Ground Station	Manager Cognito Secrets Manager	Run Serverless Containers with AWS Fargate AWS Fargate runs and scales your containers without having			
	Repository AWS Outposts EC2 Image Builder	<ul> <li>Quantum</li> <li>Technologies</li> <li>Amazon Braket</li> </ul>	Inspector Amazon Macie AWS Single Sign-On	Scalable, Durable, Secure Backup & Restore with			
	Containers Elastic Container Registry	Management & Governance	Certificate Manager Key Management Service	Amazon S3 Discover how customers are building backup & restore solutions on AWS that save money. Learn more			
	Elastic Container Service Elastic Kubernetes Service	AWS Organizations CloudWatch AWS Auto Scaling CloudFormation	Directory Service WAF & Shield AWS Firewall Manager	AWS Marketplace Find, buy, and deploy popular software products that run on			
	Storage	CloudTrail Config OpsWorks	Artifact Security Hub Detective	AWS. Learn more 🖸			-

In the list of services, navigate to Storage section and click on S3.

Figure 2: Click on S3

Click on 'Create Bucket' button in the page. Enter the bucket name as 'floodprediction-master-dataset'. Since S3 is Global, Region needs to be explicitly mentioned while bucket creation. Ensure to have the S3 bucket in the same region where the rest of the services like Sagemaker are deployed. This manual has deployed all the services at the 'US East (N. Virginia)' region. In the ML code, bucket name is explicitly mentioned. If this bucket name is unavailable and different Bucket name is being used, ensure to update the code with the new bucket name.

← → C 介	s3.console.aws.amazon.com/s3	/home?region=us-east-1#			÷ 🚽 🛎 💼	:
Apps G Goog	le 🚦 My Account 💩 School of Skill	Rediff.com: Online ONSE - National Stoc	• Account Access - L 😔 Drivers and Downl	o 🥯 Product Support   D	Solve Programming	»
aws	Services 🗸 Resource Gro	ups 🗸 🔭		🗘 Ghiridhar 🗸	Global 🕶 Support 👻	
Amazon S3		Create	bucket		×	Ê
Buckets	1 Name and region	2 Configure options	3 Set permissions	(4) Review		
Batch operations Access analyzer S3	Name and region Bucket name ()				nsole	
Block public acc	flood-prediction-master-da	taset				
(account settings	Bucket name is already	owned by you			3	
	Region					
	US East (N. Virginia)			~	:22	
	Copy settings from an ex	isting bucket			5:38	
	Create			Cancel	8:43	•
<b>e</b> Feedback	🚱 English (US)				Privacy Policy Terms of U	



Select the bucket and click on 'Edit Public access settings'.

dWS Service	s 🗸 Resource Groups 🗸 🛠		🗘 Ghiridha	ar 🕶 Global 👻 Support 👻
	S3 buckets		C	Discover the console
Amazon 53	Q Search for buckets		All access types	~
Buckets				
3atch operations	+ Create bucket Edit public access settings Empty Delete		6 Buckets	1 <sub>Regions</sub>
Access analyzer for	Bucket name 🔻	Access 🚯 🔻	Region 👻	Date created 👻
Block public access	Codepipeline-us-east-1-436189706244	Objects can be public	US East (N. Virginia)	Jul 3, 2020 11:44:22 PM GMT+0530
account settings)	Flood-prediction-master-dataset	Bucket and objects not public	US East (N. Virginia)	Jul 15, 2020 10:55:38 AM GMT+0530
Feature spotlight 2	🗍 👿 floodashboard.co.uk	Objects can be public	US East (N. Virginia)	Jul 3, 2020 11:31:46 PM GMT+0530
	ghiri-sample-quicksight	Bucket and objects not public	US East (N. Virginia)	Jul 29, 2020 11:58:43 AM GMT+0530
	ghiridhar-sagemaker	Bucket and objects not public	US East (N. Virginia)	Jul 26, 2020 7:14:17 PM GMT+0530
	Sagemaker-us-east-1-884654660367	Objects can be public	US East (N. Virginia)	Jul 25, 2020 7:42:36 PM GMT+0530

Figure 4: Change Public Access Settings

Deselect 'Block all public access' option. The reason for making bucket public will be explained in the upcoming steps. Click on Save button



Figure 5: Unblock Public Access

Type 'confirm' in the textbox and click on confirm.

Services v	Resource G	roups 🗸 🏌	Ĺ			
Amozon S2	S3 buckets			C	Discover the console	
Amazon 55	Q Search	for buckets	All acce	ess types	~	
Buckets Batch operations	+ Create	Edit block public access settings for selected	×	Buckets	1 Regions 😂	
Access analyzer for S3	🗌 Buc	Ducketa			Date created 💌	
Block public access				N. Virginia)	Jul 3, 2020 11:44:22 PM GMT+0530	
(account settings)		Updating the Amazon S3 block public access settings affects all selected buckets. This may result in some buckets and objects becoming public.		N. Virginia)	Jul 15, 2020 10:55:38 AM GMT+0530	
Feature spotlight 2				N. Virginia)	Jul 3, 2020 11:31:46 PM GMT+0530	
		to confirm the settings, type <i>confirm</i> in the field.		N. Virginia)	Jul 29, 2020 11:58:43 AM GMT+0530	
		Carrel	irm	N. Virginia)	Jul 26, 2020 7:14:17 PM GMT+0530	
				N. Virginia)	Jul 25, 2020 7:42:36 PM GMT+0530	



Click on the bucket. Click on 'Create Folder'. Enter the folder name as 'raw-datasets' and click on Save button.

$\leftrightarrow \rightarrow c$		s3.console.aw	s.amazon.com/s3	/buckets/flood-predic	tion-master-dataset/?regior	n=us-east-1&tab=overvie	W		☆	🕂 🗯 🌀 E
Apps 🕻	G Google	My Account	😡 School of Skill	Rediff.com: Online	🔇 NSE - National Stoc	• Account Access - L	🖮 Drivers and Downlo	Product Support   D	HI Solve Pro	gramming »
aw	IS 	Services 🗸	Resource Gro	ups 🗸 🛧				🗘 Ghiridhar 🗸	Global 👻	Support 👻
	<b>Q</b> Туре	a prefix and pre	ess Enter to searcl	h. Press ESC to clear.						
	🔔 Upload	d + Create	bilder Dowr	Iload Actions ~	]			US East (N	. Virginia)	ø
								< X	/iewing 1 to 8	
	Nam	ne 🔻				Last modified -	Size 🔻	Storage class	· •	
	Whe appe Cho	raw-datasets en you create a t ended by suffix ' ose the encrypti	older, S3 console '/" and that object on setting for the	creates an object with is displayed as a folde object:	h the above name er in the S3 console.					
		None (Use buck AES-256	et settings)							
	L.	Use Server-Side E	ncryption with Amazo	on S3-Managed Keys (SS	E-S3)					
	) I	AWS-KMS Use Server-Side E	ncryption with AWS F	KMS-Managed Keys (SSE	E-KMS)					
	[	Save Canc	el							
Operatio	ions	0 In	progress 2 S	uccess 0 Error						•
🗨 🗨 Feed	Iback 🔇	English (US)				© 2008 - 2020, Am	azon Web Services, Inc. or it	ts affiliates. All rights reserve	d. Privacy Pol	icy Terms of Use

Figure 7: Create Folder in Bucket

|--|

$\leftarrow \rightarrow$	CΔ	a s3.console	a.aws.amazon.com/s	3/buckets/flood-pred	iction-master-dataset/?regio	n=us-east-1&tab=overvie	2W		\$	* 😨	:
Apps	s Ġ Goo	ogle 📕 My Accou	int 🛛 Ochool of Skil	I 📔 Rediff.com: Onli	ne 🧿 NSE - National Stoc	Account Access - L	Drivers and Downlo	Product Support   D	Solve Program	ming	**
	aws	Services 🗸	<ul> <li>Resource Gr</li> </ul>	oups 🗸 🗙				🗘 Ghiridhar 🕶	Global 👻 S	upport 👻	
	Q	Type a prefix and	press Enter to sear	ch. Press ESC to clea	Ir.						-
	<b>2</b> , U	Jpload + Cre	eate folder Dow	/nload Actions	<u>·</u>			US East (N. V	/irginia) 🥲		ļ
								Viev	ving 1 to 8		
		Name 🔻				Last modified -	Size 🔻	Storage class 🔻			
		athena-files	i								
		🖢 final-datase	t-1-hr								
		🕭 final-datase	t-15-min								
		📂 ml-sensor-o	lata-1-hr								
		📂 ml-sensor-o	lata-15-min								
		predictions-	1-hr								
		predictions-	15-min								
		🖕 raw-datase	ts								
Ope	erations		) In progress 2	Success 0 Error							
<b>₹</b>	eedback	😧 English (U	S)			© 2008 - 2020, An	nazon Web Services, Inc. or it	s affiliates. All rights reserved.	Privacy Policy	Terms of Us	se

Figure 8: Create all these folders

Click on the Services drop down in the top left corner of the screen. Navigate to the Machine Learning section. Click on Amazon SageMaker.

	aws Services A	Resource Groups 🗸 🔭	🗘 Ghiridhar 🗸
	History Amazon SageMaker		
-	S3 Athena Console Home Billing	Machine Learning Amazon SageMaker Amazon Augmented Al Amazon CodeGuru	
	IAM	Amazon Forecast Amazon Fraud Detector Amazon Kendra Amazon Lex Amazon Personalize	
		Amazon Polly Amazon Rekognition Amazon Textract Amazon Transcribe Amazon Translate AMS DeenComposer	

Figure 9: Click on Sagemaker

Click on Notebook instances from the menu on the left.

Amazon SageMaker Studio	AWS Marketplace Find, buy, and deploy r				
Dashboard	Find, buy, and deploy r				×
Search		eady to use model packages,	algorithms, and data products	in AWS Marketplace	
Search	Browse Catalog				
Ground Truth					
Labeling jobs	Overview				Hide
Labeling datasets					
Labeling workforces					
Notebook	503	(B) SOS	( 22 SOS		( 22 × 0 >
Notebook instances	$( \bigcirc \ll )$			( 🚓 🕯 🦛 ) <sup>0</sup> )))	
Lifecycle configurations				/	
Git repositories	Ground Truth	Notebook	Training	Inference	Processing Run
Processing	labeling jobs for highly	SageMaker SDKs and	any scale. Leverage high	training jobs or import	and model evaluation
Processing	accurate training datasets	sample notebooks to	performance AWS	external models for	workloads with a fully
Processing jobs	human labeling.	deploy models.	own.	on new data.	managed experience.
Training				Ac	il <del>vate Windows</del>
A Los out Alexand	Labeling jobs	Notabook	Training jobs	Models	ion settings to activate wingows.

Figure 10: Click on Notebook Instances

Click on 'Create Notebook Instance'. Multiple Jupyter Notebooks can be created within an instance.

aws Services v	Resource Groups 🗸 🗙	🗘 Ghiridhar 🕶 N. Virginia 👻 Support 👻
Amazon SageMaker $~~ imes$	Amazon SageMaker > Notebook instances	
Amazon SageMaker Studio	Notebook instances	C Actions  Create notebook instance
Dashboard Search	Q Search notebook instances	< 1 > ⊚

#### Figure 11: Click on Create Notebook Instance

Enter the Instance name and type.

reate noteboo.	k instance
nazon SageMaker provides p clude example code for comm	e-built fully managed notebook instances that run Jupyter notebooks. The notebook instances ion model training and hosting exercises. Learn more 🗹
Notebook instance se	ttings
Notebook instance name	
Maximum of 63 alphanumeric ch	aracters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.
Notebook instance type	
ml.m4.xlarge	▼
Elastic Inference Learn mor	

Figure 12: State the Name and Instance Type

Choose the IAM Role for the SageMaker instance. If not created choose create a new role.

aws Services 🗸 Resource Groups 🗸 🖈			
■ Permissions and encryption			
IAM role Notebook instances require permissions to call other services including SageMaker and S AmazonSageMakerFullAccess IAM policy attached.	3. Choose a ro	ole or let us create a role with the	
AmazonSageMaker-ExecutionRole-20200712T142002			
Create a new role			
Enter a custom IAM role ARN			
Use existing role			
AmazonSageMaker-ExecutionRole-20200712T142002			
No Custom Encryption			
Network - optional			
· · ·			
<ul> <li>Git repositories - optional</li> </ul>			
► Tags - optional			
	Cancel	Create notebook instance	
🗨 Feedback 🔇 English (US)	l		© 2008 - 2020, Amazon Web

Figure 13: Select IAM Role

On clicking 'create a new role', the pop up asks to specify a particular bucket to provide access. Access can be provided to a single bucket, all buckets or no buckets. Select 'Specify S3 buckets' and enter 'flood-prediction-master-dataset'.

Create an IAM role	×
Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the AmazonSageMakerFullAccess [2] IAM policy to the role you create. The IAM role you create will provide access to:	Î
<ul> <li>S3 buckets you specify - optional</li> <li>Any S3 bucket Allow users that have access to your notebook instance access to any bucket and its contents in your account.</li> <li>Specific S3 buckets flood-prediction-master-datası Comma delimited. ARNs, "** and "/" are not supported.</li> <li>None</li> </ul>	
⊘ Any S3 bucket with "sagemaker" in the name	
⊘ Any S3 object with "sagemaker" in the name	-1
⊘ Any S3 object with the tag "sagemaker" and value "true" See Object tagging 2	<u>,</u> 2
Cancel	e

Figure 14: Specify S3 Bucket when creating IAM Role

Create Notebook instances as shown in the below figure 15. Ensure to provide 'ml.m4.xlarge' for any one instance. This instance is required for GAN generation. GAN generation is compute intensive process which is not possible in 'ml.t2.medium' or 'ml.t3.large' instances. Click on Open Jupyter link.

Not	ebook instances Search notebook instances			C	Actions 🔻	Create notebook instance
	Name	$\nabla$	Instance	Creation time	Status 🛛	Actions
0	gan-and-sensor-data-merge		ml.t2.medium	Jul 23, 2020 17:21 UTC	⊘InService	Open Jupyter   Open JupyterLab
0	time-series-algorithms		ml.t3.large	Jul 20, 2020 15:37 UTC	⊘ InService	Open Jupyter   Open JupyterLab
0	CTGAN		ml.m4.xlarge	Jul 18, 2020 17:44 UTC	⊘ InService	Open Jupyter Open JupyterLab

Figure 15: Click on Open Jupyter

Click on New and select 'conda\_python3'. This creates a jupyter notebook instance with a Python 3 environment.



Figure 16: Create New Python3 File

Click on the title of the notebook stated as 'Untitled'. Pop up emerges to rename the file. Rename the file to 'Data\_Acquisition' and click on Rename.

C Jupyter Untitled Last	Checkpoint a few seconds ago (unsaved changes)	_	Logout
File Edit View Insert	Rename Notebook	Trusted	conda_python3 O
	Enter a new notebook name:		
In []:	Data_Acquisition		
	Cancel Rename		

#### Figure 17: Rename File

Import libraries. The Environment agency API is used to retrieve the sensor data using pandas and saved as a CSV file. Public Access Rights is provided by the website <sup>1</sup>.

In [ ]:	#importing libraries
	import pandas as pd
In [ ]:	<pre>ng datasets</pre>
In [ ]:	<pre>#converting to csv rainfall.to_csv("Rainfall.csv",index=False) temperature.to_csv("Temperature.csv",index=False) rest_all.to_csv("Wind Speed Direction and River level.csv",index=False)</pre>

#### Figure 18: Acquire Data from API and save as CSV

Boto3 SDK  $^{2}$  is used to access the S3 bucket and store the CSV file.



Figure 19: Save to S3 using Boto3 SDK

 $<sup>{}^{1}</sup> Public \ Access: \ {\tt https://environment.data.gov.uk/flood-monitoring/doc/reference}$ 

 $<sup>^2</sup>Boto3$  Documentation: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/migrations3.html

# 3 Data Transformation and Formatting

Click on the services drop down and navigate to the Analytics section. Click on Athena. Athena provides SQL based data querying on S3 objects.

← → C ☆ 🔒 s3.console.aw	vs.amazon.com/s3/buckets/flood-prediction-m	naster-dataset/?region=us-east-1&tab=ove	erview	😒 🐥 🜲 😨 E
🗰 Apps 🛛 G Google 📑 My Account	😡 School of Skill 📔 Rediff.com: Online	🔕 NSE - National Stoc 💿 Account Access -	L 😔 Drivers and Downlo 😔 Produ	ct Support   D 🛚 Solve Programming »
aws Services 🔺	Resource Groups 🗸 🔸		¢	Ghiridhar 🗸 Global 🖌 Support 🗸
History S3	Find a service by name or feature (for exar	mple, EC2, S3 or VM, storage).		Group A-Z
Console Home Billing IAM Amazon SageMaker AWS Cost Explorer	Compute CC2 EC2 Lightsail C* Lambda Batch C Elastic Beanstalk	Blockchain      Amazon Managed Blockchain     Satellite     Ground Station	Analytics Athena EMR CloudSearch Elasticsearch Service Kinesis	Business Applications Alexa for Business Amazon Chime [2] WorkMail Amazon Honeycode
	Serverless Application Repository AWS Outposts EC2 Image Builder	Quantum Technologies Amazon Braket	QuickSight C Data Pipeline AWS Data Exchange AWS Glue AWS Lake Formation	End User Computing WorkSpaces AppStream 2.0 WorkDocs
	Storage S3 EFS FSx S3 Glacler Storage Gateway AWS Backup	Management & Governance AWS Organizations CloudWatch AWS Auto Scaling CloudFormation CloudTrail	MSK ) Security, Identity, & Compliance IAM Resource Access Manager	WorkLink Internet Of Things IoT Core FreeRTOS IoT 1-Click
🗨 Feedback 📢 English (US)		▲ close © 2008 - 2020	), Amazon Web Services, Inc. or its affiliates. Al	I rights reserved. Privacy Policy lerms of Use

Figure 20: Click on Athena

Choose the Data source as S3 and metadata as AWS Glue. Click on Next.

Athena Query editor Saved queri	es History Data sources Workgroup : primary	Settings	Tutorial	Help	What's new
Connect data source					
Step 1: Choose a data source	Choose where your data is located				
Step 2: Connection details	Athena queries data where it is. Data is not loaded or moved. Learn more C				
	Query data in Amazon S3     Query a data source (beta)				
	Choose an external data catalog. Configure a connector for common da	ata sources.			
	Choose a metadata catalog				
	The catalog contains the schema for the source data such as column names, data types and table names. Learn mo	re 🗗			
	AWS Glue data catalog				
		Cancel	ext		
Feedback 🔿 English (US)	@ 2008 - 2020 Amazon Web Services Inc. or	its affiliates. All rights re	eserved P	rivacy Policy	v Terms of Use

Figure 21: Choose S3 as Data Source

Choose to enter table schema manually and click on 'Continue to add Table'.



Figure 22: Choose to enter Table Structure Manually

Enter the Dataset name, Table name and S3 path to the raw datasets. The path should be a folder. Athena extracts all the data from the files as a single file. Click on Next.

Step 1: Name & Location	Step 2: Data Format Step 3: Columns Step 4: Partitions
Database	Create a new database   Choose an existing database or create a new one by selecting "Create new database".
	floodpredictiondatase
Table Name	all_sensor_data Name of the new table. Table names must be globally unique. Table names tend to correspond to the directory where the data will be stored.
Location of Input Data Set	s3://lood-prediction-master-dataset/raw-datasets/  Input the path to the data set you want to process on Amazon S3. For example if your data is stored at s3://input-data-set/logs// Logs// Log
External	Note: Amazon Athena only allows you to create tables with the EXTERNAL keyword. Dropping a table created with the External keyword does not delete the underlying data.
Next	

Figure 23: Enter Dataset and Table Name and provide path

Choose Datatype of the Data source as CSV and click on Next.

Databases > Add table					
Step 1: Name & Location	Step 2: Data Format	Step 3: Columns	Step 4: Partitions		
Data Format	Apache Web Logs     CSV     TSV     Text File with Custor     JSON     Parquet     ORC	n Delimiters			
Back					

Figure 24: Set Datatype as CSV

Enter the schema of the table. Since the schema of all the three files is same, it does not affect the table schema process. In case the schema of each file differs, additional steps would be involved. Click on Add column to add new column schema. After entering the schema as per the figure below click on Next.

Step 1: Name & Location Step 2: Data Format Step 3: Columns Step 4: Partitions	
Column Name TimeRecorded Column name must be single words that start with a letter or a digit.	
Column type string  Type for this column. Certain advanced types (namely, structs) are not exposed in this interface.	
Column Name stationur Column name must be single words that start with a letter or a digit.	x
Column type string	
Column Name Sensorvalue Column name must be single words that start with a letter or a digit.	×
Column type for this column. Certain advanced types (namely, structs) are not exposed in this interface.	
Add a column Bulk add columns	
Back Next	

Figure 25: Enter Table Structure

Click on Create Table since Partitioning is not essential.

Databases > Ad	d table		
Step 1: Name & Location	Step 2: Data Format	Step 3: Columns	Step 4: Partitions
Configure Parti Partitions are a way to grou table's data directory for ea	tions (Optional up specific information to ach unique value of a part	) gether. Partition are vi ition column. In case	rtual columns. In case of partitioned tables, subdirectories are created under the table is partitioned on multiple columns, then nested subdirectories are
created based on the order	r of partition columns in t	ne table definition. Le	am more.
Add a partition			
Back Create table			

Figure 26: Create Table

In every tab, queries can be executed based on the tables created. Click on the '+' tab to add new tab. The code in the below figure selects the created table which contains all the raw data.



Figure 27: Show all Data

Selects data based on station. This provides five categories as output since there are five sensor data.



#### Figure 28: Group Data by sensor type

Creates a River level table based on sensor station value.



#### Figure 29: Create River level Table

Creates a Rainfall table based on sensor station value.



Figure 30: Create rainfall Table

Creates a Temperature table based on sensor station value.



#### Figure 31: Create temperature Table

Creates a Wind Speed table based on sensor station value.



#### Figure 32: Create Wind Speed Table

Creates a Wind Direction table based on sensor station value.





Creates a table by applying a Join between River level table and rainfall table based on timestamp column.



Figure 34: Joining River level and rainfall

Creates a table by applying a Join with the above table and temperature table based on timestamp column.



Figure 35: Joining Temperature

Creates a table by applying a Join with the above table and Wind Speed table based on timestamp column.





Creates a table by applying a Join with the above table and Wind Direction table

based on timestamp column. This creates a table with 6 columns - one timestamp column and five sensor data columns with a time period of 15 minutes.



Figure 37: Final 15 minutes dataset

Aggregating the above table based on hour. Aggregate wind speed, wind direction and temperature to their average values. Aggregate rainfall to its sum value and river level to its max value.



Figure 38: Aggregated Dataset

Navigate to the S3 bucket inside the 'ml-sensor-data-15-min' folder. The table will be stored in a .gz format. Select the file and go to Actions. Click on 'Make Public' option. Although bucket was made public, objects are not public unless explicitly made public.

Amazon S3 > flood-prediction-master-data	iset > ml-sensor-data-15-mi	n			
flood-prediction-master-datase	ıt	-			
Overview					
<b>Q</b> Type a prefix and press Enter to search. P	ress ESC to clear.				
Upload     + Create folder     Download	d Actions ~ Add tags			US East (N. ∖	/irginia) 🛛 😂
Upload     + Create folder     Download	d Actions ~ Add tags Make public	1		US East (N. V	/irginia) 🛛 😂
Upload     Create folder     Download	d Actions ~ Add tags Make public Rename	▲ Last modified ▼	Size 🕶	US East (N. V View Storage class <del>•</del>	/irginia) 🤁
Upload	d Actions ~ Add tags Make public Rename Delete	▲ Last modified ▼ Aug 3, 2020 9:12:48 PM GMT+0530	Size ▼ 33.4 KB	US East (N. V View Storage class ▼ Standard	/irginia) 2
Upload	d Actions ~ Add tags Make public Rename Delete Undo delete	▲ Last modified ▼ Aug 3, 2020 9:12:48 PM GMT+0530	Size ▼ 33.4 KB	US East (N. V View Storage class ▼ Standard	Virginia) 2
Upload     + Create folder     Download      Name     Iml_sensor_data_15_min.gz	d Actions ~ Add tags Make public Rename Delete Undo delete Copy	▲ Last modified ▼ Aug 3, 2020 9:12:48 PM GMT+0530	Size ▼ 33.4 KB	US East (N. View Storage class • Standard View	Virginia) C
Upload	d Actions ~ Add tags Make public Rename Delete Undo delete Copy Move	▲ Last modified ▼ Aug 3, 2020 9:12:48 PM GMT+0530	Size ▼ 33.4 KB	US East (N. View Storage class • Standard View Activate V	Virginia) C ving 1 to 1

Figure 39: Making 15 Minute dataset public

Click on 'Make Public' to confirm the action. Do the same action for 1 hour dataset in the 'ml-sensor-data-1-hr' folder.

Make public	×
Selection: 1 Objects, 0 Folders Total size: 33.4 KB Total objects: 1	
ml-sensor-data-15-min/ml_sensor_data_15_min.gz     s3.4 кв	
	_
Everyone will have access to one or all of the following: read this object, read and write permissions.	
Cancel Make pu	blic

Figure 40: Confirming

Create jupyter python\_3 file with file name 'Source\_Dataset\_gz\_to\_csv' inside the CT-GAN instance. Retrieve both the .gz files and save them as CSV files.

In [ ]:	#import packages
	import pandas as pd
In [ ]:	#import source file
	<pre>hr_file = pd.read_csv("https://flood-prediction-master-dataset.s3.amazonaws.com/ml-sensor-data-1-hr/ml_sensor_data_1_hr.gz", name</pre>
	<pre>min_file = pd.read_csv("https://flood-prediction-master-dataset.s3.amazonaws.com/ml-sensor-data-15-min/ml_sensor_data_15_min.gz",</pre>
In [ ]:	#saving as csv file
	<pre>hr_file.to_csv("ml_sensor_data_1_hr.csv",index=False,header=None) min_file.to_csv("ml_sensor_data_15_min.csv",index=False,header=None)</pre>

Figure 41: retrieve file using pandas

Using Boto3, save the CSV files to their respective folders in S3. This Format conversion was done by making the bucket and the .gz file as public, as Boto3 faces issues while streaming and decoding the .gz file.

import datetime import tarfile
<pre>import boto3 # AWS SDK for python. Provides Low-Level access to AWS services from sagemaker import get_execution_role import sagemaker</pre>
<pre>m_boto3 = boto3.client('sagemaker')</pre>
<pre>sess = sagemaker.Session()</pre>
region = sess.boto_session.region_name
<pre>bucket = 'flood-prediction-master-dataset' # specify the S3 bucket to save the file</pre>
<pre>print('Using bucket ' + bucket)</pre>
# send data to S3. key_prefix is the directory path. path is the local file name which will be saved in S3 with same file name # and bucket is the bucket name
upload = sess.upload_data(path='ml_sensor_data_1_hr.csv', bucket=bucket, key_prefix='ml-sensor-data-1-hr/')
upload = sess.upload_data(path='ml_sensor_data_15_min.csv', bucket=bucket, key_prefix='ml-sensor-data-15-min/')

Figure 42: Save CSV format to S3

Navigate to the S3 bucket and inside the 'ml-sensor-data-15-min' folder. Since the .gz and .csv have the same name, a no named folder is created within which the CSV file is saved.

Amazon S3 > flood-prediction-master-dataset > ml-sensor-data-15-min flood-prediction-master-dataset Overview				
C Type a prefix and press Enter to search. Press ESC to clear.      L Upload + Create folder Download Actions			US East (N. Virginia) 🯾 🕫	
			Viewing 1 to 2	
□ Name ▼	Last modified -	Size 🔻	Storage class -	
ml_sensor_data_15_min.gz	Aug 3, 2020 9:12:48 PM GMT+0530	33.4 KB	Standard	

Figure 43: Go inside Empty Folder

Amazon S3 > flood-prediction-mast	er-dataset > ml-sensor-data-15-mi	n >			
flood-prediction-master-da	taset				
Overview					
<b>Q</b> Type a prefix and press Enter to se	arch. Press ESC to clear.				
▲ Upload + Create folder D	Actions ~	*			US East (N. Virginia) 🛛 😂
	Make public				Viewing 1 to 1
✓ Name ▼	Rename		Last modified -	Size 🔻	Storage class 🕶
✓ I ml_sensor_data_15_mir.csv	Delete		Aug 14, 2020 7:05:26 PM GMT+0530	136.5 KB	Standard
	Undo delete				
	Сору				Viewing 1 to 1
	Move				Activate Windows

Click on the folder. Select the CSV file and go to actions. Click on Move.

Figure 44: Move File

Click on the 'flood-prediction-master-dataset' S3 bucket. Do not choose the bucket, but click on the bucket name.

Choose move destination				
(1) Choose move destination				
Buckets		c		
Q Search by name		Î		
Name 👻	Region			
O codepipeline-us-east-1-436189706244	US East (N. Virginia)			
flood-prediction-master-dataset	US East (N. Virginia)			
floodashboard.co.uk	US East (N. Virginia)			
Ghiri-sample-quicksight	US East (N. Virginia)			
O ghiridhar-sagemaker	US East (N. Virginia)	_		
Sagemaker-us-east-1-884654660367	US East (N. Virginia)	•		
	Ca	Incel Choose		

Figure 45: Click on Destination S3 bucket

Choose the folder to move the file. Here select the 'ml-sensor-data-15-min'. Click on Choose.

Choose n	nove destination	×
1 Choose move destination		
S3 > flood-prediction-master-dataset		
Objects		c
		*
🔘 🖿 final-dataset-1-hr		
O in final-dataset-15-min		
O 🖿 ml-sensor-data-1-hr		
🔿 🖿 ml-sensor-data-15-min		
O predictions-1-hr		
O b predictions-15-min		
raw-datasets		-
	Cancel	ack Choose

Figure 46: Choose the destination folder

Click on Move to confirm the action.

Review	×
Choose move destination (2) Review	
Selection: 1 Objects, 0 Folders Total size: 136.5 KB Total	objects: 1
fit Source mi-sensor-data-15- Destination m min// se	ood-prediction- aster-dataset/ml- ensor-data-15-min/
☐ ml-sensor-data-15-min//ml_sensor_data_15_min.csv - 136.5 KB	
	Previous Move

Figure 47: Confirm Move Action

Ensure the process was successful. The no named folder is deleted by itself. Do the same process for the 1 hour CSV file.

Amazon S3 > flood-prediction-master-dataset > ml-sensor-data-15-min flood-prediction-master-dataset	]		
Overview			
Q Type a prefix and press Enter to search. Press ESC to clear.			
Lyload      ← Create folder Download Actions ~			US East (N. Virginia) 🛛 🤁
			Viewing 1 to 2
Name 🕶	Last modified	Size 💌	Storage class -
□ Inl_sensor_data_15_min.csv	Aug 14, 2020 7:10:59 PM GMT+0530	136.5 KB	Standard
□ B ml_sensor_data_15_min.gz	Aug 3, 2020 9:12:48 PM GMT+0530	33.4 KB	Standard

Figure 48: Ensure both files are inside the same directory.

Choose the bucket and click on 'Edit Public Access Settings'.

S3 buckets		C	Discover the console
Q Search for buckets	All access types	~	
Create bucket     Edit public access settings     Empty     Delete		6 Buckets	$1_{\text{Regions}}$
Bucket name 🔻	Access 🚯 🔻	Region 🔻	Date created 👻
codepipeline-us-east-1-436189706244	Objects can be public	US East (N. Virginia)	Jul 3, 2020 11:44:22 PM GMT+0530
Ilood-prediction-master-dataset	Objects can be public	US East (N. Virginia)	Jul 15, 2020 10:55:38 AM GMT+0530

Figure 49: Edit Bucket Public Access

Check on the 'Block all public access' option and click on Save.



Figure 50: Block access

Type 'confirm' in the textbox and click on Confirm. This converts all the public objects inside the bucket as private.

Edit block public access settings for selected $ imes$ buckets	
This will result in public access being blocked for selected buckets and all objects within.	
To confirm the settings, type <i>confirm</i> in the field.	
Cancel	

Figure 51: Confirm Action

# 4 GAN creation and Merging

Create a Python3 file named GAN\_generator\_15\_min in CTGAN instance. Go to Kernel and go to Change Kernel and Choose 'conda\_mxnet\_p36'. This Kernel enables installing Third Party libraries in AWS SageMaker environment.

File Edit View Insert Cell	Kernel	Widgets	He	elp
P + ≫ 2 b ↑ ↓ H Run In [1]: #EXECUTE THIS COMMAN ! pip installuser	Interrup Restart Restart Restart Reconn	t & Clear Outpi & Run All ect	ut .	♥
Requirement already Requirement already (from ctgan) (0.22.1 Requirement already	Change	kernel	•	me/ec2-user/.local/lib/python3. R Sparkmagic (PySpark)
Requirement already gan) (1.18.1) Requirement already Requirement already	Conda Visit and satisfied	Packages aconda.org : torch<2,	>=1.	Sparkmagic (Spark) Sparkmagic (SparkR) conda_amazonei_mxnet_p27
t-learn<0.23,>=0.21- Requirement already -learn<0.23,>=0.21- Requirement already Requirement already	satisfied >ctgan) ( satisfied ctgan) (0 satisfied	: scipy>=0 1.4.1) : joblib>= .14.1) : pillow>=	.17. 0.11 4.1.	conda_amazonei_mxnet_p36 conda_amazonei_tensorflow2_p27 conda_amazonei_tensorflow2_p36
vision<1,>=0.4.2->ct Requirement already <0.26,>=0.24->ctgan) Requirement already prom packace(2.25.26)	gan) (7.0 satisfied (2019.3) satisfied	.0) : pytz>=20 : python-d	)17.2 Jatei	conda_amazonei_tensorflow_p27 conda_amazonei_tensorflow_p36 conda_chainer_p27 conda_chainer_p36
Requirement already 0->ctgan) (0.18.2) Requirement already eutil>=2.6.1->pandas	satisfied satisfied <0.26,>=0	: future i : six>=1.5 .24->ctgan	in /ł 5 in 1) (1	conda_mxnet_p27 conda_mxnet_p36 conda_python2
book.us-east-1.sagemaker.aws/notebooks/GAN_gene	erator_15_mi	n.ipynb# 1e	).2; /hom	conda_python3

Figure 52: Change Kernal

Install the ctgan library using the pip command.

Figure 53: Install CTGAN library

Change the kernel back to 'conda\_python3'.

File	Edit	View	Insert	Cell	Kernel	Widgets	He	elp	
-	• » 4	b 🖪	<b>↑ ↓</b>	Run	Interrup Restart			✓	
					Restart	& Clear Output	t		
	Tn [1]:	#EXEC	UTE THIS C	OMMAN	Restart	& Run All		onda mxnet p36". ONLY THIS KE	RNAL
		! pip	install -	-user	Reconn	ect /n			
		Requi	rement alr	eady :				me/ec2-user/.local/lib/python	3.6/9
		Requi	rement alr	eady :	Change	kernel		R	ana
		(from	ctgan) (0	.22.1				Sparkmagic (PySpark)	c 21
		Requi	rement alr	eady :	Conda F	Packages		Sparkmagic (Spark)	3/6
		gan)	(1.18.1)		Visit ana	iconda.org			
		Requi	rement alr	eady :			,	Sparkmagic (SparkR)	/1:
		Requi	rement alr	eady :	satisfied	: torch<2,>	=1.	conda_amazonei_mxnet_p27	py1
		t loo	rement air	eady :	satistied	: scipy>=0.	1/.	conda_amazonei_mxnet_p36	nvs
		Requi	rement alr	eadv	satisfied	: ioblib>=0	.11	conda_amazonei_tensorflow2_p27	vs
		-lear	n<0.23,>=0	.21->	ctgan) (0	.14.1)		conda amazonei tensorflow2 p36	
		Requi	rement alr	eady :	satisfied	: pillow>=4	.1.	conda_amazonoi_toncoffow_p27	nvs
		visio	n<1,>=0.4.	2->ct	gan) (7.0	.0)		conda_amazonei_tensoniow_pz/	
		Kequi	rement air	eady : toon)	(2010 2)	: pytz>=201	./.2	conda_amazonei_tensorflow_p36	vs,
		Requi	rement alr	eadv	satisfied	: python-da	ter	conda_chainer_p27	acc
		rom p	andas<0.26	,>=0.1	24->ctgan	) (2.8.1)		conda_chainer_p36	
		Requi	rement alr	eady :	satisfied	: future in	1 /ł	conda mxnet p27	et_
		0->ct	gan) (0.18	.2)			.	conda mynet p36	
		Kequi Autil	rement air >-2 6 1->n	eady : andac	5atistied (0 26 \-0	: S1X>=1.5 24->ctgan>	1n (1	conda_nuthon3	xne
		LIADAITI	V-2.0.1-7P			20.0.	2;	conda_python2	lat
book.us-ea	ast-1.sagem	aker.aws/	'notebooks/GA	N gene	erator 15 mir	n.ipynb#/	.i.	conda python3	

Figure 54: Change Kernal to Python3

Import all the required libraries. Stream the 15 minutes file from S3 and specify the column names.

In [ ]:	#import packages	
	from ctgan import CTGANSynthesizer import pandas as pd import boto3	
In [ ]:	#import source file	
	<pre>s3 = boto3.client('s3')</pre>	
	<pre>bucket = 'flood-prediction-master-dataset' key = 'ml-sensor-data-15-min/ml_sensor_data_15_min.csv'</pre>	
	<pre>obj = s3.get_object(Bucket= bucket,Key= key)</pre>	
	<pre>file = pd.read_csv(obj['Body'],names=["time","river","rain","temperature","wind_direction","wind_speed"])</pre>	
	file	
In [ ]:	#specify the headers to GAN	
	<pre>discrete_columns = [     'time',     'river',     'rain',</pre>	
	'temperature', 'wind_direction', 'wind_speed'	Activate Windows Go to Settings to activate

Figure 55: Retrieve file and mention columns

Declare the CTGAN, train the model and generate records.



Figure 56: Train and Create samples.

Save the Generated file as CSV and using Boto3 save it to S3.

In [ ]:	#save the file as csv locally
	<pre>samples.to_csv("gan_data_15_min.csv")</pre>
In [ ]:	# send data to 53. key_prefix is the directory path. path is the local file name which will be saved in S3 with same file name # and bucket is the bucket name
	upload = sess.upload_data(path='gan_data_15_min.csv', bucket=bucket, key_prefix='ml-sensor-data-15-min/')

Figure 57: Convert to CSV and save to S3

Create a python3 file named GAN\_generator\_1\_hr in CTGAN instance. Repeat the same process as in the above figures 52 to 55. Stream the 1 hour dataset and train the model.

In [ ]:	#EXECUTE THIS CONMAND WITH KERNAL SET TO "conda_mxmet_p36". ONLY THIS KERNALCAN INSTALL EXTERNAL LIBRARIES. ! pip installuser ctgan
In [ ]:	#import packages
	from ctgan import CTGNSynthesizer import pandas as pd import boto3
In [ ]:	#import source file
	<pre>s3 = boto3.client('s3')</pre>
	bucket = 'flood-prediction-master-dataset' key = "ml-sensor-data-l-nr/ml sensor data 1 nr.csy'
	nhi _ r3 mat nhiart/Duckat Kau_ kau)
	obj = SS.get_objett(Bucket, key= key)
	<pre>tile = pd.read_csv(obj['Body'],names=['Time', "river', "rain', "temperature', "wind_direction', "wind_speed"])</pre>
	file
In [ ]:	#specify the headers to GAN
	discrete columns = [
	'time',
	'river', 'rain'.
	'temperature',
	'wind_speed'
	]
In [ ]:	#creating instance
	<pre>ctgan = CTGAWSynthesizer()</pre>
In [ ]:	#training
	ctgan.fit(file, discrete_columns)

Figure 58: GAN generation for 1 Hour data

Generate sample records, save the file as CSV and stream it to S3 using Boto3.

In [ ]:	#sample creation
	samples = ctgan.sample <mark>(692)</mark>
In [ ]:	import datetime import tarfile
	<pre>import boto3 # AWS SDK for python. Provides low-level access to AWS services from sagemaker import get_execution_role import sagemaker</pre>
	<pre>m_boto3 = boto3.client('sagemaker')</pre>
	<pre>sess = sagemaker.Session()</pre>
	region = sess.boto_session.region_name
	<pre>bucket = 'flood-prediction-master-dataset' # specify the S3 bucket to save the file</pre>
	<pre>print('Using bucket ' + bucket)</pre>
In [ ]:	#save the file as csv locally
	<pre>samples.to_csv("gan_data_1_hr.csv")</pre>
In [ ]:	# send data to 53. key_prefix is the directory path. path is the local file name which will be saved in 53 with the bar of the bar of the saved in 53 with the bar of the saved in 53 with the bar of the saved in 53 with the bar of the bar of the saved in 53 with the bar of the bar of the saved in 53 with the bar of t
	upload = sess.upload_data(path='gan_data_1_hr.csv' bucket=bucket, key_prefix 'ml-sensor-data-1-hr/'

Figure 59: Create sample and upload to S3

Now create a Python3 notebook named Sensor\_and\_GAN\_merger\_15\_min in 'gan-and-sensor-data-merge' instance. Import libraries and Stream the 15 minutes sensor and GAN data.

In [ ]:	#packages
	import pandas as pd import numpy as np import boto3
In [ ]:	#file import from S3
	<pre>s3 = boto3.client('s3')</pre>
	bucket = 'flood-prediction-master-dataset' key = <mark>'ml-sensor-data-15-min/ml_sensor_data_15_min.csv'</mark>
	obj = s3.get_object(Bucket= bucket,Key= key) sensor_file = pd.read_csv(obj['Body'],names=["time","river","rain","temperature","wind_direction","wind_speed"])
	<pre>key = 'ml-sensor-data-15-min/gan_data_15_min.csv'</pre>
	obj = s3.get_object(Bucket= bucket,Key= key) gan_file = pd.read_csv(obj['Body']) #GAN was saved with header. hence no need for stating column names

Figure 60: Retrieve both files

Add DateTime column and source column. Source column states the source of the record - GAN or sensor. Combine both files

In [ ]:	#converting the string datatype of time values to datetime	
	<pre>sensor_file['timerecorded'] = pd.to_datetime(sensor_file['time']) gan_file['timerecorded'] = pd_to_datetime(gan_file['time'])</pre>	
	Barline conneg ] = barcolarcerme(Barline[ crme ])	
In [ ]:	#dropping the redundant column	
	<pre>sensor_file.drop(['time'],axis=1,inplace=True)</pre>	
	<pre>gan_file.drop(['time'],axis=1,inplace=True)</pre>	
In [ ]:	#sorting by datetime column	
	<pre>gan_file.sort_values(by=['timerecorded'],inplace=True)</pre>	
	<pre>sensor_file.sort_values(by=['timerecorded'],inplace=True)</pre>	
In [ ]:	#quicksight imports all data from all the csv files from the folder and files specified.	
	#Hence, a distinguishing column is required.	
	<pre>gan_file['source'] = 'GAN' Control of the second seco</pre>	
	sensor_tile[ source ] = SENSUK	
In [ ]:	#merging both datasets	
	<pre>final_file_15_min = gan_file.append(sensor_file)</pre>	Activate Windows

Figure 61: Add Time and source column

Find the mean of each column for sensor and GAN data to assess how close the values are. Save the file as CSV.



Figure 62: Mean comparison and convert to CSV

Using Boto3, save the file to S3.



Figure 63: save to S3 folder

Similarly, create a Python3 notebook named Sensor\_and\_GAN\_merger\_1\_hr in 'ganand-sensor-data-merge' instance. Import libraries and Stream the 1 hour sensor and GAN data.



Figure 64: 1 hour data import

Create DateTime column and source column. Combine both files.

In [ ]:	#converting the string datatype of time values to datetime
	<pre>sensor_file['timerecorded'] = pd.to_datetime(sensor_file['time']) gan_file['timerecorded'] = pd_to_datetime(gan_file['time'])</pre>
	Baulittel cruelecologo l = barcologoccrue/Baulittel crue l)
In [ ]:	#dropping the redundant column
	<pre>sensor_file.drop(['time'],axis=1,inplace=True) app file.drop(['time'],axis=1,inplace=True)</pre>
	gan_iiie.urop([ time ],axis=1,inplate=irue)
In [ ]:	#sorting by datetime column
	<pre>gan_file.sort_values(by=['timerecorded'],inplace=True) corson file cont values(by=['timerecorded'] inplace=True)</pre>
	Sensor_Tite.sort_varues(by=[ timeretoroed ], inplace frue)
In [ ]:	#quicksight imports all data from all the csv files from the folder and files specified. #Hence, a distinauishina column is reauired.
	an file['course'] = 'CAU'
	sensor_file['source'] = 'SENSOR'
In [ ]:	#merge both datasets
	final file 1 hr = gan file annend(sensor file)
	Activate Window

Figure 65: Adding source and time columns

Save the file as CSV and save the file to S3.

In [ ]:	#reset file index to current dataframe
	<pre>final_file_1_hr.reset_index(drop=True, inplace=True)</pre>
In [ ]:	#save as a csv file locally
	<pre>final_file_1_hr.to_csv("final_data_1_hr.csv")</pre>
In [ ]:	import datetime import tarfile
	<pre>import boto3 # AWS SDK for python. Provides low-level access to AWS services from sagemaker import get_execution_role import sagemaker</pre>
	<pre>m_boto3 = boto3.client('sagemaker')</pre>
	<pre>sess = sagemaker.Session()</pre>
	<pre>region = sess.boto_session.region_name</pre>
	<pre>bucket = 'flood-prediction-master-dataset' # Bucket name where the file is to be saved</pre>
	<pre>print('Using bucket ' + bucket)</pre>
	# save data to 53. key_prefix is the directory path. path is the local file name which will be saved in 53. If that the backet is the bucket name file name as saved locally and bucket is the bucket name Go to Settings to activate
	upload = sess.upload data(path=' <mark>final data 1 hr.csv</mark> ', bucket=bucket, key prefix= <mark>'final-dataset-1-hr</mark> ')

Figure 66: Save 1 Hour Final dataset to S3

Find the mean for each column in Sensor and GAN file to assess how close the values are.



Figure 67: Mean comparison for 1 Hour Data

# 5 QuickSight configuration and Visualization

QuickSight expects a JSON file named as 'manifest.json' which states the list of files to be extracted along with Upload Settings for the same. The below figure states the manifest file for generating a QuickSight visualization for sensor and GAN data for the 15 minutes time period.



Figure 68: manifest file of 15 minutes sensor and GAN data

Upload the file by navigating to the 'final-dataset-15-min' by clicking on Upload.

Amazon S3 > flood-prediction-master-dataset > final-dataset-15-min							
flood-prediction-master-dataset							
Overview							
Q Type a prefix and press Enter to search. Press ESC to clear.							
▲ Upload         ← Create folder         Download         Actions ∨			US East (N. Virginia) 💈	C			
			Viewing 1 to 2				
□ Name ▼	Last modified -	Size 🕶	Storage class -				
final_data_15_min.csv	Aug 12, 2020 2:45:42 PM GMT+0530	337.4 KB	Standard				
manifest.json	Jul 24, 2020 12:12:52 AM GMT+0530	372.0 B	Standard				

Figure 69: S3 Folder of 15 minutes data

The below figure shows the manifest file for 1 hour time period file.

"fileLocations": [
"URIs": [
"https://flood-prediction-master-dataset.s3.amazonaws.com/final-dataset-1-hr/final_data_1_hr.csv"
]
}
],
"globalUploadSettings": {
"format": "CSV",
"delimiter": ",",
"containsHeader": "true"
}
}

Figure 70: Manifest file for 1 hour time period for sensor and GAN data comparison

The 'final-dataset-1-hr' should seem like the figure below.

Amazon S3 > flood-prediction-master-dataset > final-dataset-1-hr flood-prediction-master-dataset							
Overview							
Q Type a prefix and press Enter to search. Press ESC to clear.							
♣ Upload + Create folder Download Actions ~			US East (N. Virginia) 🛛 🤁				
			Viewing 1 to 2				
□ Name ▼	Last modified -	Size 🔻	Storage class 🕶				
final_data_1_hr.csv	Aug 12, 2020 2:43:47 PM GMT+0530	84.2 KB	Standard				
manifest.json	Jul 24, 2020 12:13:36 AM GMT+0530	368.0 B	Standard				

Figure 71: S3 folder of 1 hour data

Click on Services and navigate to the Analytics Section. Click on QuickSight.

aws Services	Re	source Groups 🗸 🔸					ф «	Ghiridhar 🕶 Global 👻 Support 👻	
History S3 Athena	Fir	Id a service by name or feature (for e	examp	le, EC2, S3 or VM, storage).		Analytics	сяĨ	Group A-Z	
Console Home Billing IAM Amazon SageMaker		EC2 Lightsall @ Lambda Batch Elastic Beanstalk	Ŷ	Amazon Managed Blockchain Satellite Ground Station		Athena EMR CloudSearch Elasticsearch Service Kinesis		Alexa for Business Amazon Chime (2 WorkMali Amazon Honeycode	
		Serverless Application Repository AWS Outposts EC2 Image Builder	\$ <b>6</b> 8	Quantum Technologies Amazon Braket		QuickSight C Data Pipeline AWS Data Exchange AWS Glue AWS Lake Formation	Ð	End User Computing WorkSpaces AppStream 2.0 WorkDocs	
		Storage S3 EFS FSx S3 Glacier Storage Gateway AWS Backuin	(HI)	Management & Governance AWS Organizations CloudWatch AWS Auto Scaling CloudFormation CloudTrail	0	MSK Security, Identity, & Compliance IAM Resource Access Manager	Ŷ	WorkLink Internet Of Things IoT Core FreeRTOS /JPT;1/;Click/Winclows	
								Go to Sattings to activate Windows	

Figure 72: QuickSight Menu

The following screens appear only once. A different account has been used to guide the process. Click on 'Sign Up for QuickSight'. The AWS Account number is unique for every account.

QuickSight		
	Your AWS Account is not signed up for QuickSight. Would you like to sign up now? AWS Account 317468511322 Sign up for QuickSight To access QuickSight with a different account, log in again.	
	aws	Activate Windows Go to Settings to activate Windows.

Figure 73: Sign Up to QuickSight

Click on Standard. Choose Enterprise in case you plan to avail those additional services. Click on Continue.

Edition	Standard	O Enterprise
First author with 1GB SPICE	FREE	FREE
Team trial for 60 days (4 authors)*	FREE	FREE
Additional author per month (yearly)**	\$9	\$18
Additional author per month (monthly)**	\$12	\$24
Additional readers (Pay-per-Session)	N/A	\$0.30/session (max \$5/reader/month) ****
Additional SPICE per month	\$0.25 per GB	\$0.38 per GB
Single Sign On with SAML or OpenID Connect	✓	$\checkmark$
Connect to spreadsheets, databases & business apps	✓	$\checkmark$
Access data in Private VPCs		$\checkmark$
Row-level security for dashboards		$\checkmark$
Hourly refresh of SPICE data		$\checkmark$
Secure data encryption at rest		$\checkmark$
Connect to your Active Directory		✓
Use Active Directory Groups ***		$\checkmark$
Send email reports		$\checkmark$
Trial authors are auto-converted to month-to-month subscription up     Each additional author includes 10GB of SPICE capacity     Active Directory groups are available in accounts connected to Activ     Sessions of 30-minute duration. Total charges for each reader are ca	oon trial expiry 2 Directory pped at \$5 per month.Con	ditions apply

Figure 74: Choose Plan

Enter the account name and email address. Check the Amazon S3 to select the buckets to be provided access to QuickSight.

QuickSight region	
Select a region.	0
US East (N. Virginia)	$\sim$
QuickSight account name	
sample-account	0
You will need this for you and others to sign in.	
Notification email address	
ghiridhar2712@rediffmail.com	
For QuickSight to send important notifications.	
Enable invitation by email	
$\checkmark$	
Allow inviting new users by email. This setting cannot be changed after sign-up is complete.	
Inable autodiscovery of data and users in your Amazon Redshift, Amazon RDS, and AWS IAM services.	
🗸 Amazon Athena	
Enables QuickSight access to Amazon Athena databases	
Please ensure the right Amazon S3 buckets are also enabled for QuickSight.	
	Activate Wih
Amazon S3	Choose S3 buckets
Enables Quicksigne to auto-discover your Annazon 55 backets	

Figure 75: Set Account Name and email address

A Sample Bucket has been selected. Click on 'flood-prediction-master-dataset' and click on Finish. This process was already executed by the main account. Since this cannot be reiterated, another account was used for showing these steps.

Select Amazon S3 buckets	>
S3 Buckets Linked To QuickSight Account	S3 Buckets You Can Access Across AWS
elect the buckets that you want QuickSight to be able to access.	
elected buckets have read only permissions by default. However, yo	ou must give write permissions for Athena Workgroup feature.
✓ Select all	
S3 Bucket	Write permission for Athena Workgroup
ghiri-sample-bucket	$\checkmark$
	A 
Cancel	Finish

Figure 76: Choose S3 buckets to provide access

Click on Go to Amazon QuickSight.

QuickSight	t
	Congratulations! You are signed up for Amazon QuickSight!
	Access QuickSight with the following information Account name: sample-ghiri-quicksight
	Go to Amazon QuickSight

Figure 77: Go to QuickSight Page

The below figure shows the QuickSight Analyses tab.

VuickSight	Search for analyses, data s	ets, and dashboards			Q A 31746
★ Favorites	Analyses		La	ast updated (newest first) $\sim$	🗰 🔚 New analysis
() Recent		<b>E</b>			
uu Dashboards					
🗠 Analyses	People Overview analysis	Sales Pipeline analysis	Web and Social Media Anal	Business Review analysis	
Datasets	SAMPLE &	SAMPLE &	SAMPLE 🛱 :	SAMPLE 🕁 :	

#### Figure 78: QuickSight main page

Click on Datasets tab to the left.

	<b>V</b> QuickSight
7	Favorites
C	Recent
đ	1 Dashboards
Þ	∠ Analyses
5	Datasets

Figure 79: Click on Datasets

Click on 'New dataset'.

VuickSight	Search for analyses, data sets, and dashboards			Q	ج 88465
★ Favorites	Datasets			Ne	ew dataset
() Recent	Name		Owner	Last Modified $$	
G Recent	🏟 sample_s3	SPICE	Me	2 days ago	
uu Dashboards	prediction_compare_15_min	SPICE	Me	14 days ago	
🗠 Analyses	🏟 sample_15_min	SPICE	Me	14 days ago	
	prediction_compare_1_hr	SPICE	Me	15 days ago	
Datasets	1_hr_prediction_compare	SPICE	Me	15 days ago	

Figure 80: Adding a New Dataset

Click on 'S3' to enter the manifest path.

QuickSight			ද 88465.
Data Sets			
Create a Data Set FROM NEW DATA SOURCES			
Upload a file (.csv, .tsv, .clf, .elf, .xlsx, .json)	Salesforce Connect to Salesforce	S3 Analytics	
<b>s</b> 3	Athena	RDS	
Redshift Auto-discovered	Redshift Manual connect	MySQL	Activate Windows

Figure 81: S3 as data source

Enter a name for the visualization process. Provide the path for the manifest.json file.

Data source name	
15_min_gan_compare	
Upload a manifest file	🖲 URL 🔿 Upload
rediction-master-dataset.s3.amazonaws.com	n/final- <u>dataset</u> -15-min/ <u>manifest.ison</u>

Figure 82: Provide Manifest file path

Click on Visualize.

Finish data set	creation	×
Table: Estimated table si	15_min_gan_compare 674.9KB	
Data source:	15_min_gan_compare	
Import to SPICE		✓ 1000MB available spice
✓ Email owners wh	en a refresh fails	
Edit/Preview dat	a	Visualize

Figure 83: Visualize

Click on the line chart from the list at the left bottom. Drag and Drop 'timerecorded' column to the X-Axis, 'river' to Value and 'source' to Color. The visualization is created below. QuickSight extracts all the file mentioned in the manifest.json and dumps the data into the SPICE storage (storage of QuickSight). Hence a differentiating column is required. Hence, source column was created. The same process is to be followed for obtaining a visualization for sensor and GAN data with a time period of 1 hour. Click on Print at the top right to print/save the visualization.



Figure 84: Select Visualization and columns

Click on 'Go to Preview'.

$\sim$
$\sim$



Click on 'Print'.

Preview Print Ready to print		
Sum of River by Source and Timerecorded SHOWING TOP 200 IN TIMERECORDED AND BOTTOM 2 IN SOURCE 15	Source	
10	✓ SENSOR	
40 <sup>-100</sup> -10 <sup>-10</sup> -10 <sup>-1</sup>		

Figure 86: Preview Print

A pop up opens which provides an option to print or save as PDF.

Print	1 sheet of paper	
Destination	Microsoft Print to PDF	•
Pages	All	•
Layout	Portrait	•
Color	Color	•
More settings		~
	Print A tiva	te Win ttings to

Figure 87: Print Visual

Print or Save the file as PDF.

File <u>n</u> ame:	sample		~
Save as <u>t</u> ype:	PDF Docum	ient (*.pdf)	~
∧ Hide Folders		Save	

Figure 88: Save the Visual as PDF

### 6 Model Creation, Prediction & Visualization

Create a Python3 file named 'Random\_Forest\_Prediction\_15\_min' inside time-series-algorithms instance. Import libraries and stream the dataset.

<pre>In [ ]: #import Libraries import pandas as pd import pandas as pd import boto3</pre> In [ ]: #import data s3 = boto3.client('s3') bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])		
<pre>import pandas as pd import numpy as np import boto3 In []: #import data s3 = boto3.client('s3') bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>	In [ ]:	#import libraries
<pre>In [ ]: #import data s3 = boto3.client('s3') bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>		import pandas as pd import numpy as np import boto3
<pre>In [ ]: #import data s3 = boto3.client('s3') bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>		
<pre>s3 = boto3.client('s3') bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>	In [ ]:	#import data
<pre>bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>		<pre>s3 = boto3.client('s3')</pre>
<pre>obj = s3.get_object(Bucket= bucket,Key= key) dataset_15min = pd.read_csv(obj['Body'])</pre>		<pre>bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv'</pre>
<pre>dataset_15min = pd.read_csv(obj['Body'])</pre>		obj = s3.get_object(Bucket= bucket,Key= key)
		<pre>dataset_15min = pd.read_csv(obj['Body'])</pre>

Figure 89: Stream 15 Minutes dataset for Random Forest

Remove the index column and rearrange the column order. GAN and sensor data are stored into two files. GAN data's timestamp is incremented by month and appended to the sensor data.

In [ ]:	#drop existing index column	
	<pre>dataset_15min.drop(['Unnamed: 0'],axis=1,inplace=True)</pre>	
	#setting datetime datatype from string. Default is string when reading from csv	
	<pre>dataset_15min['timerecorded'] = pd.to_datetime(dataset_15min['timerecorded'])</pre>	
	#rearranging columns	
	<pre>dataset_15min = dataset_15min[['timerecorded', 'river', 'rain', 'temperature', 'wind_direction', 'wind_speed', 'source']]</pre>	
	#splitting GAN and sensor data	
	<pre>gan_file = dataset_15min.loc[dataset_15min['source']=='GAN'] sensor_file = dataset_15min.loc[dataset_15min['source']=='SENSOR']</pre>	
In [ ]:	#makes the GAN datetime go ahead by 1 month. June - July sensor data. June to august is summer. Hence GAN 1 month ahead.	
	<pre>gan_file['timerecorded'] = gan_file['timerecorded'] + pd.DateOffset(months=1)</pre>	
	#merging both files and resetting index	
	dataset_15min = sensor_file.append(gan_file) dataset_15min.reset_index(drop=True, inplace=True) Activate Window	<i>t</i> s

Figure 90: DateTime conversion and GAN timestamp Incrementation

Time Features are added to the dataset. Dataset is splitted in a ratio of 95:5.

In [ ]:	# adding the datetime column value as a feature. River level being time dependent, datetime column value is saved as # continuous columns.
	<pre>dataset_15min['dayofweek'] = dataset_15min['timerecorded'].dt.dayofweek dataset_15min['hour'] = dataset_15min['timerecorded'].dt.hour</pre>
	<pre>dataset_15min('minute') = dataset_15min('timerecorded').dt.minute dataset_15min('month') = dataset_15min('timerecorded').dt.month dataset_15min('year') = dataset_15min('timerecorded').dt.year</pre>
	<pre>dataset_15min['dayofmonth'] = dataset_15min['timerecorded'].dt.day dataset_15min['dayofyear'] = dataset_15min['timerecorded'].dt.dayofyear</pre>
In [ ]:	dataset_15min.shape
In [ ]:	#splitting into train and test dataset
	train_dataset = dataset_15min[:5240] test_dataset = dataset_15min[5240]

Figure 91: Data Splitting for Random Forest of 15 Minutes Time Period

Training and testing files are saved as CSV and bucket name is provided to Boto3.



Figure 92: Saving Files as CSV

Training and Testing datasets are uploaded to S3. SageMaker ML either accepts streaming data or S3 path as input.



Figure 93: Uploading files to S3

Random Forest is Scripted based on SageMaker Python SDK<sup>3</sup>. Libraries are imported and a model is loaded if already present. Arguments are defined for the script.

 $<sup>^{3}</sup>$ https://sagemaker.readthedocs.io/en/stable/frameworks/sklearn/using\_sklearn.html

In [ ]:	%%writefile rftimeseries15min.py
	#doing by scripting
	import argparse
	import os
	import numpy as np import pandas as pd from sklearn.ensemble import RandomForestRegressor import joblib
	<pre>def model_fn(model_dir): clf = joblib.load(os.path.join(model_dir, "rfmodel.joblib")) return clf</pre>
	ifname =='main':
	<pre>print('extracting arguments') parser = argparse.ArgumentParser()</pre>
	<pre># hyperparameters sent by the client are passed as command-line arguments to the script.</pre>
	parser.add_argument('n-estimators', type=int, default=1500) parser.add_argument('max-leaf-nodes', type=int, default=15)

Figure 94: Scripting the Random Forest algorithm - Model Loading

The environmental variables are used to retrieve the Datasets and models within the AWS EC2 ML Instance. The datasets are retrieved and unnecessary columns are removed.



Figure 95: Using Environmental Variables to retrieve the required files for prediction

Training and prediction is defined in the script. Model is saved to a ML Instance Folder.



Figure 96: Training, Prediction and Saving Model

The Sagemaker estimator is provided the script, instance type and script argument values. Datasets and Training is executed. Estimator is passed the Environment Folder where the model persists.

In [ ]:	# use of Estimator from the SageMaker Python SDK. stating the script and hyperparameters	
	from sagemaker.sklearn.estimator import SKLearn	
	<pre>sklearn_estimator = SKLearn( entry_point='rftimeseries15min.py', role = get_executon_role(), train_instance_tope='ml.md.xlarge', framework_version='0.23-1', base_job_name='randomforest-15-min', hyperparameters = {</pre>	
In [ ]:	# launch training job, with asynchronous call	
	<pre>sklearn_estimator.fit({['train':trainpath, 'test': testpath], wait=False)</pre>	
In [ ]:	# after training the model is created which is used for prediction. Here the model is generated. The path is	displayed.
	<pre>sklearn_estimator.latest_training_job.wait(logs='None') artifact = m boto3.describe training job(</pre>	
	TrainingJobName=sklearn_estimator.latest_training_job.name)['ModelArtifacts']['S3ModelArtifacts']	Activate Win
	<pre>print('Model artifact persisted at ' + artifact)</pre>	Go to Settings to

Figure 97: Random Forest Model Deployment

The model is deployed to an AWS Endpoint. Test dataset is passed to get the predictions. The timestamp column values from the test dataset and predictions from the algorithms are saved as a dataframe. A source column is created with value 'RF' to identify the prediction source. The file is saved to S3.



Figure 98: Prediction and saving it to S3

Create a Python3 file named 'XGBoost\_Prediction\_15\_min' inside time-series-algorithms instance. Import libraries and stream the dataset.

In [ ]:	#import libraries
	import pandas as pd import numpy as np import boto3
In [ ]:	#import data
	<pre>s3 = boto3.client('s3')</pre>
	<pre>bucket = 'flood-prediction-master-dataset' key = 'final-dataset-15-min/final_data_15_min.csv'</pre>
	<pre>obj = s3.get_object(Bucket= bucket,Key= key)</pre>
	<pre>dataset_15min = pd.read_csv(obj['Body'])</pre>

Figure 99: Stream 15 Minutes Data for XGBoost

The below figures shows the process of streaming the data, incrementing the GAN timestamp value, feature generation and splitting the file in a 95:5 ratio.



Figure 100: DateTime conversion and GAN Timestamp incrementation

In [ ]:	<pre># adding the datetime column value as a feature. River level being time dependent, datetime column value is # continuous columns.</pre>	saved as
	<pre>dataset_15min['dayofweek'] = dataset_15min['timerecorded'].dt.dayofweek dataset_15min['hour'] = dataset_15min['timerecorded'].dt.hour dataset_15min['minute'] = dataset_15min['timerecorded'].dt.minute dataset_15min['month'] = dataset_15min['timerecorded'].dt.month dataset_15min['dayofmonth'] = dataset_15min['timerecorded'].dt.day dataset_15min['dayofmonth'] = dataset_15min['timerecorded'].dt.day dataset_15min['dayofyear'] = dataset_15min['timerecorded'].dt.day</pre>	
In [ ]:	dataset_15min.shape	
In [ ]:	<pre>train_dataset = dataset_15min[:5240] test_dataset = dataset_15min[5240:]</pre>	
In [ ]:	<pre>train_dataset.head()</pre>	
In [ ]:	<pre>test_dataset.head()</pre>	
In [ ]:	# removing dependent columns from test dataset. timerecorded is not required for prediction but for further	processes.
	<pre>y_test = test_dataset[['timerecorded', 'river']] test_dataset.drop(['timerecorded', 'river'],axis=1,inplace=True)</pre>	Activate
	<pre># converting training and testing datasets into csv files train_dataset.drop(['timerecorded','source'],axis=1,inplace=True) train_dataset.to_csv('train.csv',header-Mone,index-False)</pre>	Go to Settin

Figure 101: Feature Generation and File Splitting

Import libraries, set the source bucket and select an AWS container comprising of XG-Boost algorithm. AWS provides containers with preloaded XGBoost algorithm. Training and deployment processes are not benefitted but the scripting time is saved.



Figure 102: Choose AWS XGBoost Containers

Instance type, Hyperparameters and path to the datasets are provided.

In [ ]:	# setting hyperparameters, bucket and session data
	<pre>sess = sagemaker.Session()</pre>
	<pre>xgb = sagemaker.estimator.Estimator(containers[my_region], role, train_instance_count=1,</pre>
	<pre>train_instance_type='mL.m4.xlarge', output_path='s3://{}/{}/output'.format(bucket_name, prefix), sagemaker session=sess)</pre>
	xgb.set_hyperparameters(eta=0.06,
	<pre>silent=0, early_stopping_rounds=5, objective='reg:linear', num_round=1000)</pre>
In [ ]:	<pre># saving data to S3. SageMaker will take training data from s3 boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')</pre>
	<pre>trainpath = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')</pre>
T. F. 1.	
τυ [ ]:	<pre>S3 = DOTO3.LILENT( S3 ) S3.get_object(Bucket=bucket_name)</pre>

Figure 103: Uploading Datasets to S3 and defining the estimator

The model is trained and deployed.

<pre>In [ ]: # training the model xgb.fit({'train': trainpath}) In [ ]: # deploying to a endpoint xgb_predictor = xgb.deploy(initial_instance_count-1,instance_type=['ml.m4.xlarge')) In [ ]: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				
<pre>xgb.fit({'train': trainpath}) In []: # deploying to a endpoint xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type="ml.m4.xlarge") In []: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In []: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>	Ir	1	]:	# training the model
<pre>In [ ]: # deploying to a endpoint xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type=['ml.m4.xlarge') In [ ]: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				<pre>xgb.fit({'train': trainpath})</pre>
<pre>In []: # deploying to a endpoint xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type=[ml.m4.xlarge') In []: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In []: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				
<pre>xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type=[ml.m4.xlarge') In []: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In []: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>	Ir	C	]:	# deploying to a endpoint
<pre>In [ ]: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				<pre>xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')</pre>
<pre>In [ ]: xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				
<pre>In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>	Ir	[	]:	<pre>xgb_predictor.content_type = 'text/csv' # set the data type for an inference xgb_predictor.serializer = csv_serializer # set the serializer type</pre>
<pre>In [ ]: # removing unrequired columns test_dataset.drop(['source'],axis=1,inplace=True)</pre>				
<pre>test_dataset.drop(['source'],axis=1,inplace=True)</pre>	Ir	1	]:	# removing unrequired columns
				<pre>test_dataset.drop(['source'],axis=1,inplace=True)</pre>

Figure 104: XGBoost Training and Deployment 15 Minutes Time Period Data

The prediction is obtained which is combined with the test dataset timestamp. The 'source' column is created with value 'XGB'. The test dataset is added a 'source' column with value 'ACTUAL'. Both files are saved to S3.

In [ ]:	<pre>#predictions contains ML predictions #see how to add column name to prediction output predictions = xgb_predictor.predict(test_dataset.values).decode('utf-8') # prediction predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array</pre>
	<pre>outcome = pd.DataFrame(predictions_array) outcome.rename(columns={0:"river"},inplace=True)</pre>
	<pre>rf_final = y_test['timerecorded'].to_frame() rf_final.reset_index(drop=True,inplace=True) rf_final['river'] = outcome['river'].astype(float) rf_final['source'] = 'XGB'</pre>
	<pre>rf_final.to_csv("xgboost_predictions_15_min.csv")</pre>
	<pre>y_test['source'] = 'ACTUAL'</pre>
	<pre>y_test.to_csv("actual_15_min.csv")</pre>
	<pre>sess.upload_data(     path='xgboost_predictions_15_min.csv', bucket=bucket_name,     key_prefix='predictions-15-min')</pre>
	<pre>sess.upload_data(     path='actual_15_min.csv', bucket=bucket_name,     key_prefix='predictions-15-min')</pre>
	<pre>print("Success!")</pre>

Figure 105: Predicting and Uploading the file to S3

Manifest file for visualizing the actual river level and algorithm predictions for 15 minutes time period is provided below.

```
{
"fileLocations": [
{
    "URIs": [
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/actual_15_min.csv",
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/random_forest_predictions_15_min.csv",
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/random_forest_predictions_15_min.csv",
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/random_forest_predictions_15_min.csv",
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/random_forest_predictions_15_min.csv",
    "https://flood-prediction-master-dataset.s3.amazonaws.com/predictions-15-min/random_forest_predictions_15_min.csv"
    ]
}
// "globalUploadSettings": {
    "format": "CSV",
    "delimiter": ",",
    "containsHeader": "true"
}
```



Below figure shows the 'predictions-15-min' folder wherein three datasets and manifest file is present.

Amazon S3 > flood-prediction-master-dataset > predictions-15-min			
flood-prediction-master-dataset			
Overview			
Q Type a prefix and press Enter to search. Press ESC to clear.			
▲ Upload + Create folder Download Actions ~			US East (N. Virginia)
			Viewing 1 to 5
Name -	Last modified 🕶	Size 💌	Storage class 👻
🗌 📂 train			
actual_15_min.csv	Jul 31, 2020 8:45:04 PM GMT+0530	8.0 KB	Standard
manifest.json	Jul 31, 2020 8:16:29 PM GMT+0530	598.0 B	Standard
random_forest_predictions_15_min.csv	Jul 31, 2020 8:45:05 PM GMT+0530	7.9 KB	Standard
xgboost_predictions_15_min.csv	Jul 31, 2020 8:45:05 PM GMT+0530	8.2 KB	Standard

Figure 107: S3 bucket folder of 15 minute predictions

Provide QuickSight the manifest file's link and configure the QuickSight as shown below.

	+ Ŋ C <sup>1</sup> Add Undo Redo	★ prediction_compare_15_min analysis Autosave ON ∨	Print Capture
Visualize	Data set	Field wells           X axis         Value         Color	
<b>V</b> Filter	Fields list Q # ColumnId-1	timerecorded V Source source	~
Story	# river     source	Sheet 1 V +	
<b>↓↓↑</b> Parameters	C timerecorded	SHOWING TOP 200 IN TIMERECORDED AND BOTTOM 4 IN SOURCE  O.015K  Source  Source  ···	
Actions	Visual types ✓ ↓↑ ∩ O ●		
Themes	을 해 드 가 드		
© Settings			Activate Windows
			Go to Settings to activate V

Figure 108: Visualization of 15 Minutes predictions

Create a Python3 file named 'XGBoost\_Prediction\_1\_hr' inside time-series-algorithms instance. The below code predicts the river level based on 1 hour dataset and follows the same flow as in the above XGBoost file.

In [ ]:	#import Libraries
	import pandas as pd import numpy as np import boto3
In [ ]:	#import data
	<pre>s3 = boto3.client('s3')</pre>
	<pre>bucket = 'flood-prediction-master-dataset' key = 'final-dataset-1-hr/final_data_1_hr.csv'</pre>
	<pre>obj = s3.get_object(Bucket= bucket,Key= key)</pre>
	<pre>dataset_1hr = pd.read_csv(obj['Body'])</pre>

Figure 109: Streaming the 1 hour time period datasets for XGBoost

In [ ]:	#drop existing index column
	<pre>dataset_1hr.drop(['Unnamed: 0'],axis=1,inplace=True)</pre>
	#setting datetime datatype from string. Default is string when reading from csv
	<pre>dataset_1hr['timerecorded'] = pd.to_datetime(dataset_1hr['timerecorded'])</pre>
	#rearranging columns
	<pre>dataset_1hr = dataset_1hr[['timerecorded', 'river', 'rain', 'temperature', 'wind_direction', 'wind_speed', 'source']]</pre>
	#splitting GAN and sensor data
	<pre>gan_file = dataset_lhr.loc[dataset_lhr['source']=='GAN'] senson_file = dataset_lhr.loc[dataset_lhr['source']=='SENSOR']</pre>
In [ ]:	#makes the GAN datetime go ahead by 1 month. June - July sensor data. June to august is summer. Hence GAN 1 month ahead.
	<pre>gan_file['timerecorded'] = gan_file['timerecorded'] + pd.DateOffset(months=1)</pre>
	#merging both files and resetting index
	dataset_1hr = sensor_file.append(gan_file) dataset_1hr.reset_index(drop=True, inplace=True)

Figure 110: DateTime conversion and GAN Timestamp incrementation

In [ ]:	<pre># adding the datetime column value as a feature. River level being time dependent, datetime column value is # continuous columns.</pre>	saved as
	<pre>dataset_1hr['dayofweek'] = dataset_1hr['timerecorded'].dt.dayofweek</pre>	
	<pre>dataset_1hr['hour'] = dataset_1hr['timerecorded'].dt.hour</pre>	
	dataset_hhp['minute'] = dataset_hhp['timerecorded'].dt minute	
	dataset_inr[_month] = dataset_inr[_timerecorded j.dt.month	
	dataset 1hr['dayofmonth'] = dataset 1hr['timerecorded'].dt.day	
	dataset_1hr['dayofyear'] = dataset_1hr['timerecorded'].dt.dayofyear	
In [ ]:	dataset_1hr.shape	
In [ ]:	<pre>train_dataset = dataset_1hr[:1315]</pre>	
	<pre>test_dataset = dataset_1hr[1315:]</pre>	
In [ ]:	<pre>train_dataset.tail()</pre>	
In [ ]:	test_dataset.head()	
In [ ]:	# removing dependent columns from test dataset. timerecorded is not required for prediction but for further	processes.
	v test - test detest[['timescanded' 'niven']]	
	<pre>cst_=cst_ustast([cimerecorded', 'river'])atis=1.inplace=True)</pre>	
		Activate
	# converting training and testing datasets into csv files	Go to Setti
	<pre>train_dataset.drop(['timerecorded','source'],axis=1,inplace=True)</pre>	
	<pre>train dataset.to csv("train.csv",header=None,index=False)</pre>	

#### Figure 111: Feature Generation and Splitting



Figure 112: Choosing the AWS XGBoost Containers



Figure 113: Defining the estimators, uploading the files and training the model



Figure 114: Deploying the XGBoost model

In [ ]:	<pre>#predictions contains ML predictions #see how to add column name to prediction output predictions = xgb_predictor.predict(test_dataset.values).decode('utf-8') # prediction predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array</pre>
	outcome = pd.DataFrame(predictions_array) outcome.rename(columns={0:"river"},inplace=True)
	<pre>rf_final = y_test['timerecorded'].to_frame() rf_final.reset_index(drop=True,inplace=True) rf_final['river'] = outcome['river'].astype(float) rf_final['source'] = 'XGB'</pre>
	<pre>rf_final.to_csv("xgboost_predictions_1_hr.csv")</pre>
	<pre>y_test['sounce'] = 'ACTUAL'</pre>
	<pre>y_test.to_csv("actual_1_hr.csv")</pre>
	<pre>sess.upload_data(     path='xgboost_predictions_1_hr.csv', bucket=bucket_name,     key_prefix='predictions-1-hr')</pre>
	<pre>sess.upload_data(     path='actual_1_hr.csv', bucket=bucket_name,     key_prefix='predictions-1-hr')</pre>
	print("Success!")

Figure 115: Predicting and uploading the file to S3

Create a Python3 file named 'Random\_Forest\_Prediction\_1\_hr' inside time-series-algorithms instance. The below code predicts the river level based on 1 hour dataset and follows the same flow as in the above Random Forest file.

	In [ ]:	#import libraries
		import pandas as pd import numpy as np import boto3
Î		
	In [ ]:	#import data
		<pre>s3 = boto3.client('s3')</pre>
		<pre>bucket = 'flood-prediction-master-dataset'</pre>
		<pre>key = 'final-dataset-1-hr/final_data_1_hr.csv'</pre>
		obj = s3.get_object(Bucket= bucket,Key= key)
		<pre>dataset_1hr = pd.read_csv(obj['Body'])</pre>

Figure 116: Streaming the 1 hour time period dataset for Random Forest

In [ ]:	#drop existing index column
	<pre>dataset_1hr.drop(['Unnamed: 0'],axis=1,inplace=True)</pre>
	#setting datetime datatype from string. Default is string when reading from csv
	<pre>dataset_1hr['timerecorded'] = pd.to_datetime(dataset_1hr['timerecorded'])</pre>
	#rearranging columns
	<pre>dataset_1hr = dataset_1hr[['timerecorded', 'river', 'rain', 'temperature', 'wind_direction', 'wind_speed', 'source']]</pre>
	#splitting GAN and sensor data
	<pre>gan_file = dataset_lhr.loc[dataset_lhr['source']=='GAN'] sensor_file = dataset_lhr.loc[dataset_lhr['source']=='SENSOR']</pre>
In [ ]:	#makes the GAN datetime go ahead by 1 month. June - July sensor data. June to august is summer. Hence GAN 1 month ahead
	<pre>gan_file['timerecorded'] = gan_file['timerecorded'] + pd.DateOffset(months=1)</pre>
	#merging both files and resetting index
	dataset_1hr = sensor_file.append(gan_file) dataset_1hr_reset_index(dron=True_inplace=True)
	Activate Activate

#### Figure 117: DateTime conversion and GAN Timestamp incrementation



#### Figure 118: Feature Generation and Dataset Splitting



Figure 119: Uploading file to S3

In [ ]:	%%writefile rftimeseries1hr.py
	#doing by scripting
	import argparse import os
	import numpy as np import pandas as pd from sklearn.ensemble import RandomForestRegressor import joblib
	<pre>def model_fn(model_dir):     clf = joblib.load(os.path.join(model_dir, "rfmodel.joblib"))     return clf</pre>
	ifname =='main':
	print('extracting arguments') parser = argparse.ArgumentParser()
	<pre># hyperparameters sent by the client are passed as command-line arguments to the script.</pre>
	<pre>parser.add_argument('n-estimators', type=int, default=1500) parser.add_argument('max-leaf-nodes', type=int, default=15)</pre>
	<pre># Data, model, and output directories parser.add_argument('model_dir', type=str, default=os.environ.get('SM_MODEL_DIR')) parser.add_argument('train', type=str, default=os.environ.get('SM_CHANNEL_TRAIN')) parser.add argument('test', type=str, default=os.environ.get('SM_CHANNEL_TEST'))</pre>

Figure 120: Scripting the 1 hour time period Random Forest



#### Figure 121: Creating Script arguments and splitting dataset



Figure 122: Random Forest Training, Model Creation and Storage



Figure 123: Random Forest for 1 Hour Time Period Training and Deployment

In [ ]:	# An EC2 model is deployed based on the script and model
	<pre>predictor = sklearn_estimator.deploy(instance_type='ml.m4.xlarge',initial_instance_count=1)</pre>
In [ ]:	# removing unrequired columns
	<pre>test_dataset.drop(['source'],axis=1,inplace=True)</pre>
In [ ]:	# "outcome" contains ML predictions. rf_final has the datetime value for each prediction taken from y_test. # "rf_final" is then provided prediction values saved as a column. Also source column states the algorithm name. # By just counting the number of predictions above fload level, a better algorithm can be decided, # but the datetime column will help analyze the delay between two algorithms.
	<pre>outcome = pd.DataFrame(predictor.predict(test_dataset)) outcome.rename(columns={0:"river"},inplace=True)</pre>
	<pre>rf_final = y_test['timerecorded'].to_frame() rf_final.reset_index(drop=True,inplace=True) rf_final['river'] = outcome['river'].astype(float) rf_final['source'] = 'RF'</pre>
	<pre># saving as a csv file locally rf_final.to_csv("random_forest_predictions_1_hr.csv")</pre>
	<pre># saving file to s3 sess.upload_data(     path='random_forest_predictions_1_hr.csv', bucket=bucket,     key_prefix='predictions-1-hr')</pre>
	print("Success!")

Figure 124: Random Forest Prediction and Upload to S3

Manifest file for visualizing the actual river level and algorithm predictions for 1 hour time period is provided below. Follow the same process as above to generate a QuickSight visualization.



Figure 125: Manifest File for 1 hour prediction data visualization

## 7 Flood Trigger Evaluation

The Environmental agency also provides publicly the flood warning trigger level for each river. Based on this value, the Evaluation of the prediction is assessed.



Figure 126: Flood Trigger River Level Value

Create a Python3 file named 'Flood\_Trigger\_Comparison\_1\_Hour' inside the timeseries-algorithms instance. Import the libraries and stream the CSV files from the S3 'predictions-1-hr' folder.

In [ ]:	<pre>import pandas as pd import numpy as np from sklearn.metrics import r2_score,mean_absolute_error import boto3</pre>
In [ ]:	#importing predicted data
	<pre>s3 = boto3.client('s3')</pre>
	<pre>bucket = 'flood-prediction-master-dataset'</pre>
	<pre>key = 'predictions-1-hr/actual_1_hr.csv' obj = s3.get_object(Bucket= bucket,Key= key) actual = pd.read_csv(obj['Body'])</pre>
	<pre>key = 'predictions-1-hr/xgboost_predictions_1_hr.csv' obj = s3.get_object(Bucket= bucket,Key= key) xgb_predict = pd.read_csv(obj['Body'])</pre>
	<pre>key = 'predictions-1-hr/random_forest_predictions_1_hr.csv' obj = s3.get_object(Bucket= bucket,Key= key) rf_predict = pd.read_csv(obj['Body'])</pre>

Figure 127: Importing 1 Hour Time Period Files

Difference between actual and prediction values (can also be termed as Prediction Error) is saved for both algorithms separately. The sum and max of prediction error for

both algorithms is calculated. Also the first flood trigger time is calculated. If the value is within 6 hours, then flash flood prediction is successful.



Figure 128: Assessing Prediction Error

All three datasets are joined without any join condition since all datasets have the same number of records and timestamp. Columns, either actual or prediction greater than 6.2 are retrieved and stored to a different variable. Status column is added to define the validity of the trigger by the predictions.



Figure 129: Extracting the Flood Triggered Records

Status is set to HIT, MISS or FALSE. HIT means the prediction triggered rightly, MISS means prediction missed the flood trigger and FALSE means the Trigger is erroneous.

Figure 130: HIT, MISS and Flase Triggers

This status column for both algorithms is assessed using PASS evaluation to determine the performance of the algorithms. Precision, Accuracy, Specificity and Sensitivity (PASS). R-Square value is also calculate to assess its value with respect to the PASS performance.



Figure 131: PASS Evaluation

Create a Python3 file named 'Flood\_Trigger\_Comparison\_15\_Min' inside the timeseries-algorithms instance. Import the libraries and stream the CSV files from the S3 'predictions-15-mins' folder. The below figures examine the predictions for 1 hour time period as performed above.

In [ ]:	<pre>import pandas as pd import numpy as np from sklearn.metrics import r2_score,mean_absolute_error import boto3</pre>
In [ ]:	#importing predicted data
	<pre>s3 = boto3.client('s3')</pre>
	<pre>bucket = 'flood-prediction-master-dataset'</pre>
	<pre>key = 'predictions-15-min/actual_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) actual = pd.read_csv(obj['Body'])</pre>
	<pre>key = 'predictions-15-min/xgboost_predictions_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) xgb_predict = pd.read_csv(obj['Body'])</pre>
	<pre>key = 'predictions-15-min/random_forest_predictions_15_min.csv' obj = s3.get_object(Bucket= bucket,Key= key) rf_predict = pd.read_csv(obj['Body'])</pre>

Figure 132: Streaming 15 Minutes Time Period Files

Tn [ ]:	#dronning unnevessary columns
TO 1 1.	nd opping unickessury coounns
	<pre>actual.drop(['Unnamed: 0','source'],axis=1,inplace=True) web predict dep(('Unpaged, 0', 'source'] avis 1 implace True)</pre>
	<pre>rf_predict.drop(['Unnamed: 0', 'source'],axis=1,inplace=True)</pre>
In [ ]:	actual
In [ ]:	xgb_predict
In [ ]:	rf_predict

Figure 133: Prediction Error in 15 Minutes Predictions



Figure 134: Extracting Flood Triggered Records



#### Figure 135: HIT, MISS and FALSE triggers



Figure 136: PASS Evaluation on 15 Minutes Predictions

### 8 Results

The visualization of GAN and sensor data is presented below. It shows that the distribution of the values is similar to that of the sensor dataset. The GAN values are not similar but near enough to mimic the sensor values.



Figure 137: GAN and Sensor Values Comparison for 15 minutes time period

The below graph displays the graph between sensor and GAN data with a time period of 1 hour. These two graphs show good ability of the GAN to imitate the data. But there is one improvement that is yet to be addressed in GAN.



Figure 138: GAN and Sensor Values Comparison for 1 hour time period

The below graph shows the comparison between actual and prediction values for 15 minutes time period. The predictions by both algorithms is very close to the actual value. From the graph it can be concluded that the accuracy of both the algorithms is very high. The trend of the graph is not smooth as the sensor data. The GAN can imitate the distribution of the source data but could not really mimic the trend or smoothness of the sensor data.



Figure 139: 15 Minutes Predictions and actual values

The below graph is the visualization between actual and prediction values for 1 hour time period. This graph shows distinct gap between actual and prediction values. The accuracy is high but is not that high as in 15 minutes time period. This can be due to less trend details - for 2 hours of data, 1 hour time period has 2 records whereas 15 minutes time period has 8 records. Probably more historical data can assist in understanding the trend and improving the performance of the models.

Since this dataset has all the required parameters like rainfall, temperature, wind, etc. and also a desirable time period, it was an ideal dataset for this research. Since the dataset download was restricted to 1 month by the API, more data could not be obtained.



Figure 140: 1 Hour Predictions and actual values

A mission critical prediction model cannot be judged solely based on Statistical Tests. The extent of fit between the forecast and prediction conveys least on the errors and achievements of the prediction model. Accuracy is the extent of error in the prediction. This is assessed by finding the sum of the difference between actual and predicted values (can be termed as Prediction Error). Sensitivity/Efficiency in this scenario is assessed based on the number of flood warnings triggered correctly. An algorithm can be efficient to trigger the flood warning but should not be erroneous. Specificity/Reliability is assessed based on the number of erroneous flood warnings triggered. Precision is the number of accurate warnings triggered divided by the actual number of warnings. Influenced by Furquim et al. (2018), the below table summarises these details for both algorithms on both time periods. As per Hagen et al. (2020), although the statistical tests indicate that the model is a very accurate fit, but the hit and miss rates of the algorithm conveys room for improvement.

Sum of Prediction Error (Accuracy): XGBoost: -7.98 Random Forest: 21.58
Highest Prediction Error: XGBoost: 0.38 Random Forest: 0.74
Total Number of Flood Warning Triggers: 103
Total Correct Flood Warnings Triggered (Sensitivity): XGBoost: 102 Random Forest: 94
Total Flood Warnings missed: XGBoost: 1 Random Forest: 9
Total Erroneous Flood Warnings Triggered (Specificity): XGBoost: 0 Random Forest: 0
Precision: XGBoost: 99.03 Random Forest: 91.26
R-Square Score: XGBoost: 99.4 Random Forest: 98.75

Table 1: 15 Minute Time Period

Sum of Prediction Error (Accuracy): XGBoost: -1.86 Random Forest: 24.79
Highest Prediction Error: XGBoost: 1.32 Random Forest: 1.60
Total Number of Flood Warning Triggers: 29
Total Correct Flood Warnings Triggered (Sensitivity): XGBoost: 27 Random Forest: 21
Total Flood Warnings missed: XGBoost: 2 Random Forest: 8
Total Erroneous Flood Warnings Triggered (Specificity): XGBoost: 1 Random Forest: 1
Precision: XGBoost: 93.10 Random Forest: 72.41
R-Square Score: XGBoost: 94.01 Random Forest: 85.75

Table 2: 1 Hour Time Period

Figure 141: PASS Evaluation Table

From the above table it is clear that XGBoost has outperformed Random Forest in all aspects. It has saved time and lives of the people. As evident from the graph, the accuracy of 15 minutes time period is greater than the 1 hour time period. Hence, with increasing time period, the accuracy decreases possibly due to less detail of trend. Neural Networks were not implemented due to time and complexity issues. Also a historical dataset with more number of records would enable better accuracy of models. Use of Data assimilation would be required. As mentioned in Hu et al. (2019), as the number of historical records increases with a small time period, there would be instances where river level has least to no change which distort the trend. Hence removal of those field and using Data Assimilation would be required. Also, use of ensemble data in Hagen et al. (2020) enabled prediction beyond one week.

Flash Flood Prediction as well as flood prediction for about 3 days was achieved. Also, XGBoost was implemented for the first time in flood prediction domain which has outperformed Random Forest, a popular algorithm used for flood prediction.

### References

- Furquim, G., Filho, G. P. R., Jalali, R., Pessin, G., Pazzi, R. W. & Ueyama, J. (2018), 'How to improve fault tolerance in disaster predictions: A case study about flash floods using iot, ml and real data', *Sensors* 18(3). Impact Factor = 2.475. URL: https://www.mdpi.com/1424-8220/18/3/907
- Hagen, J. S., Cutler, A., Trambauer, P., Weerts, A., Suarez, P. & Solomatine, D. (2020),
  'Development and evaluation of flood forecasting models for forecast-based financing using a novel model suitability matrix', *Progress in Disaster Science* 6, 100076. Impact Factor = 2.1.
  URL: http://www.sciencedirect.com/science/article/pii/S2590061720300132
- Hu, R., Fang, F., Pain, C. & Navon, I. (2019), 'Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method', *Journal of Hydrology* 575, 911 920. Impact Factor = 3.73.

URL: http://www.sciencedirect.com/science/article/pii/S0022169419305323