

Smart Bridge-Remote Summer Internship Program

Project Title : **Prediction of Floods using Environmental
Analysis**

Project ID : **SPS_PRO_359**

Developed by: Apeksha B, Chinmaya G R, Kruthika C S, Madhumati J B, Veena M R

1. ABSTRACT

Floods are among the most destructive natural disasters, which are highly complex to model. The research on the advancement of flood prediction models contributed to risk reduction, policy suggestion, minimization of the loss of human life, and reduction the property damage associated with floods. To mimic the complex mathematical expressions of physical processes of floods, during the past two decades, machine learning (ML) methods contributed highly in the advancement of prediction systems providing better performance and cost-effective solutions. Due to the vast benefits and potential of ML, its popularity dramatically increased among hydrologists. Researchers through introducing novel ML methods and hybridizing of the existing ones aim at discovering more accurate and efficient prediction models. The main contribution of this paper is to demonstrate the state of the art of ML models in flood prediction and to give insight into the most suitable models. In this paper, the literature where ML models were benchmarked through a qualitative analysis of robustness, accuracy, effectiveness, and speed are particularly investigated to provide an extensive overview on the various ML algorithms used in the field. The performance comparison of ML models presents an in-depth understanding of the different techniques within the framework of a comprehensive evaluation and discussion. As a result, this paper introduces the most promising prediction methods for both long-term and short-term floods. Furthermore, the major trends in improving the quality of the flood prediction models are investigated. Among them, hybridization, data decomposition, algorithm ensemble, and model optimization are reported as the most effective strategies for the improvement of ML methods. This survey can be used as a guideline for hydrologists as well as climate scientists in choosing the proper ML method according to the prediction task.

The Godavari is India's second longest river after the Ganga. Its source is in Triambakeshwar, Maharashtra. Hydraulic model can be used to predict the consequences of flooding events. In this project, the hydraulic model of Godavari river near Bhadrachalam was constructed using the software Hydrologic Engineering Center-River Analysis System (HEC-RAS). The model include (i) a 1-dimensional (1D) model, where river and floodplain flow is modelled in 1D, and (ii) a pure 2D model, where river and floodplain flow is modelled in 2D. Important differences between data requirements, pre-processing, model set-up and results were highlighted and summarized, and a rough guide that may be used when deciding the appropriate type of model for a project, was presented. In addition, the subgrid technique used in 2D HEC-RAS modeling was studied by investigating the influence of computational mesh structure and coupling between 1D and 2D areas. The results showed that a model could successfully reproduce a historic flooding event. The 2D model could also provide more detailed information regarding flood propagation and velocities on the floodplain. The results obtained from HEC-RAS are then used for providing various mitigation measures such as modifying the cross-section, design of flood embankments, for various river training works etc.

2. INTRODUCTION

Among the natural disasters, floods are the most destructive, causing massive damage to human life, infrastructure, agriculture, and the socioeconomic system. Governments, therefore, are under pressure to develop reliable and accurate maps of flood risk areas and further plan for sustainable flood risk management focusing on prevention, protection, and preparedness. Flood prediction models are of significant importance for hazard assessment and extreme event management. Robust and accurate prediction contribute highly to water recourse management strategies, policy suggestions and analysis, and further evacuation modelling. Thus, the importance of advanced systems for short-term and long-term prediction for flood and other hydrological events is strongly emphasized to alleviate damage. However, the prediction of flood lead time and occurrence location is fundamentally complex due to the dynamic nature of climate condition. Therefore, today's major flood prediction models are mainly data-specific and involve various simplified assumptions. Thus, to mimic the complex mathematical expressions of physical processes and basin behavior, such models benefit from specific techniques e.g., event-driven, empirical black box, lumped and distributed, stochastic, deterministic, continuous, and hybrids

The Godavari is India's second longest river after the Ganga. Its source is in Triambakeshwar, Maharashtra. It flows east for 1,465 kilometres (910 mi) draining the

states of Maharashtra (48.6%), Telangana (18.8%), Andhra Pradesh (4.5%), Chhattisgarh (10.9%), Madhya Pradesh (10.0%), Odisha (5.7%), Karnataka (1.4%) and Puducherry (Yanam) and emptying into Bay of Bengal through its extensive network of tributaries. Measuring up to 312,812 km² (120,777 sq mi), it forms one of the largest river basins in the Indian subcontinent, with only the Ganges and Indus rivers having a larger drainage basin. In terms of length, catchment area and discharge, the Godavari river is the largest in peninsular India, and had been dubbed as the Dakshina Ganga – Ganges of the South.

The river has been revered in Hindu scriptures for many millennia and continues to harbour and nourish a rich cultural heritage. In the past few decades, the river has been barricaded by a number of barrages and dams, restricting its flow. The river delta supports 729 persons/km², and has been categorized as having substantial to greater risk of flooding with rising sea levels.

Godavari enters into Telangana in Nizamabad district at Kandakurthy where Manjira, Haridra Rivers joins Godavari and forms Triveni Sangamam. The river flows along the border between Nirmal and Mancherial districts in the north and Nizamabad, Jagityal, Peddapalli Ramagundam districts to its south. About 12 km (7.5 mi) after entering Telangana it merges with the backwaters of the Sriram Sagar Dam. The river after emerging through the dam gates enjoys a wide river bed, often splitting to encase sandy islands. The river further swells after receiving a minor tributary Kinnerasani River and exits into Andhra Pradesh.

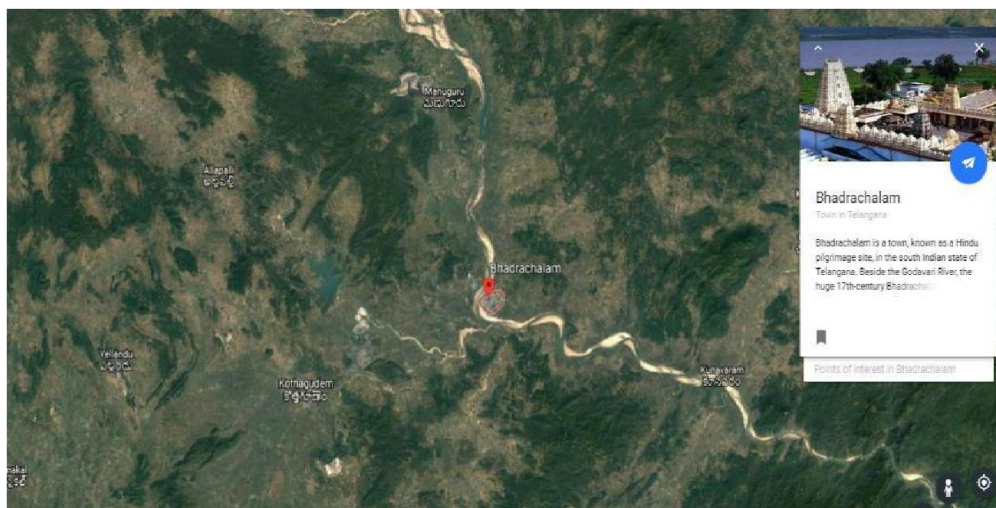


Fig : 1.1. Godavari River In Bhadrachalam

3. OVERVIEW

The extreme-flood analysis of upstream areas induced by a downstream dam, especially in a large river has been an interesting topic for many years. The standing-water in downstream of a dam can influence the upstream of the river, by causing the water surface elevation (WSE) to back up towards the upstream. This phenomenon can increase the depth of the river gradually upstream to form a smooth transition between a quasi-normal flow and standing water, forcing an alteration of the hydraulic conditions. The upstream of the river affected by this response would exceed several hundreds of kilometers in low slope rivers, which is termed as the backwater zone [1]. Dams act as a block on rivers and by forming backwater conditions, which in turn affect the water surface profile at upstream of the river [2–6]. There have been numerous studies on dams that these hydraulic structures can limit discharge and elevate water level at the upstream by creating backwater [7,8]. It is a matter of the fact that the backwater effect can induce upstream flooding, depending on the river, geometry, and on the flow and floodplain characteristics [9]. The backwater from downstream to upstream has caused most of the flood disasters. The consequences of flooding by backwater are mainly caused by the faulty design and operations of a large dam. The backwater phenomenon leads to an increase in the water surface level of upstream regions, thereby imposing the threat of submergence during flood events and affecting the longitudinal extent of the river reach.

The Godavari basin lies in the Deccan Plateau and is situated between 16°16' N and 22°43' N Latitude and 73°26' E and 83°07' E Longitude. It is roughly triangular in shape and the main river itself runs practically along the base of the triangle. The river Godavari, the largest of the peninsular rivers, and the third largest in India, drains about 10% of India's total geographical area. The catchment area of the river is 3,12,812 sq. km and is spread in the states of Maharashtra (48.6%), Telangana (18.5%), Andhra Pradesh (4.9%), Madhya Pradesh (10%), Chhattisgarh (10.9%), Odisha (5.7%) and Karnataka (1.4%). The river Godavari rises at an elevation of 1,067 m in the Western Ghats near Triambak hills in the Nasik district of Maharashtra. After flowing for about 1,465 km generally in south-east direction, through Maharashtra, Telangana and Andhra Pradesh, it falls into Bay of Bengal. A line diagram of Godavari River is attached as Annexure-1. The Godavari receives the waters of the Dharna on its right bank at downstream of Nasik town. Further down, river Kadwa joins near NMD Weir from the left. The combined waters of the Pravara and the Mula that rise in the hills of Aravali join the river from its right bank upstream of Jaikwadi Dam. In the Maharashtra state itself the river receives the combined waters from the Purna and Dudhna rivers. After crossing the border of Maharashtra the waters of the river Manjira joins it from right bank in Telangana. At this point, the Godavari flows at an elevation of about 329

m. The river Pranhita, conveying the combined waters of Penganga, Wardha and Wainganga, which drain the southern slopes of the Satpura range, falls into Godavari at Kaleswaram about 306 km downstream of its confluence with the Manjira. About 48 km downstream from Kaleswaram the waters of the river Indrāvati join Godavari. Both the Pranhita and the Indrāvati are major rivers in their own right. The last major tributary is Sabari from Odisha that falls into Godavari about 100 km upstream of Rajahmundry. The largest tributary of Godavari is the Pranhita with about 34.9% coverage of drainage area. The Pravara, Manjira and Maner are notable right bank tributaries covering about 16.1%, the Purna, Pranhita, Indrāvati and Sabari are important left bank tributaries, covering nearly 59.7% of the total catchment area of the basin. The 2 | P a g e Godavari in the Upper, Middle and Lower reaches make up for the balance of 24.2%.

3.1. Data Description

The dataset contains:

- Discharge
- Flood Runoff
- Daily Runoff
- Weekly Runoff

3.2. Software Design

- Jupyter Notebook Environment
- Spyder Ide
- Machine learning algorithms.
- Python (pandas, numpy, matplotlib,seaborn,sklearn)
- HTML
- Flask

3.3. Flow Chart

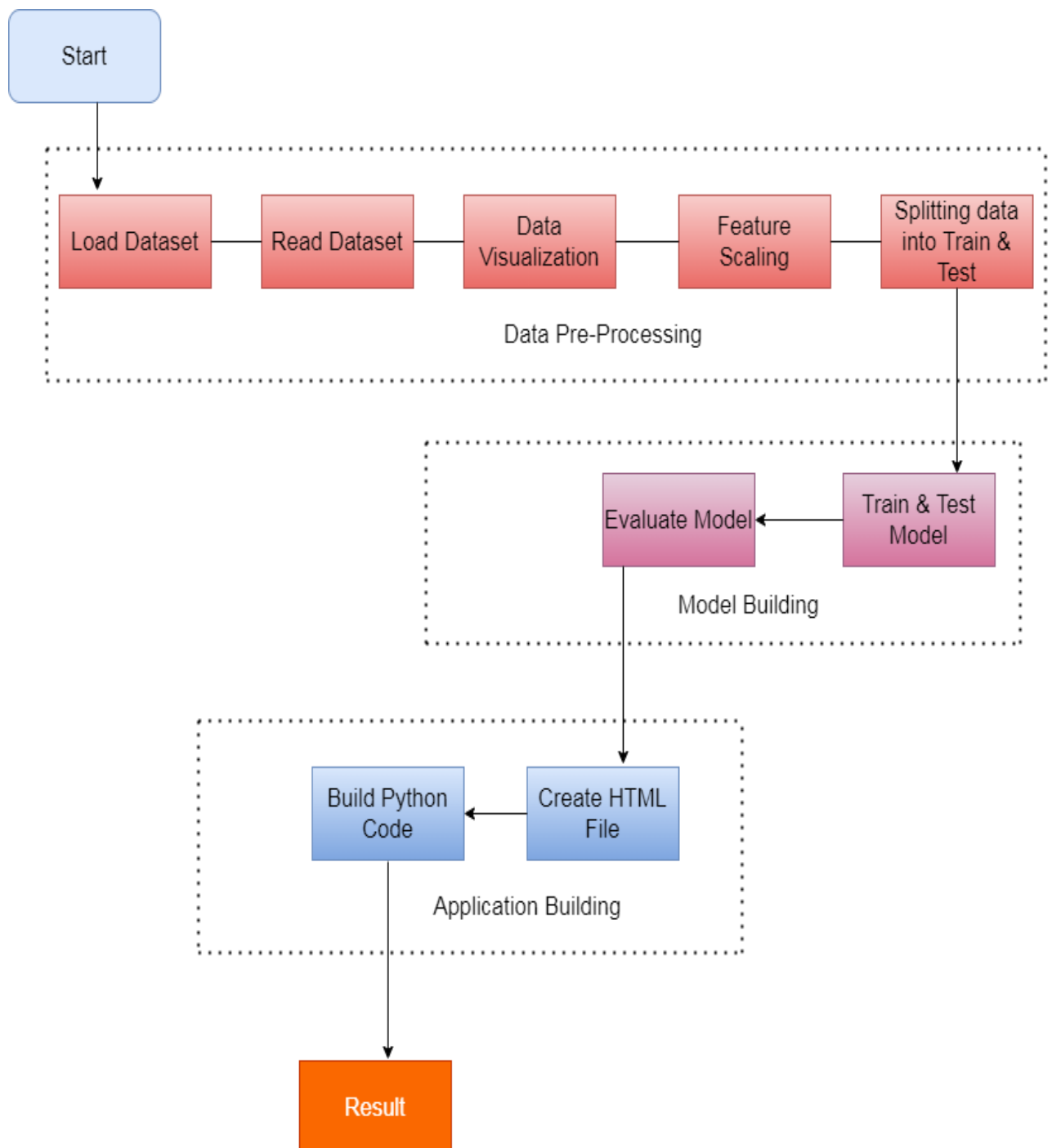


Fig 2.1.: Flow Chart

3.4. Importing Libraries and Dataset:

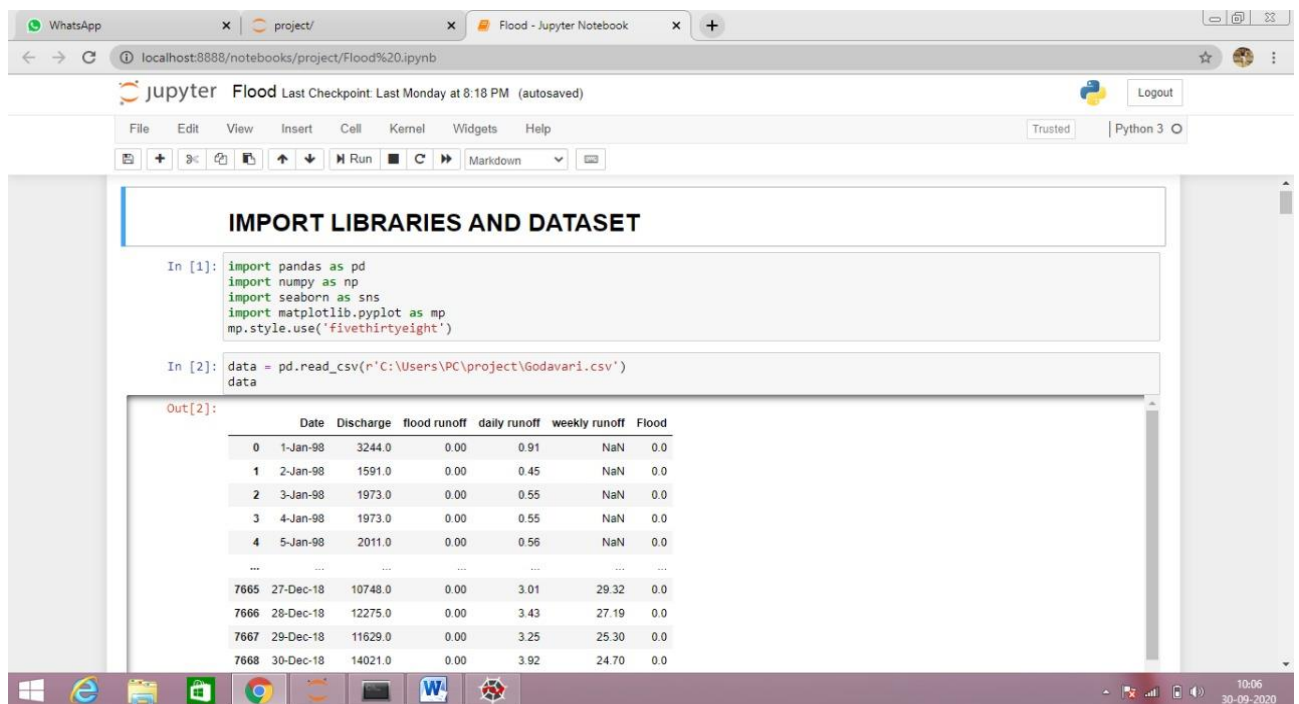


Fig 3.1. : Screenshot of Libraries and Dataset

3.5. Data visualization by Seaborn

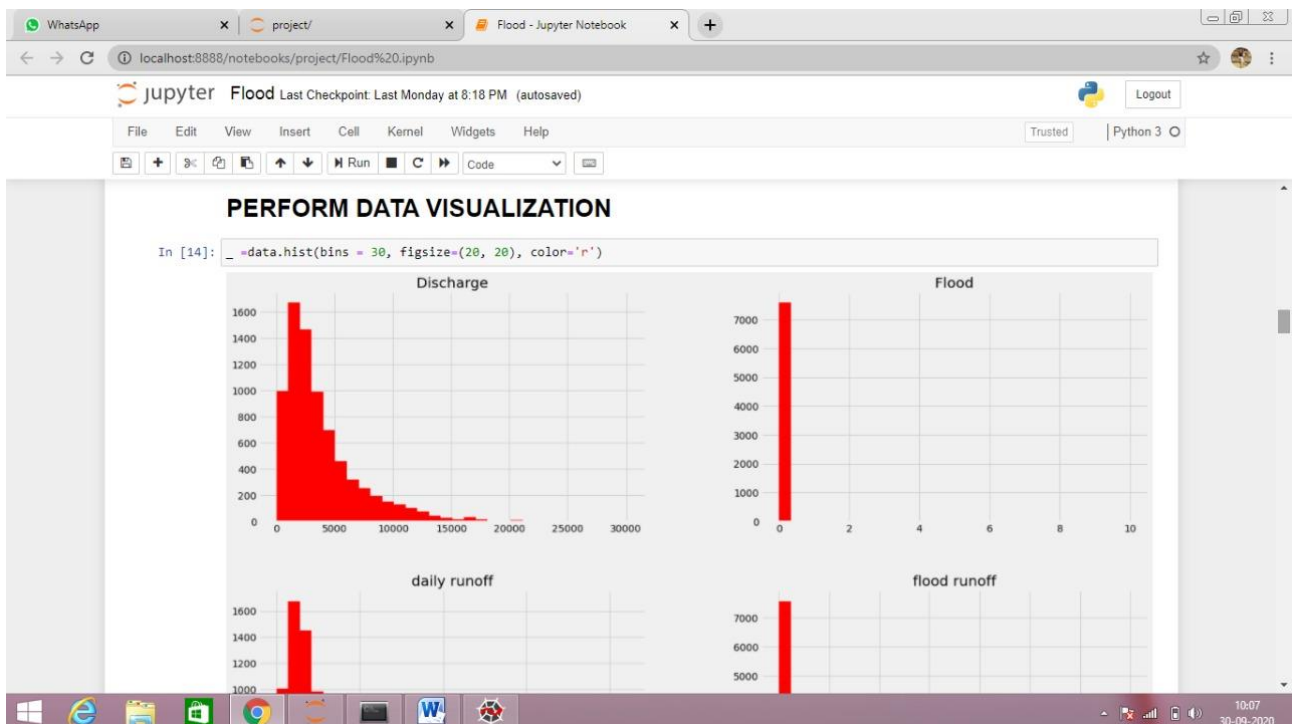


Fig 3.2.1 : Screenshot of Data Visualisation by Seaborn

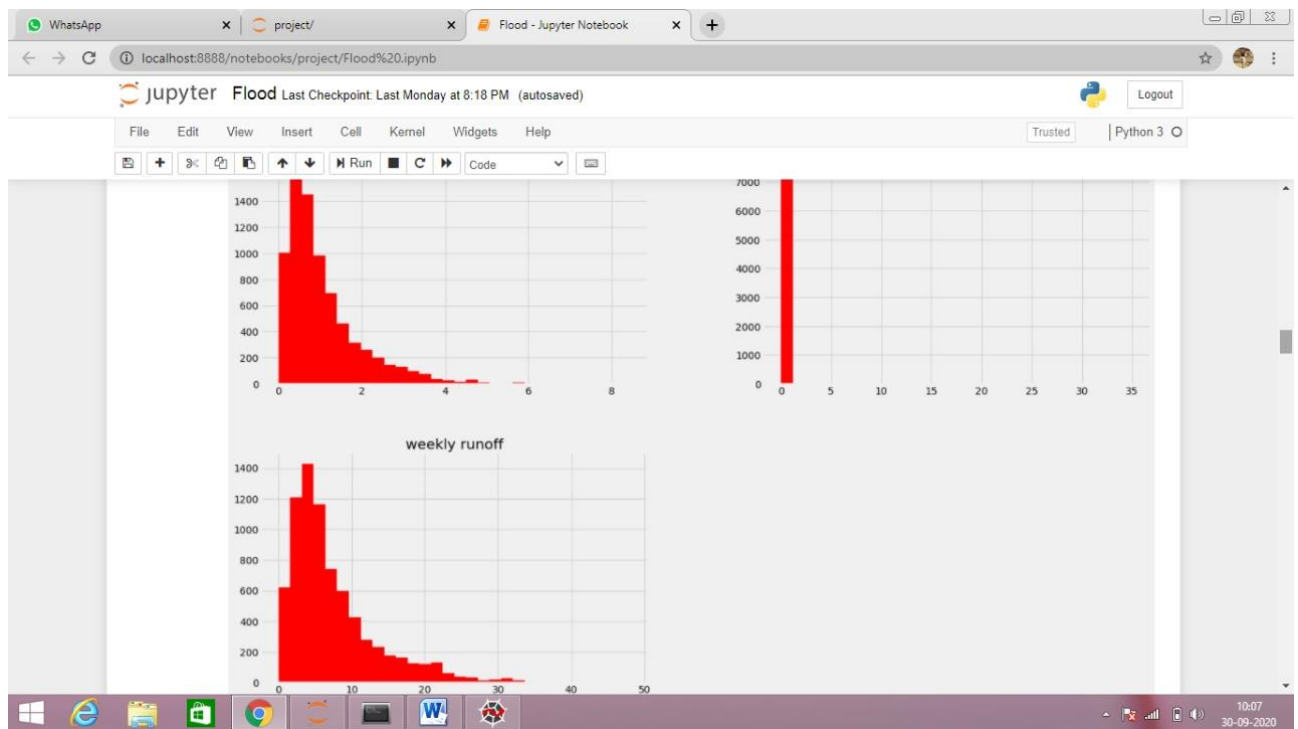


Fig 3.2.2. : Screenshot of Data Visualisation by Seaborn

3.6. Data visualization by Pairplot

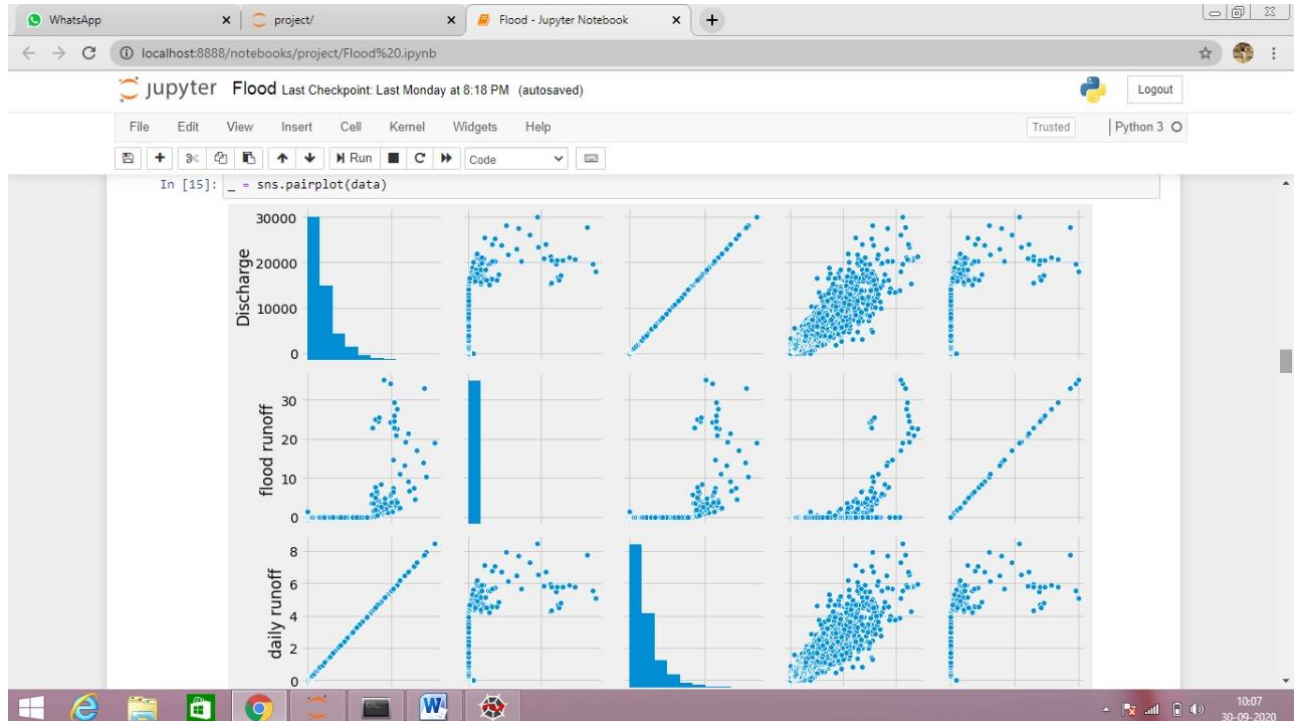


Fig 3.3.1. : Screenshot of Data Visualisation by Pairplot

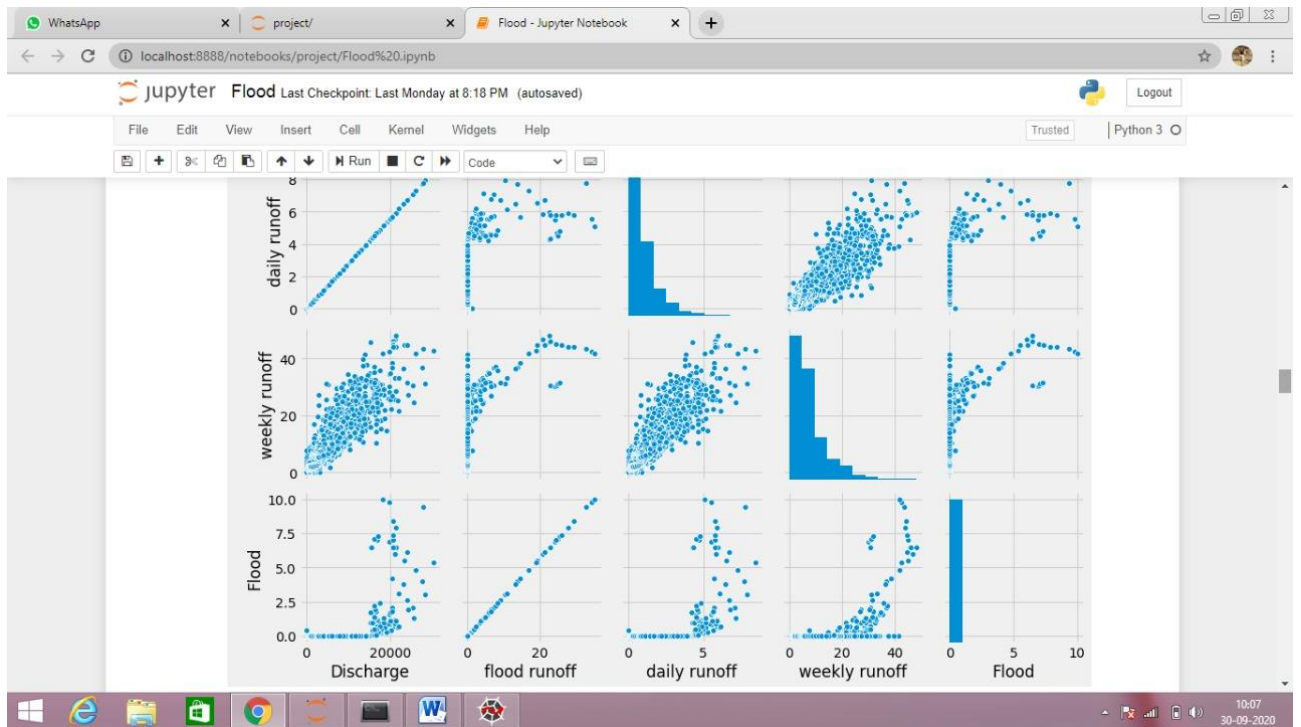


Fig 3.3.2. : Screenshot of Data Visualisation by Pairplot

3.7. Heatmap

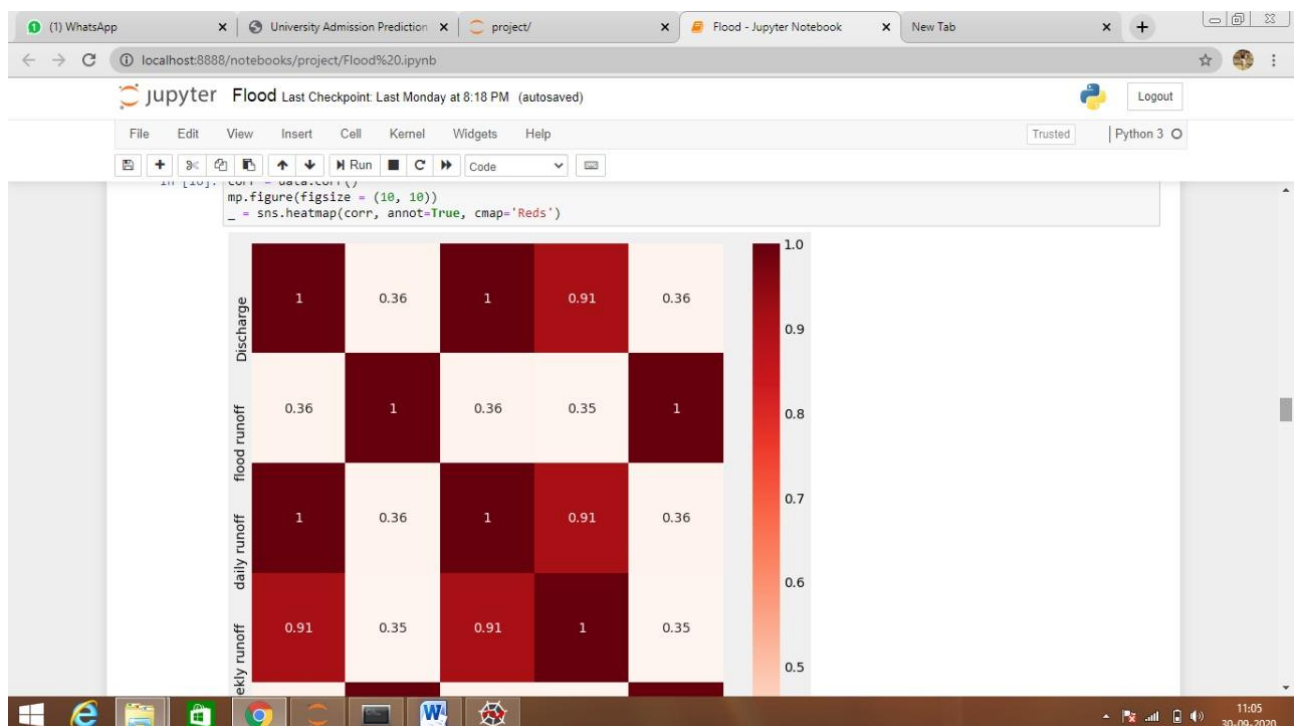


Fig 3.4. : Screenshot of Heatmap

3.8. Outliers

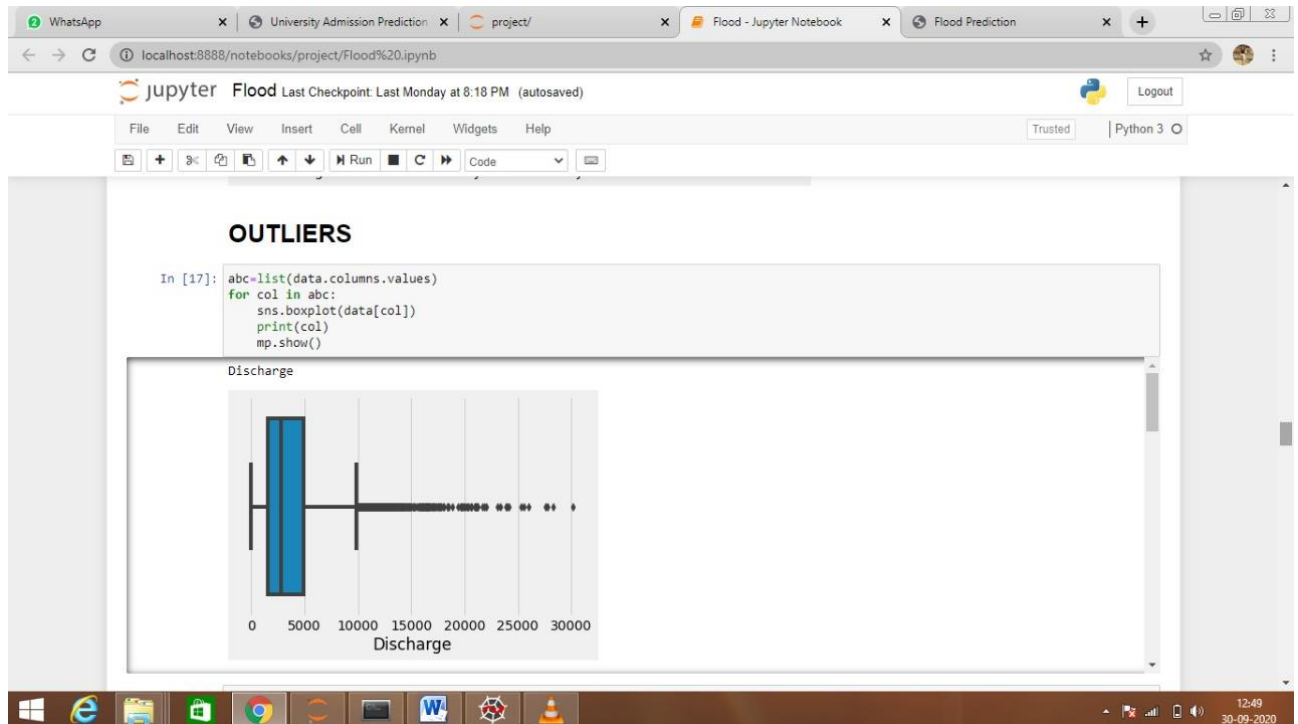


Fig 3.5.1. : Screenshot of Outliers of Discharge

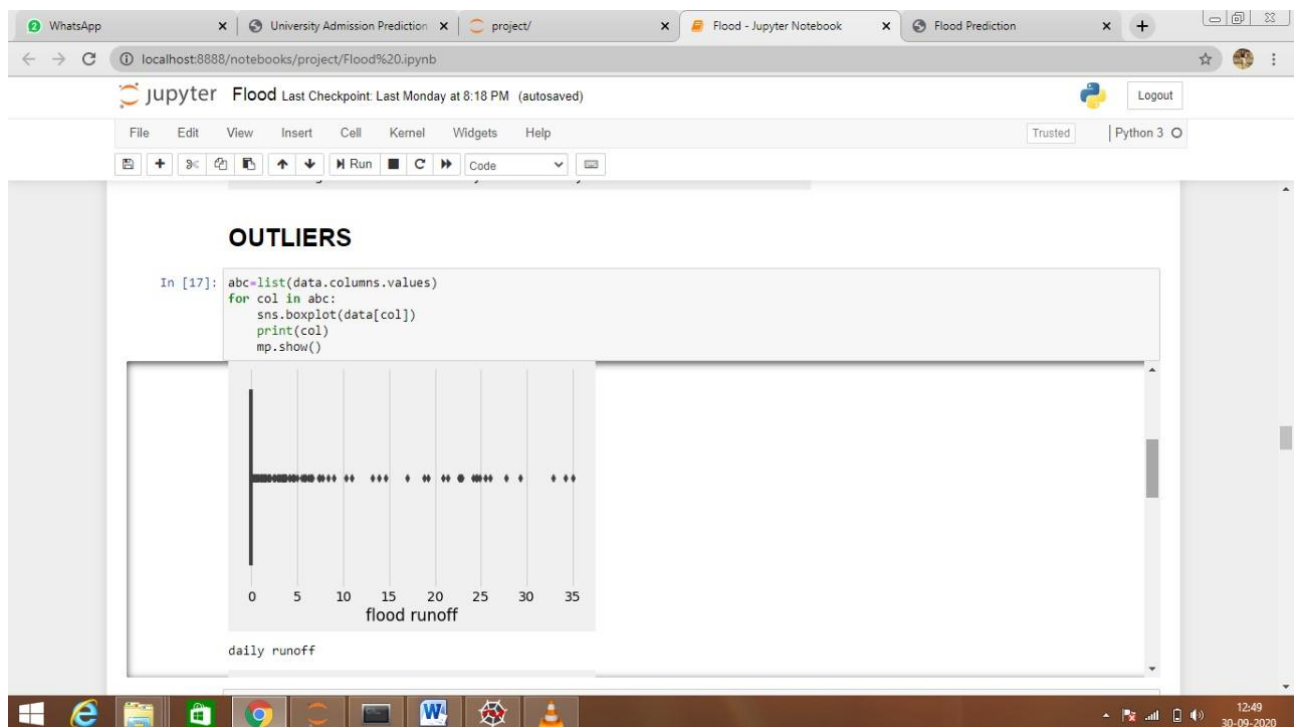


Fig 3.5.2. : Screenshot of Outliers of Flood Runoff

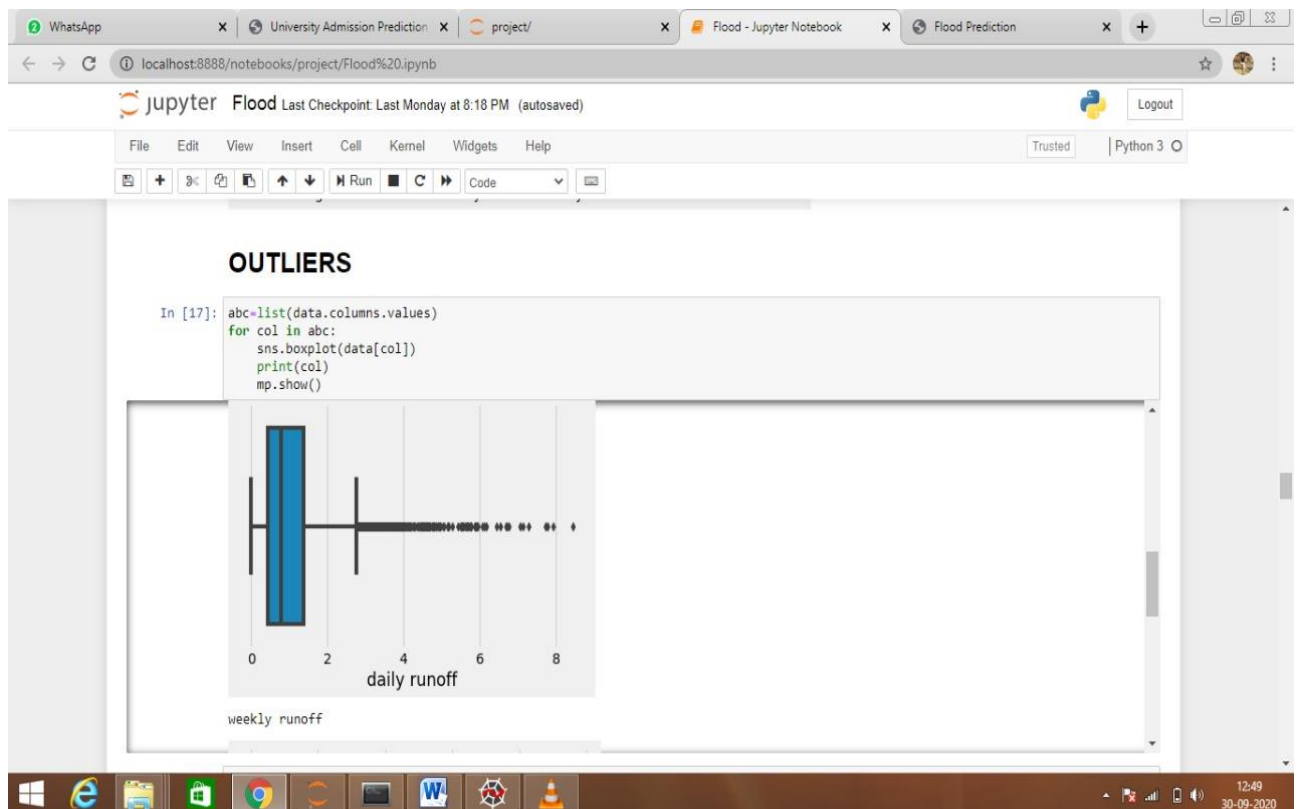


Fig 3.5.3. : Screenshot of Outliers of Daily Runoff

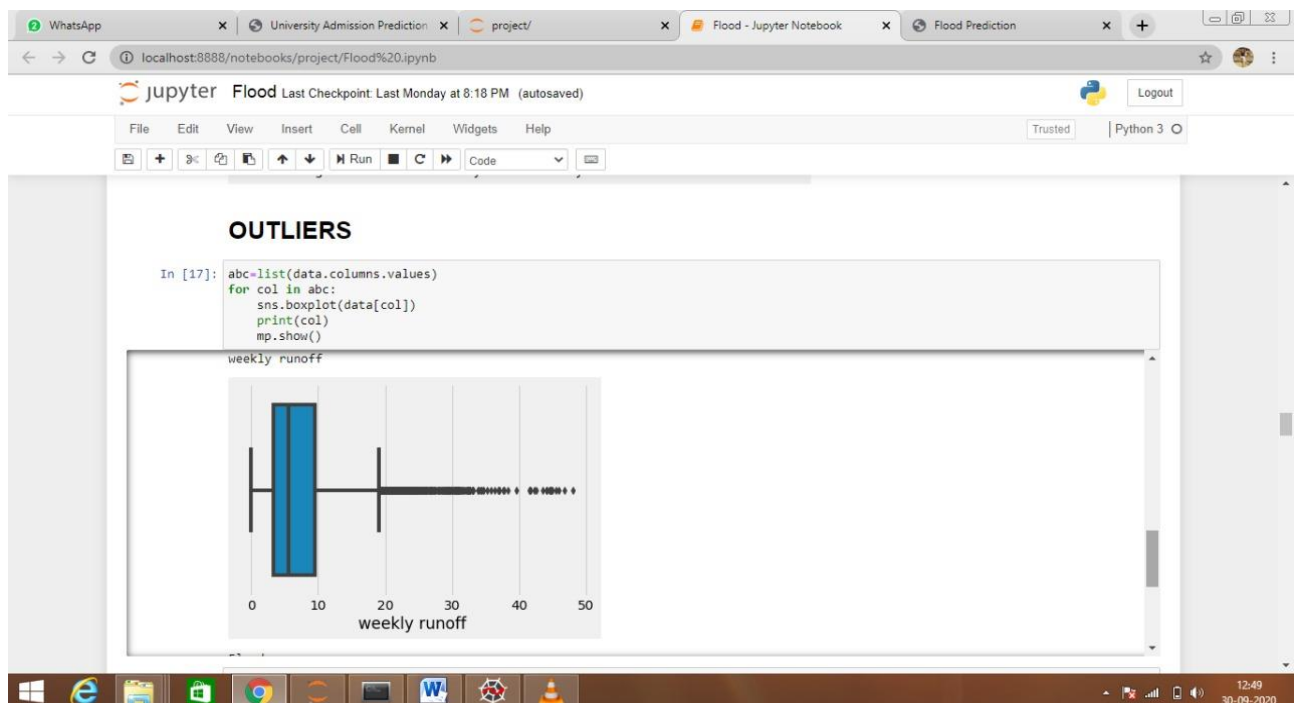


Fig 3.5.4. : Screenshot of Outliers of Weekly Runoff

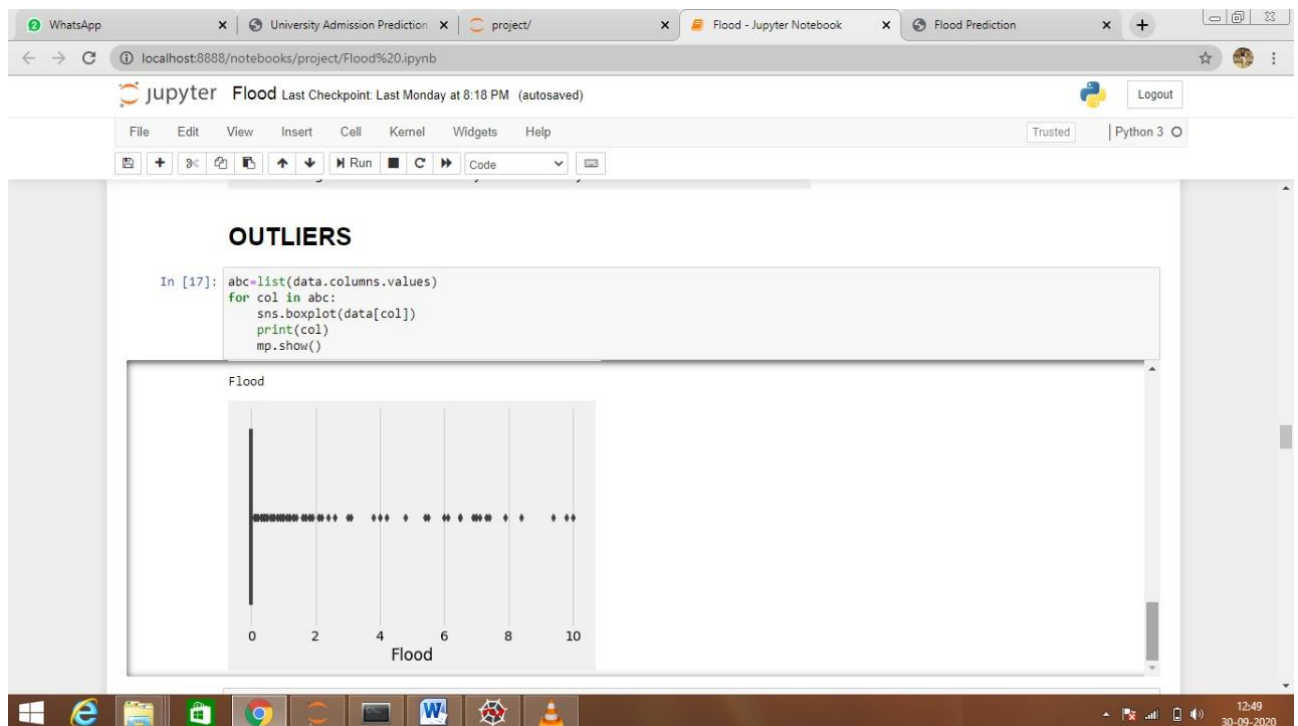


Fig 3.5.5. : Screenshot of Outliers of Food

3.9. Create Training and Testing Dataset

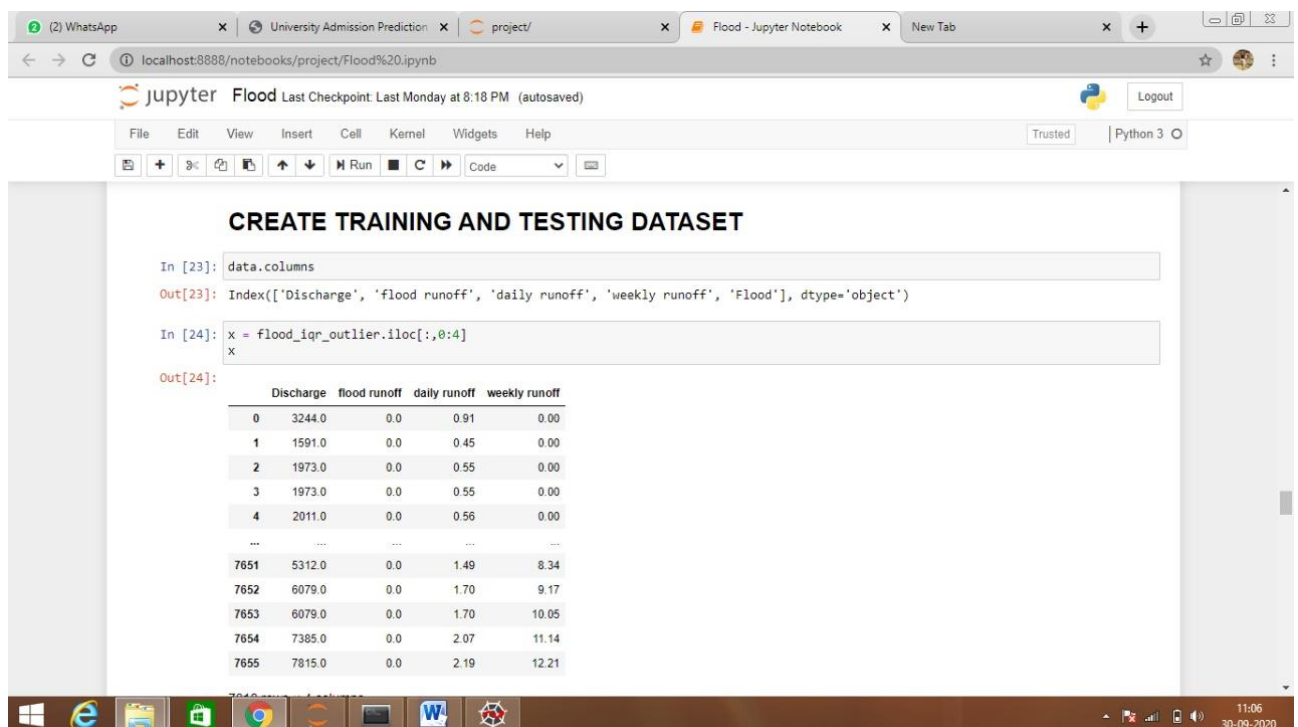
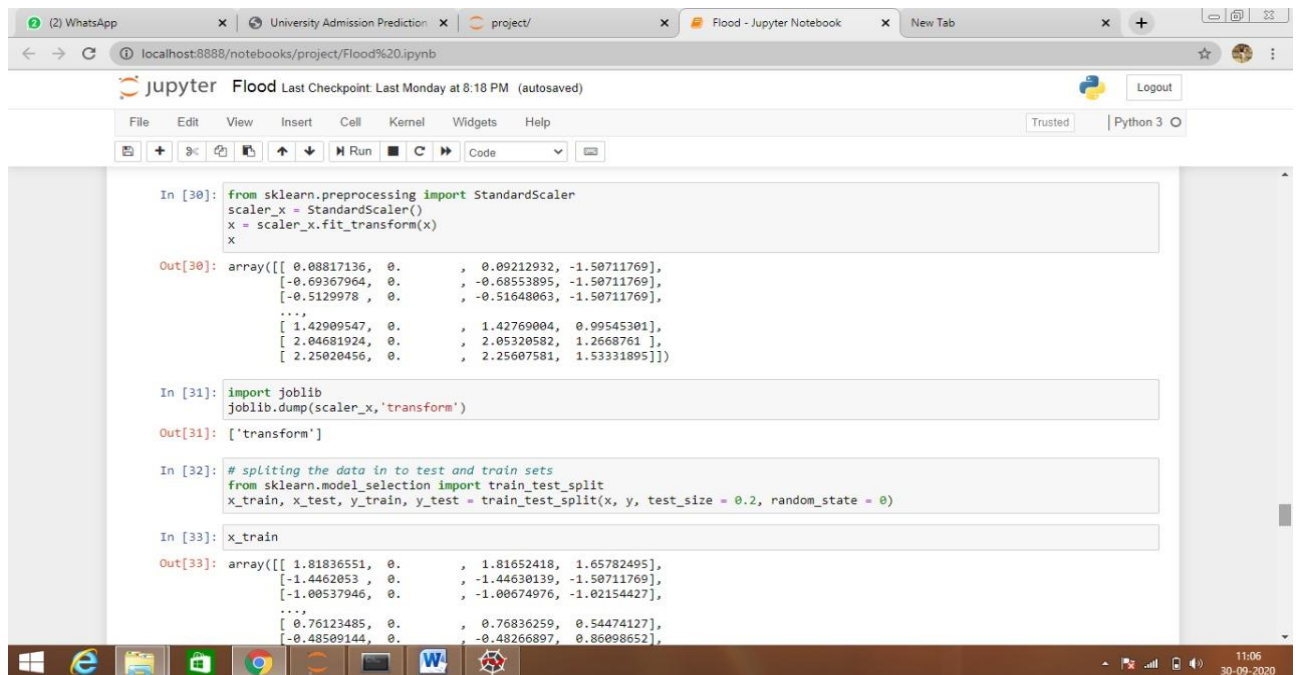


Fig 3.6. : Screenshot of Training and Testing Dataset

3.10. Feature Scaling and Model Building



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [30]: from sklearn.preprocessing import StandardScaler
scaler_x = StandardScaler()
x = scaler_x.fit_transform(x)
x
```

```
Out[30]: array([[ 0.08817136,  0.          ,  0.09212932, -1.50711769],
 [-0.69367964,  0.          , -0.68553895, -1.50711769],
 [-0.5129978 ,  0.          , -0.51648063, -1.50711769],
 ...,
 [ 1.42909547,  0.          ,  1.42769004,  0.99545301],
 [ 2.04681924,  0.          ,  2.05320582,  1.2668761 ],
 [ 2.25020456,  0.          ,  2.25607581,  1.53331895]])
```

```
In [31]: import joblib
joblib.dump(scaler_x, 'transform')
```

```
Out[31]: ['transform']
```

```
In [32]: # splitting the data in to test and train sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

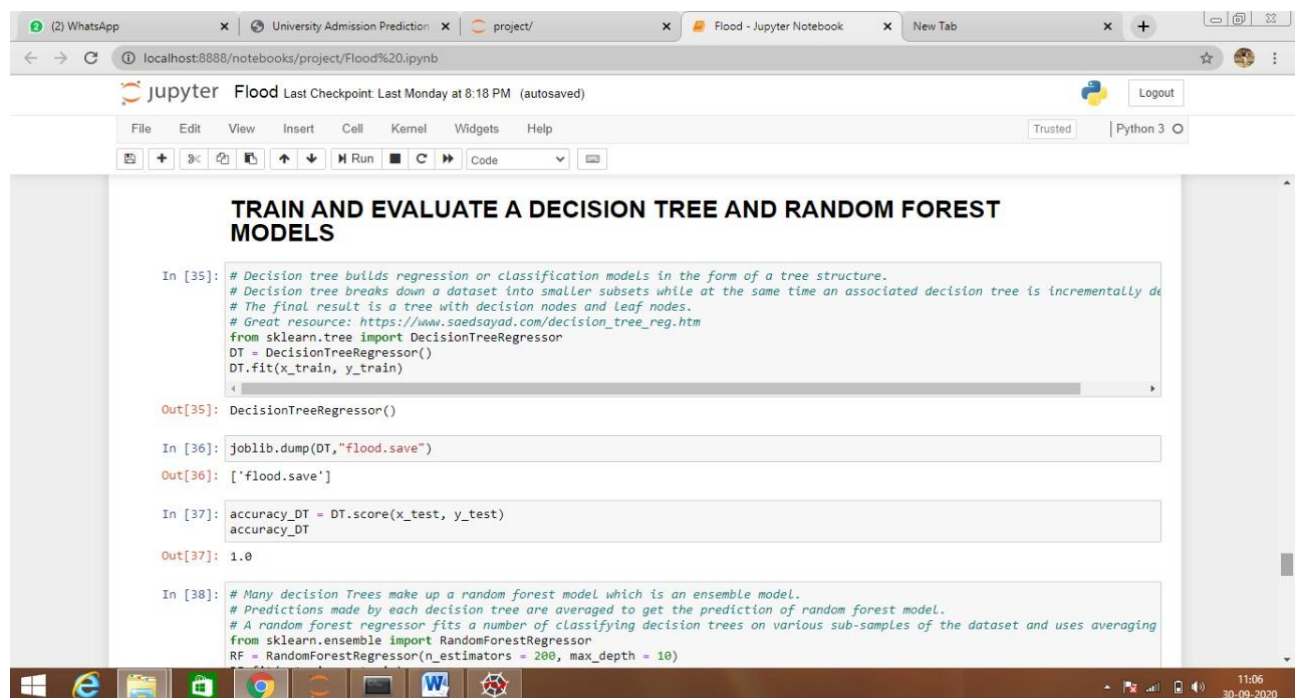
```
In [33]: x_train
```

```
Out[33]: array([[ 1.81836551,  0.          ,  1.81652418,  1.65782495],
 [-1.44620253,  0.          , -1.44630139, -1.50711769],
 [-1.00537946,  0.          , -1.00674976, -1.02154427],
 ...,
 [ 0.76123485,  0.          ,  0.76836259,  0.54474127],
 [-0.48509144,  0.          , -0.48266897,  0.86098652],
```

Fig 3.7. : Screenshot of Feature Scaling and Model Building

3.11. Train And Evaluate a Decision Tree And Random Forest Model

3.11.1. Decision Tree Model



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
TRAIN AND EVALUATE A DECISION TREE AND RANDOM FOREST MODELS
```

```
In [35]: # Decision tree builds regression or classification models in the form of a tree structure.
# Decision tree breaks down a dataset into smaller subsets while at the same time an associated decision tree is incrementally de
# The final result is a tree with decision nodes and leaf nodes.
# Great resource: https://www.saedsayad.com/decision_tree_reg.htm
from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor()
DT.fit(x_train, y_train)
```

```
Out[35]: DecisionTreeRegressor()
```

```
In [36]: joblib.dump(DT, "flood.save")
```

```
Out[36]: ['flood.save']
```

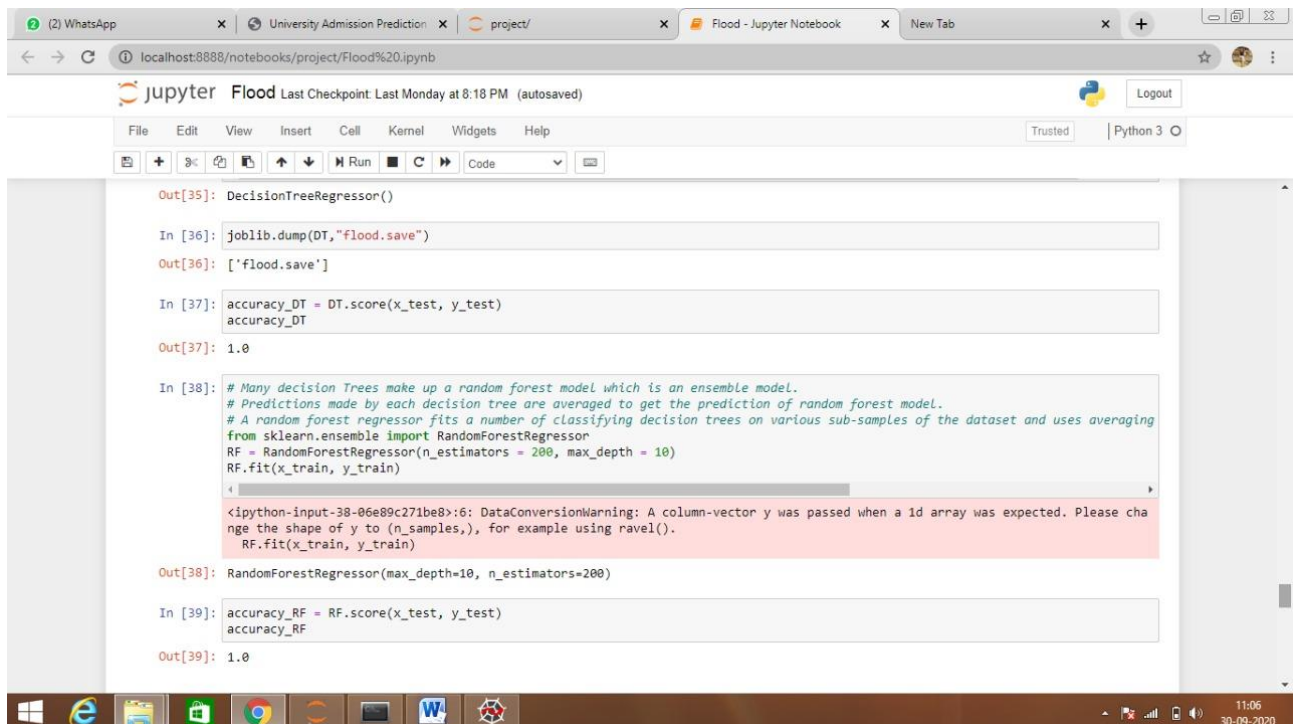
```
In [37]: accuracy_DT = DT.score(x_test, y_test)
accuracy_DT
```

```
Out[37]: 1.0
```

```
In [38]: # Many decision Trees make up a random forest model which is an ensemble model.
# Predictions made by each decision tree are averaged to get the prediction of random forest model.
# A random forest regressor fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(n_estimators = 200, max_depth = 10)
```

Fig 3.8. : Screenshot of Evaluate a Decision Tree

3.11.2. Random Forest Model



```
Out[35]: DecisionTreeRegressor()

In [36]: joblib.dump(DT, "flood.save")

Out[36]: ['flood.save']

In [37]: accuracy_DT = DT.score(x_test, y_test)
accuracy_DT

Out[37]: 1.0

In [38]: # Many decision Trees make up a random forest model which is an ensemble model.
# Predictions made by each decision tree are averaged to get the prediction of random forest model.
# A random forest regressor fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(n_estimators = 200, max_depth = 10)
RF.fit(x_train, y_train)

<ipython-input-38-06e89c271be8>:6: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
RF.fit(x_train, y_train)

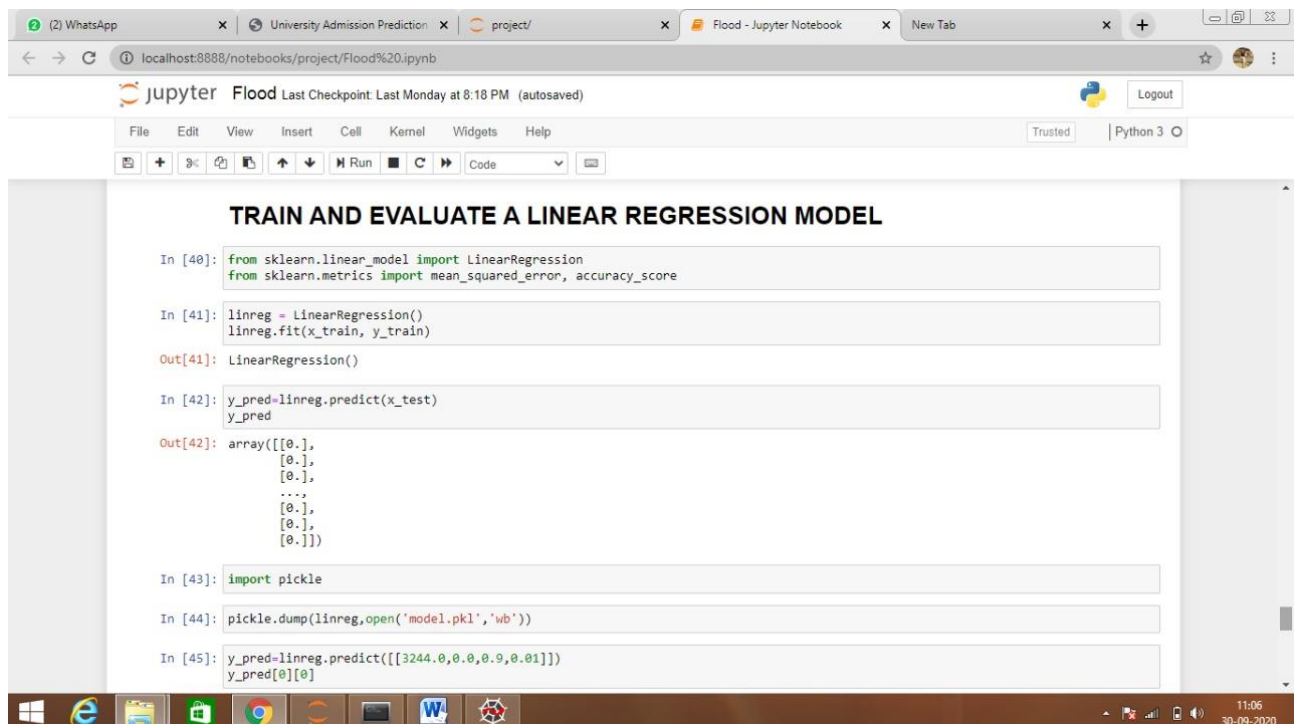
Out[38]: RandomForestRegressor(max_depth=10, n_estimators=200)

In [39]: accuracy_RF = RF.score(x_test, y_test)
accuracy_RF

Out[39]: 1.0
```

Fig 3.9. : Screenshot of Evaluate a Random Forest

3.12. Train And Evaluate a Linear Regression Model



```
TRAIN AND EVALUATE A LINEAR REGRESSION MODEL

In [40]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score

In [41]: linreg = LinearRegression()
linreg.fit(x_train, y_train)

Out[41]: LinearRegression()

In [42]: y_pred=linreg.predict(x_test)
y_pred

Out[42]: array([[0.],
               [0.],
               [0.],
               ...,
               [0.],
               [0.],
               [0.]])

In [43]: import pickle

In [44]: pickle.dump(linreg, open('model.pkl', 'wb'))

In [45]: y_pred=linreg.predict([[3244.0,0.0,0.9,0.01]])
y_pred[0][0]
```

Fig 3.10. : Screenshot of Train And Evaluate a Linear Regression Model

3.13. HTML File :

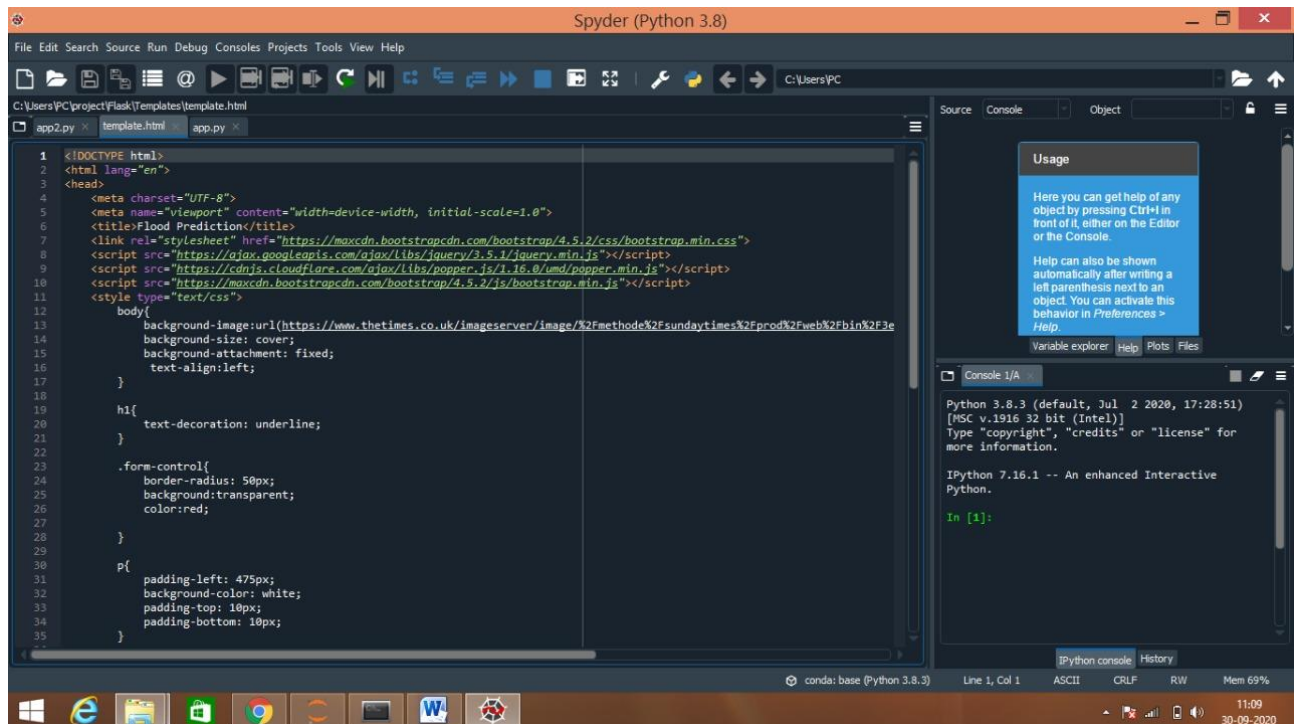


Fig 3.11.1. : Screenshot of HTML File

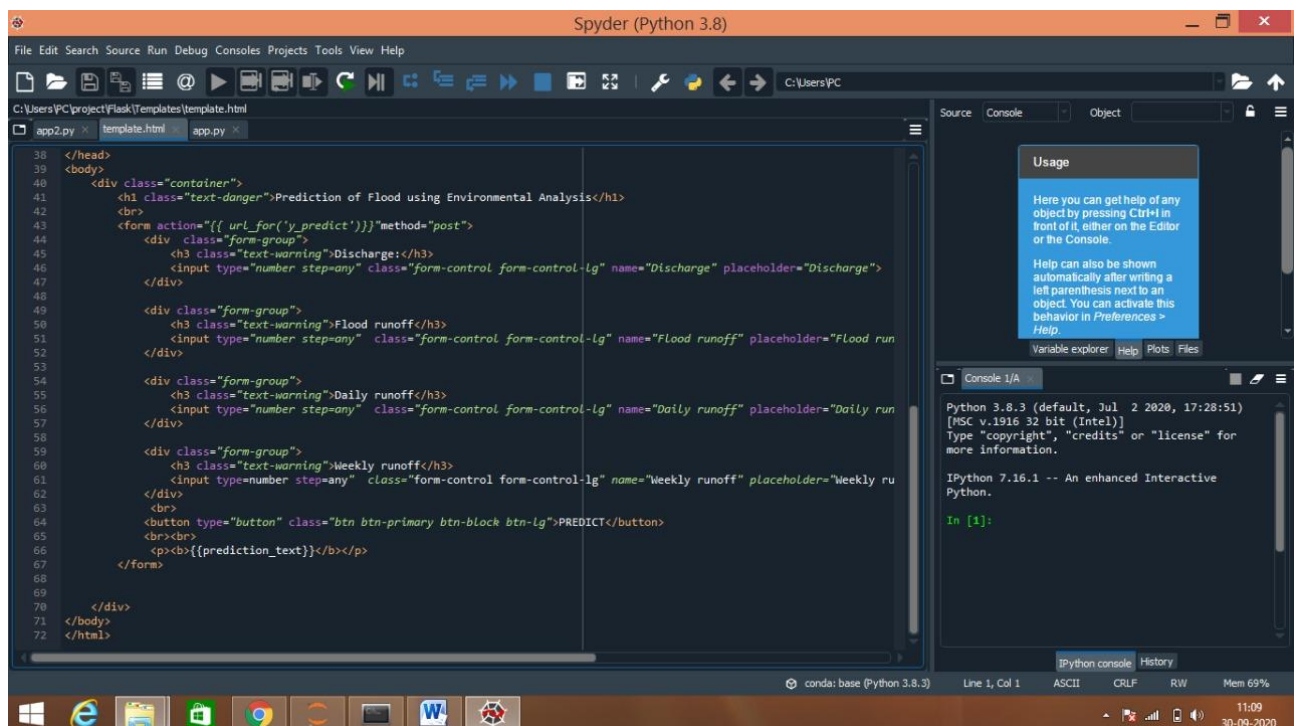


Fig 3.11.2. : Screenshot of HTML File

3.14. Python file

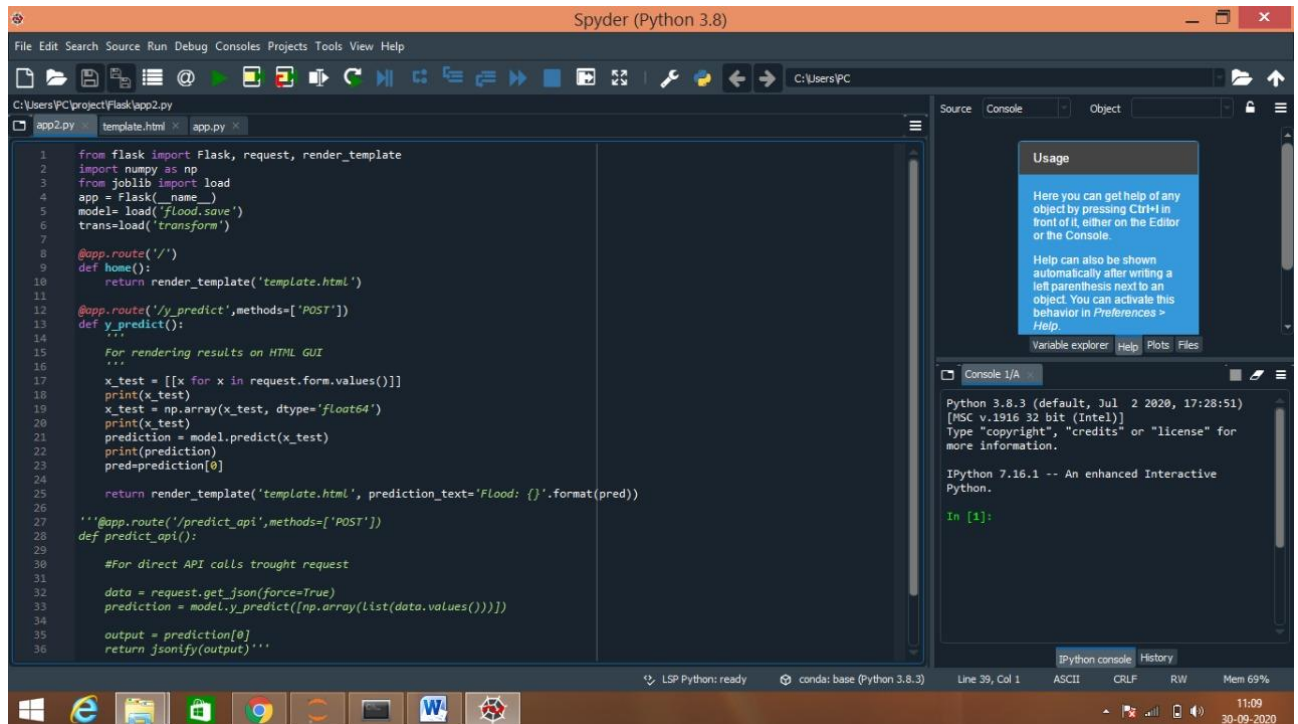


Fig 3.12. : Screenshot of Python code (app.py file)

4. OUTPUT:

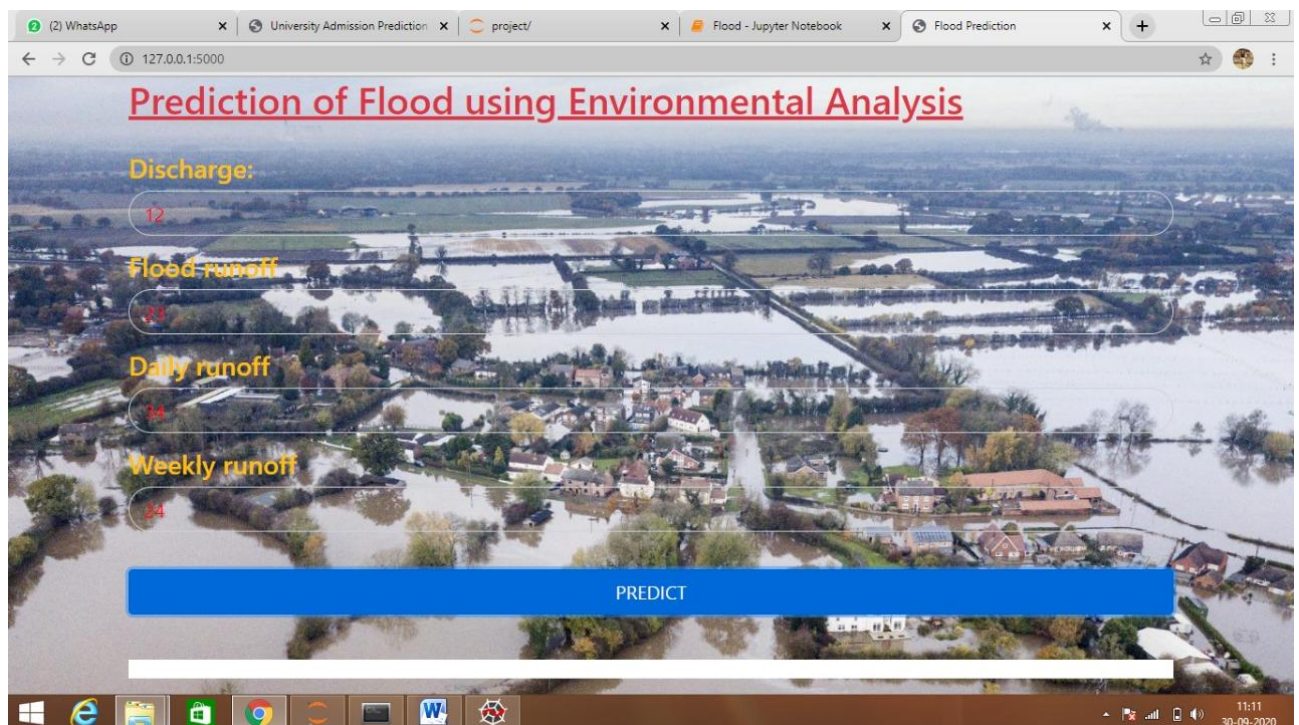


Fig 3.13. : Screenshot of Output screen

5. ADVANTAGES AND DISADVANTAGES OF PROPOSED SYSTEM

5.1. Advantages

The advantages is the meteorological ensembles and the hydrological multi model to generate a probabilistic forecast offers an improved skill score compared to the deterministic forecast, for all lead times. The approach allows the different sources of uncertainty to be assessed separately in an operational setting. This knowledge can be used to give decision makers information on the total uncertainty and on where that uncertainty is coming from, offering richer information for decisions. The approach does not rely on a specific gauged location and results with uncertainty estimation can be extracted anywhere in the modelled domain and can be applied to any forecasting system where probabilistic forecasts are required.

5.2. Disadvantages

The disadvantages of the ensemble method are that creating an ensemble depends on the definition of the feasible model structure and parameter space. However, defining this space in itself is uncertain, which can lead to the ensemble being under or over dispersive and thus not representing the predictive uncertainty accurately. For some forecasting systems the total ensemble size of 50,000 members will be a significant drawback for use in practice, owing to the consequent required increase in computation resources, data management and fast running of models

6. CONCLUSION

The simulations show that the computed hydrographs match well with the observed hydrographs. These hydrographs match very well when the discharge in the river is less than 30,000 m³ s⁻¹. Error will increase slightly when discharge is beyond this range, due to floodplain inundation. Floodplain inundation causes slightly higher model computations as compared to the observed values. With this hydrological modelling approach, accuracy in discharge computations is improved compared to conventional methods, flood forecast at any river confluence can be issued, and influence of any tributary can be examined separately. In the real-time application the flood forecast model improved the lead time by 12 hours compared to conventional methods of forecasting. The calibrated model is found quite stable in real-time operations. Model accuracy can be further improved by computing floodplain inundations..

7. BIBLIOGRAPHY

1. Azhar Hussain, "Flood Modelling by Using HEC-RAS", Department of Civil Engineering, Jamia Millia Islamia, (central university), New Delhi, August 2017.
2. K. Somya, C. M. Jain, and N. K. Shrivastava, "Urban flood vulnerability zoning of Cochin City, southwest coast of India, using remote sensing and GIS" *Nat Hazards* (2015) 75:1271–1286.
3. Eldho T. I., and Anand T. Kulkarni "Flood Management in Coastal Cities using GIS, Remote Sensing and Numerical Models" *IJSER*, Volume 5, Issue 7, July-2014 ISSN 2229-5518