

FLOOD PREDICTION

Name : Y. Pavan Ganesh Reddy

Reg No : 39110758

College : Sathyabama Institute of Science and Technology

Branch : B.E , CSE.

Project Title : Flood prediction using Machine Learning.

INDEX

1. INTRODUCTION

- a. Overview
- b. Purpose

2. LITERATURE SURVEY

- a. Existing Solution
- b. Proposed solution

3. THEORETICAL ANALYSIS

4. HARDWARE / SOFTWARE REQUIREMENTS

5. FLOW CHART

6. RESULT

7. ADVANTAGES AND DISADVANTAGES

8. APPLICATIONS

9. CONCLUSION

10. FEATURE SCOPE

11. BIBILOGRAPHY

12. APPENDIX

- a. Source code
- b. UI output Screen-shots

INTRODUCTION

Floods are inevitable, but with timely alerts, their effects can be minimized. There are many people who die every year due to devastating floods, the number of people become homeless and many people die due to lack of proper help after a flood. The lack of timely alerts has always been an issue concerning it.

Delay in alerts in flood-prone areas is the biggest loophole of an economy. Conventional systems run a little low in forecasting floods at the right time so that proper actions could be taken before any disaster.

By using machine learning we can predict floods or forecast floods with better accuracy. This project aims at building predictive modeling based on the historical weather data of particular areas in order to predict the occurrence of floods.

The predictive model is built on different machine learning algorithms. The concerned authority monitor this flood prediction system through a web application.

LITERATURE SURVEY

For this project, we need to use classification algorithms as the output we will get is categorical values which is either floods will occur or not so, we need to build the most accurate classification algorithm to get the desired output

In this project, we will be using classification algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in save format. We will be doing flask integration and cloud deployment.

THEORETICAL ANALYSIS

1. Install Required Libraries.
 - Collect the dataset or Create the dataset
2. Data collections
 - Import the Libraries.
3. DataPreprocessing
 - Importing the dataset.
 - Understanding Data Type and Summary of features.
 - Take care of missing data
 - Data Visualization.
 - Drop the column from DataFrame & replace the missing value.
 - Splitting the Dataset into Dependent and Independent variables
 - Splitting Data into Train and Test.
4. Model Building
 - Training and testing the model
 - Evaluation of Model
 - Saving the Model
5. Application Building
 - Create an HTML file
 - Build a Python Code
6. Final UI
 - Dashboard Of the flask app.

HARDWARE / SOFTWARE REQUIREMENTS

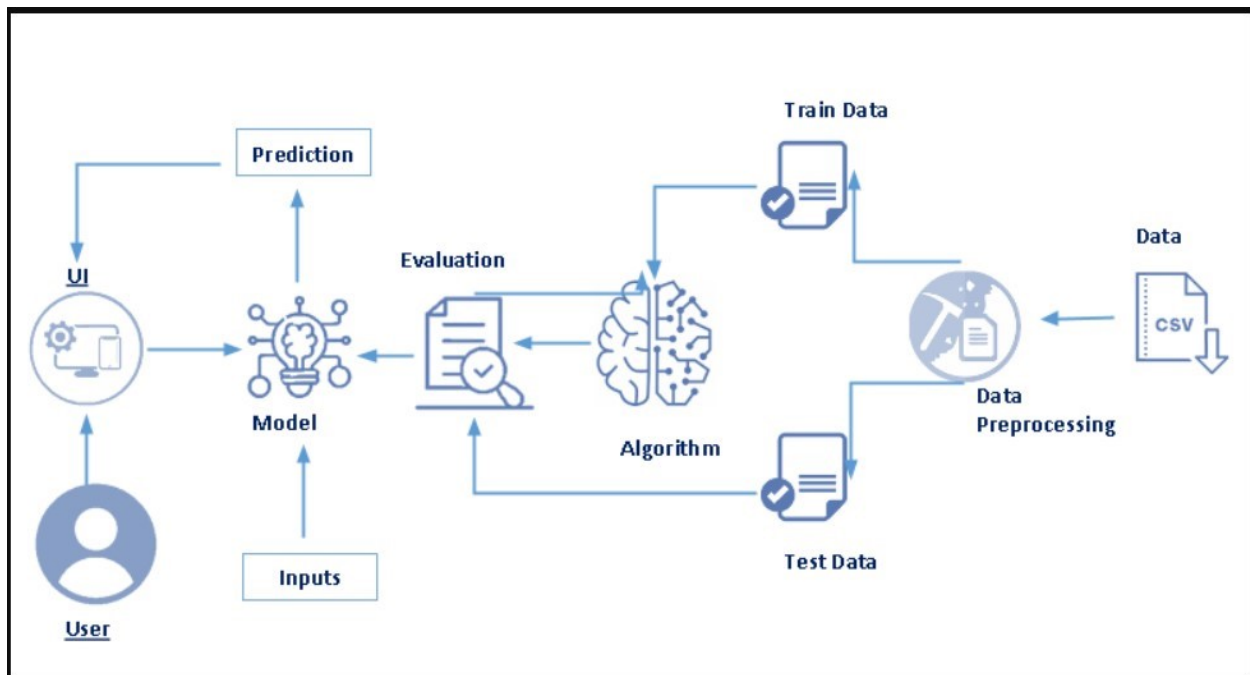
Category: Machine Learning

Skills Required:

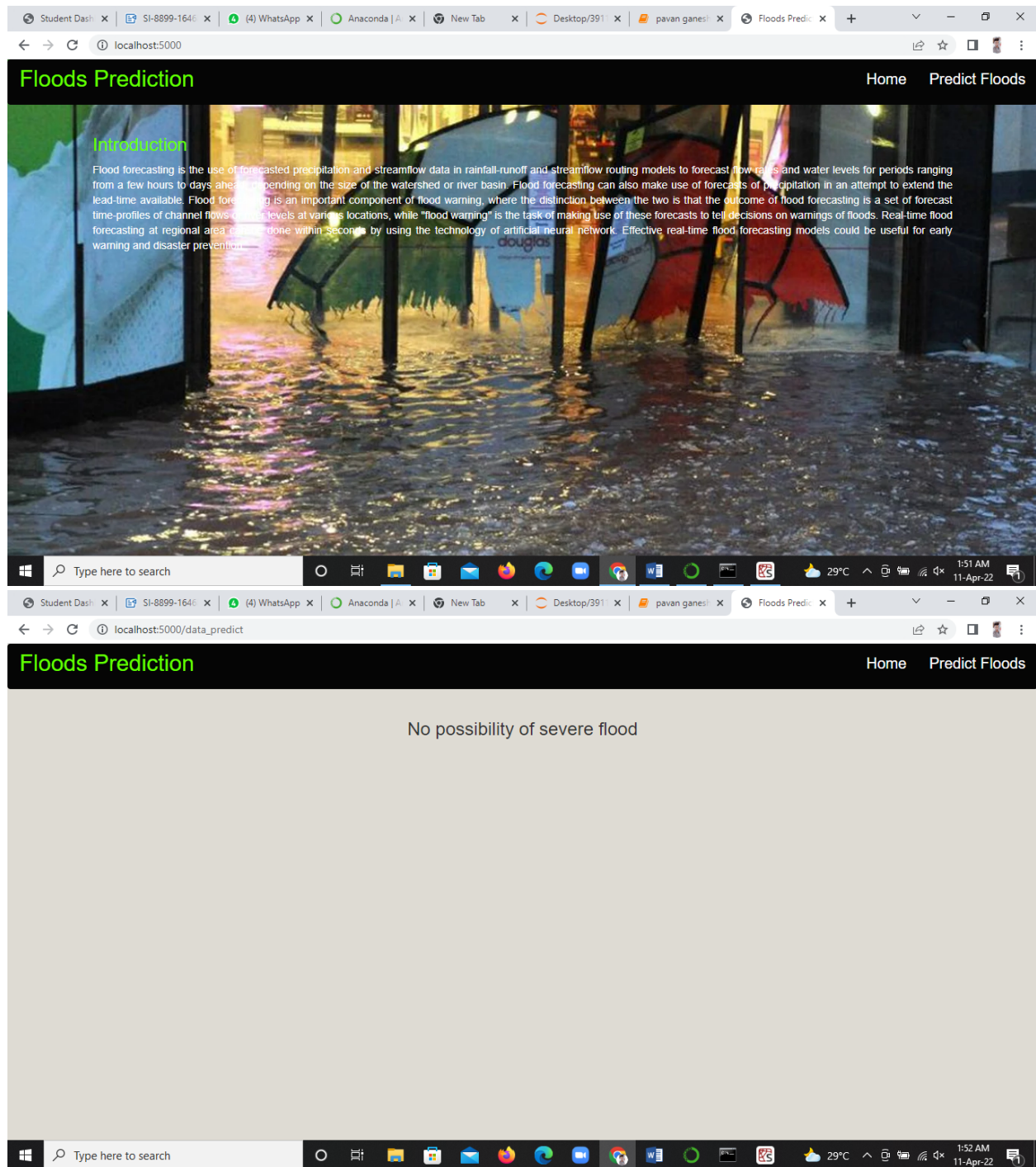
Python,Python Web Frame Works,Python For Data Analysis,Exploratory Data Analysis,Data Preprocessing Techniques,Machine Learning

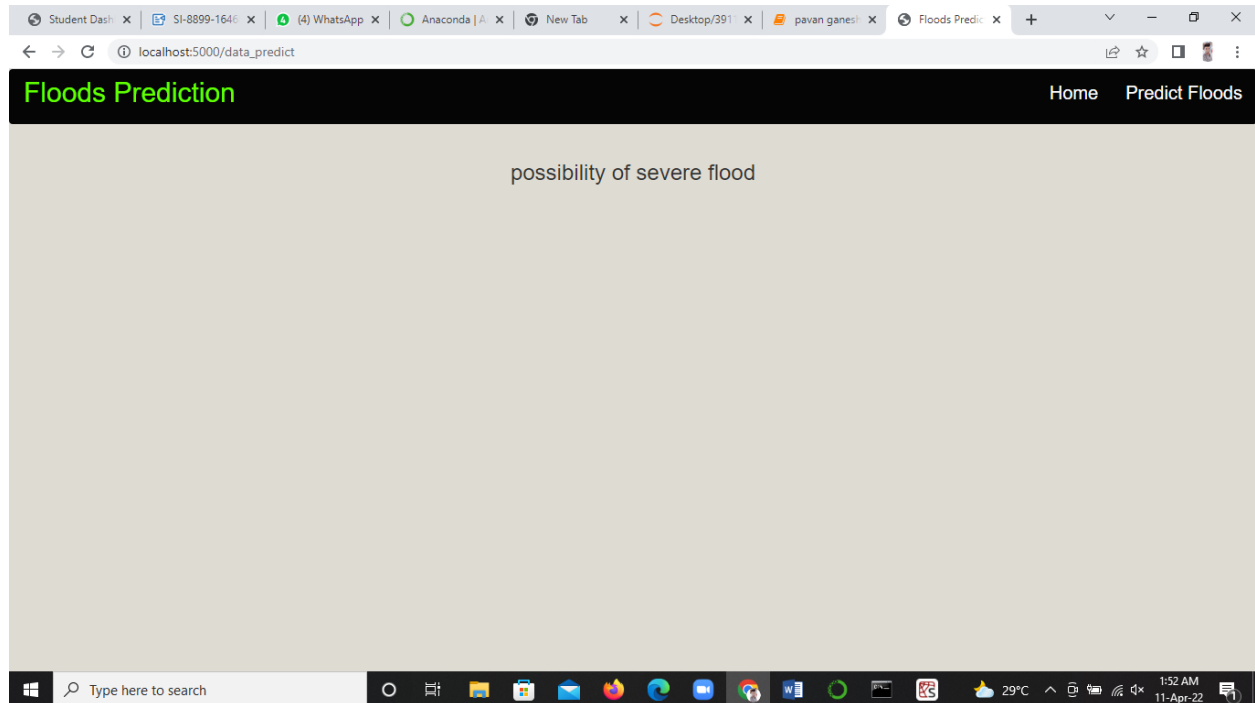
While working on the solution, I have seen the dataset of the floods and its quite frequent to see the whether forecast change drastically and cloud cover increases , rainfalls in the months April to June and lastly floods occur.There is interconnection of all the factors which leads to floods.

FLOW CHART



RESULT





ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

- >The main purpose of flood warning is to save life by allowing people, support and emergency services time to prepare for flooding.
- >The model which I built is 96 % accurate so one can rely on the prediction done by the model.
- >With the help of this model , you can determine and manage environmental and water resource systems.
- >Knowledge about the characteristics of a river's drainage basin, such as soil-moisture conditions, ground temperature, snowpack, topography, vegetation cover, and impermeable land area, which can help to predict how extensive and damaging a flood might become.

DISADVANTAGES :

- >One of the algorithm used in this model is Random Forest, where it can become a plot as it gets prediction accuracy on complex

problems is usually inferior to Gradient - Boosted Trees.

APPLICATIONS

The applications in flood prediction can be classified according to flood resource variables, i.e., water level, river flood, soil moisture, rainfall–discharge, precipitation, river inflow, peak flow, river flow, rainfall–runoff, flash flood, rainfall, streamflow, seasonal stream flow.

CONCLUSION

The summary of the project is that even though Floods are inevitable, they can be prevented with accurate measures like predicting the outcomes, changes to be taken to be on the safe side like building dams, reservoirs etc..and especially the technology has taken huge turn so, it will be matter of time before we gain control on what we are challenged.

FUTURE SCOPE

There can be future changes can be made to my model by adding the data like water level, soil moisture, average temperature of the area, rainfall - discharge , precipitation , drainage systems in the area etc.... in the dataset makes even more accurate and since the data will be enhanced, the analysis becomes the key part in the prediction.

BIBLIOGRAPHY

There are some references which are to be considered while making the project.

URL - 1 : [Flood prediction - Oxford Reference](#)

URL - 2 : [Flood Forecasting - an overview | ScienceDirect Topics](#)

APPENDIX - SOURCE CODE

operable program or batch file.

```
In [1]: #import required libraries
import numpy as np #for dealing high dimensional data
import pandas as pd #to do statistical data analysis
import matplotlib.pyplot as plt #for 2D visualization
import seaborn as sns #High end data visualization

In [2]: #read the dataset
dataset=pd.read_excel(r'C:\Users\Pavan\Desktop\Y PAVAN GANESH-FLOOD PREDICTION\DATASET\flood dataset7889.xlsx')
```

```
In [3]: #check the first 5 observations
dataset.head()
```

```
Out[3]:
```

| | Temp | Humidity | Cloud Cover | ANNUAL | Jan-Feb | Mar-May | Jun-Sep | Oct-Dec | avgjune | sub | flood |
|---|------|----------|-------------|--------|---------|---------|---------|---------|------------|-------|-------|
| 0 | 29 | 70 | 30 | 3248.6 | 73.4 | 386.2 | 2122.8 | 666.1 | 274.866667 | 649.9 | 0 |
| 1 | 28 | 75 | 40 | 3326.6 | 9.3 | 275.7 | 2403.4 | 638.2 | 130.300000 | 256.4 | 1 |
| 2 | 28 | 75 | 42 | 3271.2 | 21.7 | 336.3 | 2343.0 | 570.1 | 186.200000 | 308.9 | 0 |
| 3 | 29 | 71 | 44 | 3129.7 | 26.7 | 339.4 | 2398.2 | 365.3 | 366.066667 | 862.5 | 0 |
| 4 | 31 | 74 | 40 | 2741.6 | 23.4 | 378.5 | 1881.5 | 458.1 | 283.400000 | 586.9 | 0 |

```
In [4]: print(dataset.shape)

(115, 11)

In [5]: print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    Temp        115 non-null    int64
1    Humidity    115 non-null    int64
2    Cloud Cover  115 non-null    int64
3    ANNUAL      115 non-null    float64
4    Jan-Feb     115 non-null    float64
5    Mar-May     115 non-null    float64
6    Jun-Sep     115 non-null    float64
7    Oct-Dec     115 non-null    float64
8    avgjune     115 non-null    float64
9    sub         115 non-null    float64
10   flood       115 non-null    int64
dtypes: float64(7), int64(4)
memory usage: 10.0 KB
None

In [6]: dataset.columns

Out[6]: Index(['Temp', 'Humidity', 'Cloud Cover', 'ANNUAL', 'Jan-Feb', 'Mar-May',
              'Jun-Sep', 'Oct-Dec', 'avgjune', 'sub', 'flood'],
              dtype='object')

In [7]: dataset.describe().T

Out[7]:
```


Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

In [7]: dataset.describe().T

Out[7]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|-------------|-------|-------------|------------|--------|-------------|-------------|-------------|-------------|
| Temp | 115.0 | 29.600000 | 1.122341 | 28.0 | 29.000000 | 30.000000 | 31.000000 | 31.000000 |
| Humidity | 115.0 | 73.852174 | 2.947623 | 70.0 | 71.000000 | 74.000000 | 76.000000 | 79.000000 |
| Cloud Cover | 115.0 | 36.286957 | 4.330158 | 30.0 | 32.500000 | 36.000000 | 40.000000 | 44.000000 |
| ANNUAL | 115.0 | 2925.487826 | 422.112193 | 2068.8 | 2627.900000 | 2937.500000 | 3164.100000 | 4257.800000 |
| Jan-Feb | 115.0 | 27.739130 | 22.361032 | 0.3 | 10.250000 | 20.500000 | 41.600000 | 98.100000 |
| Mar-May | 115.0 | 377.253913 | 151.091850 | 89.9 | 276.750000 | 342.000000 | 442.300000 | 915.200000 |
| Jun-Sep | 115.0 | 2022.840870 | 386.254397 | 1104.3 | 1768.850000 | 1948.700000 | 2242.900000 | 3451.300000 |
| Oct-Dec | 115.0 | 497.636522 | 129.860643 | 166.6 | 407.450000 | 501.500000 | 584.550000 | 823.300000 |
| avgjune | 115.0 | 218.100870 | 62.547597 | 65.6 | 179.666667 | 211.033333 | 263.833333 | 366.066667 |
| sub | 115.0 | 439.801739 | 210.438813 | 34.2 | 295.000000 | 430.600000 | 577.650000 | 982.700000 |
| flood | 115.0 | 0.139130 | 0.347597 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |

In [8]: #checking null values
dataset.isnull().any()

Out[8]:

| | |
|-------------|-------|
| Temp | False |
| Humidity | False |
| Cloud Cover | False |
| ANNUAL | False |
| Jan-Feb | False |
| Mar-May | False |
| Jun-Sep | False |

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

flood False
dtype: bool

In [9]: #Correlation
dataset.corr()

Out[9]:

| | Temp | Humidity | Cloud Cover | ANNUAL | Jan-Feb | Mar-May | Jun-Sep | Oct-Dec | avgjune | sub | flood |
|-------------|-----------|-----------|-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Temp | 1.000000 | -0.012727 | -0.046568 | -0.063014 | -0.080076 | 0.099519 | -0.081965 | -0.063034 | -0.019751 | -0.088331 | -0.080946 |
| Humidity | -0.012727 | 1.000000 | 0.085824 | -0.054767 | -0.185965 | -0.101232 | -0.029583 | 0.059739 | 0.017656 | 0.029981 | 0.020250 |
| Cloud Cover | -0.046568 | 0.085824 | 1.000000 | 0.051166 | 0.004376 | 0.096645 | 0.010833 | 0.020966 | -0.089843 | -0.106455 | 0.089801 |
| ANNUAL | -0.063014 | -0.054767 | 0.051166 | 1.000000 | 0.033639 | 0.387790 | 0.861190 | 0.232069 | 0.474644 | 0.220009 | 0.626874 |
| Jan-Feb | -0.080076 | -0.185965 | 0.004376 | 0.033639 | 1.000000 | 0.066479 | 0.001178 | -0.143670 | 0.164691 | 0.201266 | -0.084446 |
| Mar-May | 0.099519 | -0.101232 | 0.096645 | 0.387790 | 0.066479 | 1.000000 | -0.029007 | 0.171805 | 0.019183 | -0.475750 | -0.017598 |
| Jun-Sep | -0.081965 | -0.029583 | 0.010833 | 0.861190 | 0.001178 | -0.029007 | 1.000000 | -0.141467 | 0.511113 | 0.431997 | 0.705202 |
| Oct-Dec | -0.063034 | 0.059739 | 0.020966 | 0.232069 | -0.143670 | 0.171805 | -0.141467 | 1.000000 | -0.028055 | -0.050862 | -0.024852 |
| avgjune | -0.019751 | 0.017656 | -0.089843 | 0.474644 | 0.164691 | 0.019183 | 0.511113 | -0.028055 | 1.000000 | 0.780445 | 0.379778 |
| sub | -0.088331 | 0.029981 | -0.106455 | 0.220009 | 0.201266 | -0.475750 | 0.431997 | -0.050862 | 0.780445 | 1.000000 | 0.349828 |
| flood | -0.080946 | 0.020250 | 0.089801 | 0.626874 | -0.084446 | -0.017598 | 0.705202 | -0.024852 | 0.379778 | 0.349828 | 1.000000 |

In [10]: import seaborn as sns
fig=plt.gcf()
fig.set_size_inches(15,15)
fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',
linewidths=1,linewidth='k',square=True,
cbar_kws={'label':'Correlation'})

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

```

In [10]: import seaborn as sns
fig=plt.gcf()
fig.set_size_inches(15,15)
fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',
                linewidths=1,linecolor='k',square=True,
                mask=False, vmin=-1, vmax=1,
                cbar_kws={"orientation": "vertical"},cbar=True)

```

Temp Humidity Cloud Cover ANNUAL Jan-Feb Mar-May Jun-Sep avgjune sub flood

```

In [12]: dataset.head()
Out[12]:

```

| | Temp | Humidity | Cloud Cover | ANNUAL | Jan-Feb | Mar-May | Jun-Sep | avgjune | sub | flood |
|---|------|----------|-------------|--------|---------|---------|---------|------------|-------|-------|
| 0 | 29 | 70 | 30 | 3248.6 | 73.4 | 386.2 | 2122.8 | 274.866667 | 649.9 | 0 |
| 1 | 28 | 75 | 40 | 3326.6 | 9.3 | 275.7 | 2403.4 | 130.300000 | 256.4 | 1 |
| 2 | 28 | 75 | 42 | 3271.2 | 21.7 | 336.3 | 2343.0 | 186.200000 | 308.9 | 0 |
| 3 | 29 | 71 | 44 | 3129.7 | 26.7 | 339.4 | 2398.2 | 386.066667 | 862.5 | 0 |
| 4 | 31 | 74 | 40 | 2741.6 | 23.4 | 378.5 | 1881.5 | 283.400000 | 586.9 | 0 |

```

In [13]: dataset['flood'].value_counts()
Out[13]:
0    99
1    16
Name: flood, dtype: int64

In [14]: #independent features
x=dataset.iloc[:,2:7].values

In [15]: #dependent feature
y=dataset.iloc[:,9:].values

In [16]: x.shape
Out[16]: (115, 5)

In [17]: y.shape

```

Type here to search

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%2... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

In [18]: #split the data into train and test set from our x and y
#import train_test_split function
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=10)

In [19]: #checking the shape of our 4 variables
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

(86, 5)
(29, 5)
(86, 1)
(29, 1)

In [20]: #import StandardScaler
from sklearn.preprocessing import StandardScaler
#create object to StandardScaler class
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)

In [21]: #import dump class from joblib
from joblib import dump
dump(sc,"transform.save")

Out[21]: ['transform.save']

In [22]: from sklearn.ensemble import GradientBoostingClassifier
xg_cla = GradientBoostingClassifier(learning_rate = 0.1,
max_depth = 5, n_estimators = 10)

In [23]: #fit the model
xg_cla.fit(x_train,y_train)

E:\anaconda\lib\site-packages\sklearn\utils\validation.py:63: 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().
return f(*args, **kwargs)

Out[23]: GradientBoostingClassifier(max_depth=5, n_estimators=10)

In [24]: #predictions with unseen data by model
y_pred_xgb = xg_cla.predict(x_test)
y_pred_xgb

Out[24]: array([1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0], dtype=int64)

In [25]: #checking the accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix
acc=accuracy_score(y_test,y_pred_xgb)
acc

Out[25]: 0.9655172413793104

In [26]: #summary of predictions

Type here to search 29°C 2:03 AM 11-Apr-22

Type here to search 29°C 2:15 AM 11-Apr-22

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

Out[26]: array([[25, 1],
[0, 3]], dtype=int64)

In [27]: `from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
print("RMSE: %f" % (rmse))`

RMSE: 0.185695

In [28]: `dataset.head()`

Out[28]:

| | Temp | Humidity | Cloud Cover | ANNUAL | Jan-Feb | Mar-May | Jun-Sep | avgjune | sub | flood |
|---|------|----------|-------------|--------|---------|---------|---------|------------|-------|-------|
| 0 | 29 | 70 | 30 | 3248.6 | 73.4 | 386.2 | 2122.8 | 274.866667 | 649.9 | 0 |
| 1 | 28 | 75 | 40 | 3326.6 | 9.3 | 275.7 | 2403.4 | 130.300000 | 256.4 | 1 |
| 2 | 28 | 75 | 42 | 3271.2 | 21.7 | 336.3 | 2343.0 | 186.200000 | 308.9 | 0 |
| 3 | 29 | 71 | 44 | 3129.7 | 26.7 | 339.4 | 2398.2 | 366.066667 | 862.5 | 0 |
| 4 | 31 | 74 | 40 | 2741.6 | 23.4 | 378.5 | 1881.5 | 283.400000 | 586.9 | 0 |

In [29]: `rand_pred = xg_cla.predict(sc.transform([[30,3248.6,73.4,386.2,2122.8]]))
rand_pred`

Out[29]: array([0], dtype=int64)

In [30]: `rand_pred1 = xg_cla.predict(sc.transform([[40,3326.6,9.3,275.7,2403.4]]))
rand_pred1`

Out[30]: array([1], dtype=int64)

Windows Type here to search 29°C 2:15 AM 11-Apr-22

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood F x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

Out[29]: array([0], dtype=int64)

In [30]: `rand_pred1 = xg_cla.predict(sc.transform([[40,3326.6,9.3,275.7,2403.4]]))
rand_pred1`

Out[30]: array([1], dtype=int64)

In [31]: `from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)`

Out[31]: DecisionTreeClassifier()

In [32]: `y_predict = dtc.predict(x_test)
y_predict`

Out[32]: array([1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0], dtype=int64)

In [33]: `from sklearn.metrics import accuracy_score, confusion_matrix`

In [34]: `acc=accuracy_score(y_test,y_predict)
acc`

Out[34]: 0.9655172413793104

In [35]: `acc=accuracy_score(y_test,y_pred_xgb)
acc`

Out[35]: 0.9655172413793104

Windows Type here to search 29°C 2:18 AM 11-Apr-22

Student x SI-8899 x (4) What x Anaconda x New Tab x Desktop x pavan g x Floods F x Flood pr x Flood Fc x +

localhost:8888/notebooks/Desktop/39110758-Y%20PAVAN%20GANESH%20REDDY-FLOOD%20PREDICTION%20USING%20MACHINE%20LEARNING/Training/pavan%20... Logout

jupyter pavan ganesh 7889 Last Checkpoint: 8 hours ago (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

```

acc

Out[34]: 0.9655172413793104

In [35]: acc=accuracy_score(y_test,y_pred_xgb)
acc

Out[35]: 0.9655172413793104

In [36]: cm1=confusion_matrix(y_test,y_predict)
cm1

Out[36]: array([[25,  1],
               [ 0,  3]], dtype=int64)

In [37]: cm2=confusion_matrix(y_test,y_pred_xgb)
cm2

Out[37]: array([[25,  1],
               [ 0,  3]], dtype=int64)

In [38]: #saving the file
from joblib import dump
dump(xg_cla,'floods.save')

Out[38]: ['floods.save']

In [ ]:

```

Type here to search

Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Pavan\Desktop\Y PAVAN GANESH-FLOOD PREDICTION\Flask

C:\Users\Pavan\Desktop\Y PAVAN GANESH-FLOOD PREDICTION\Flask\app.py

```

temp.py x app.py x
5 #import load from joblib to load the saved model file
6 from joblib import load
7
8 app=Flask(__name__) # our flask app
9 #load model file
10 model =load('C:/Users/Pavan/Desktop/Y PAVAN GANESH-FLOOD PREDICTION/Flask/floods.save')
11 sc=load('C:/Users/Pavan/Desktop/Y PAVAN GANESH-FLOOD PREDICTION/Flask/transform.save')
12
13
14 @app.route('/') # rendering the html template
15 def home():
16     return render_template('home.html')
17 @app.route('/predict') # rendering the html template
18 def index():
19     return render_template("index.html")
20
21
22 @app.route('/data_predict', methods=['POST']) # route for our prediction
23 def predict():
24     temp = request.form['temp']
25     Hum = request.form['Hum']
26     db = request.form['db']
27     ap = request.form['ap']
28     aal = request.form['aal']
29
30     data = [[float(temp),float(Hum),float(db),float(ap),float(aal)]]
31     prediction = model.predict(sc.transform(data))
32     output=prediction[0]
33     if(output==0):
34         return render_template('noChance.html', prediction='No possiblity of severe flood')
35     else:
36         return render_template('chance.html', prediction='possiblity of severe flood')
37
38 if __name__ == '__main__':
39     app.run(debug=False)

```

File Explorer

| Name | Date Modified |
|----------------|-------------------|
| templates | 10-Apr-22 7:38 PM |
| app.py | 11-Apr-22 1:48 AM |
| floods.save | 06-Apr-22 6:55 PM |
| transform.save | 06-Apr-22 5:57 PM |

Help Variable Explorer Plots Files

Console I/A x

```

C:\Users\Pavan\Desktop\Y PAVAN GANESH-FLOOD PREDICTION\Flask>python app.py
Warning: Using a legacy version of joblib. Please consider upgrading with the latest release. See the
Warning: estimator StandardScaler from version 0.23.0 when using version 0.24.2. This might
Warning: lead to breaking code or invalid results. Use at your own risk.
Warning: warnings.warn(
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Apr/2022 01:48:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:48:58] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Apr/2022 01:49:00] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:50:10] "POST /data_predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:51:09] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:51:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:52:18] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:52:27] "POST /data_predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:52:34] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 01:52:42] "POST /data_predict HTTP/1.1" 200 -

```

Python console History

LSP Python: ready conda: base (Python 3.9.7) Line 11, Col 85 ASCII CRLF RW Mem 59%

Type here to search

