# Evaluation of the Flood Forecasting Capability of a Machine Learning Model

Dr. S. W. Mohod[1], Pallavi Bangare[2], Unnati Bokade[3], Mayur Dhamankar[4], Chetan Dhawale[5]
[1,2,3,4,5]*Bapurao Deshmukh College of Engineering, Sewagram, Department of Computer Engineering*

**Abstract-Predicting floods involves forecasting future levels of water or runs at one or more areas along a river system over a given time period. Flood control measures necessitate precise and consistent forecasting in order to plan, implement, and rehab. In spite of problems with data scarcity, soft computing technique-based models for operational flood forecasting systems are frequently better in terms of accuracy and dependability. When a significant amount of water overflows onto a plot of land, flooding occurs. Based on water level or discharges from hydraulic structures, the flood forecasting (FF) system will give an advisory. In our project we have collect kerela dataset based on kaggle.com website. Then we have to apply preprocessing technique then cleaning the null values from the dataset. Then data can be split into two dataset that is training and testing dataset. we have to used training and testing techniques to analyse the dataset and to identify final accuracy and improve model performance then we have to show the results that are flood may happen or flood may not happen.**

**Keywords-Rainfall, SVM classifier, Naïve Bayes classifier, Decision tree, KNN classifier.**

## 1.INTRODUCTION

The affected population can be evacuated to safer areas, multipurpose reservoirs can be controlled and operated with a focus on controlling incoming floods, and structural measures for mitigating flood damages can all be done with the aid of flood forecasting systems. GIS tools and the use of hydrological models are both very helpful in this regard. To aid in decision-making, models are frequently used for flood forecasting and hydrological simulation. Models are mathematical representations of the hydrological processes that take place in the basin's real-world dynamics. When developing a flood forecasting model, many factors must be considered, including input data, parameters, structures, and the scales at which forecasting is required. Flood forecasting is the process of estimating and forecasting flooding based on known river basin characteristics, and it is used to protect property, human life, and the environment. Floods are without a doubt the most devastating natural disasters, affecting many regions worldwide each year. In recent decades, flood damage has increased exponentially. The situation is being exacerbated by increased rainfall frequency, changes in upstream land use, and a growing concentration of people and assets in flood-prone areas. Floods are the most common natural disasters that cause significant economic damage in developing countries. Significant efforts have been and continue to be made at the national and community levels to implement appropriate countermeasures, both structural and non-structural, in order to reduce the ongoing threat of water-related disasters.

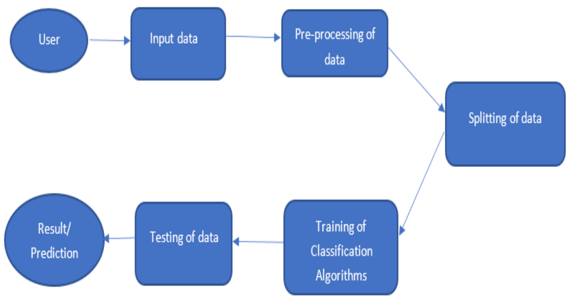


Fig1 Architecture diagram of flood prediction

## 2.PROCEDURE FOR IMPLEMENTING FLOOD PREDICTION MODEL

Some of the flood resource variables that can be used to classify flood prediction are the level of water, stream flood, moisture in the soil, precipitation-discharge, the process of precipitation river inflow, peak flow, river flow, rainfall-runoff, flash flood, rainfall, stream flow, seasonal stream flow, flood peak discharge, urban flood, plain flood, groundwater level, rainfall stage, flood analysis of frequency, flood quintiles, surge level, extremely flow, storm surge, storm rainfall, and every day flows. In runoff and flood modelling, Rainfall and water cycle spatial analysis were the most important factors influencing flood resource. As a result, quantitative rainfall prediction is commonly used for flood prediction, particularly for flash floods or short-term floods that take avalanches, slush flow, and melting snow into account. Rainfall forecasts have been display to be insufficient for accurate flood prediction. Estimates of soil moisture in a catchment, for example, are used to predict stream flow in addition to rainfall In an extensive forecasting of floods situation. Other flood resource variables were taken into account, including regardless of how important accurate precipitation forecasting is. This literature review's methodology does exactly what it says in order to include the search queries with the most effective flood resource variables. Peer-reviewed articles in the most prestigious subject areas are examined to determine the state of the art of ML techniques for flood prediction. Priority was given to articles that included a performance evaluation and comparison of machine learning methods in order to determine which ML methods discovered through search queries using the search strategy performed better in specific applications.

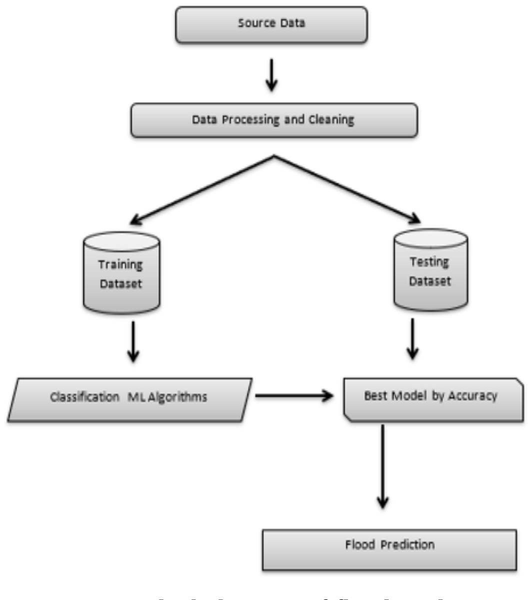


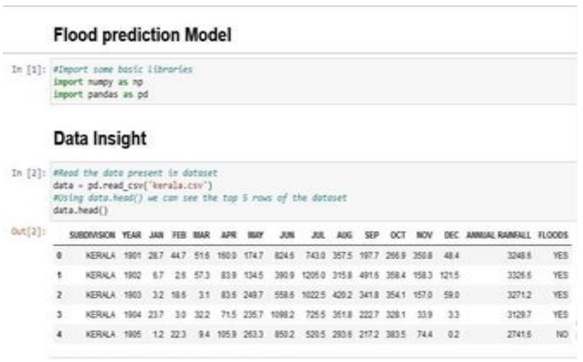
Fig2. Block diagram of flood prediction

## 3. MODEL CLASSIFIER

SVM classifiers, Naive Bayes classifiers, logistic regression models, decision trees, and the K-Nearest Neighbours algorithm are just a few of the classification models that have been used. The SVM Classifier, one of the most widely used algorithms for categorising and forecasting data, was designed with supervised learning in mind and is primarily used in machine learning for categorising problems. Naïve Bayes is a
subset of "probabilistic classifiers" known as naive Bayes is depend on the connection between the Bayes theorem and ardent expectations of structural independence. Although they are fundamental Bayesian network models, high accuracy levels can be attained when the kernel density approximation is applied. The outcome of a class-based variance is predicted
by Logistic Regression. As a result, a grouping or a different value will be produced. True or false, yes or no, zero or one, etc., but rather than giving a straightforward value like zero or one, it gives likely values between zero and one. The Decision Tree method is a supervised learning approach that can be used to solve classification and regression problems. Internal nodes represent dataset elements, branches represent decision-making rules, and leaf nodes represent outcomes. This supervised learning method is non-parametric. It can be used for classification and regression. Both classification and regression take as input the k closest training instances in the data set. K-NN can be used for regression or classification depending on the results. As a result, a K-NN classification class membership is obtained.

## 4. PERFORMANCE AND RESULT ANALYSIS

There are following steps for executing results of flood prediction:
Step1: Data Collection: Our first step is toGather the ('../input/kerela-flood/kerala.csv') dataset from the local area network, specifically kaggle.gov.com. Then, using a Pandas object, we imported that dataset by reading a csv file. The top 5 rows from the dataset were then chosen for analysis. Based on the information provided, we calculated the annual rainfall from 1901 to 1905 to determine whether or not a flood would occur. Additionally, SCR01 below from the flood prediction model forecasts the likelihood of flooding.



**Flood prediction Model**

```
In [1]: #Import some basic libraries
        import numpy as np
        import pandas as pd
```

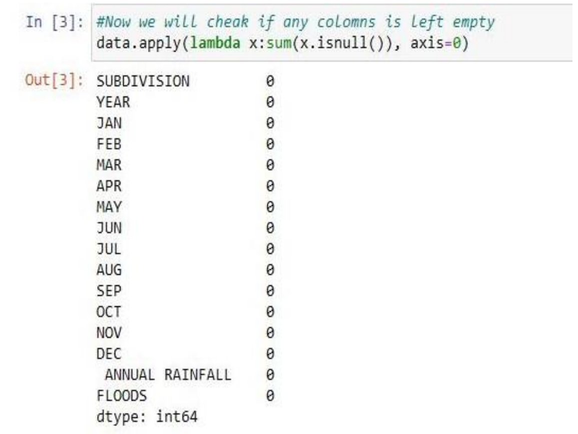
**Data Insight**

```
In [2]: #Read the data present in dataset
        data = pd.read_csv('kerala.csv')
        #Using data.head() we can see the top 5 rows of the dataset
        data.head()
Out[2]:
```

| | SUBDIVISION | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC | ANNUAL RAINFALL | FLOODS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | KERALA | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 350.8 | 48.4 | 3248.6 | YES |
| 1 | KERALA | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 158.3 | 121.5 | 3326.6 | YES |
| 2 | KERALA | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 157.0 | 59.0 | 3271.2 | YES |
| 3 | KERALA | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 33.9 | 3.3 | 3129.7 | YES |
| 4 | KERALA | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 74.4 | 0.2 | 2741.6 | NO |

SCR01: Importing dataset and analysis

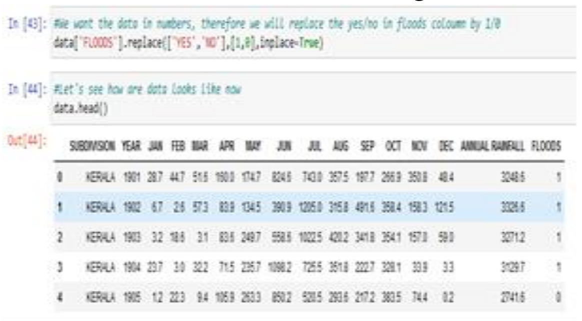
Step2: Technique for Preprocessing Data: As shown in the following SCR2, after data import, we used a data preprocessing technique to eliminate duplicate values from the data set.



```
In [3]:  #Now we will cheak if any colomns is left empty
         data.apply(lambda x:sum(x.isnull()), axis=0)

Out[3]:  SUBDIVISION          0
         YEAR                 0
         JAN                  0
         FEB                  0
         MAR                  0
         APR                  0
         MAY                  0
         JUN                  0
         JUL                  0
         AUG                  0
         SEP                  0
         OCT                  0
         NOV                  0
         DEC                  0
          ANNUAL RAINFALL     0
         FLOODS               0
         dtype: int64
```

SCR02: Cleaning null values

Step3: Converting Boolean values to integer values: After that, we replace Yes/No in floods with 1/0 to convert boolean values to integers. As shown in SCR03, means 1 denotes "flood came" and 0 denotes "flood not came." From the aforementioned dataset, we were able to determine that floods occurred in the years 1901, 1902, 1903, 1904, and 1905. The results of the annual rainfall are displayed as a 1 or a 0. SCR03 below illustrates how the flood prediction model forecasts whether there will be flooding or not.

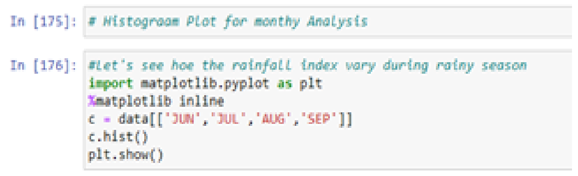


```
In [43]: #We want the data in numbers, therefore we will replace the yes/no in floods coloumn by 1/0
         data['FLOODS'].replace(['YES','NO'],[1,0],inplace=True)

In [44]: #Let's see how are data looks like now
         data.head()
Out[44]:
```

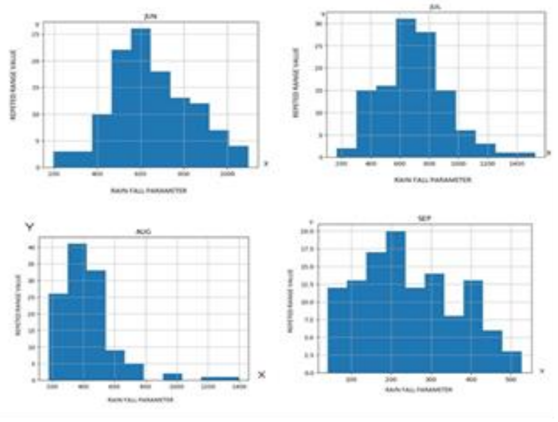
| | SUBDIVISION | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC | ANNUAL RAINFALL | FLOODS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | KERALA | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 350.8 | 48.4 | 3248.6 | 1 |
| 1 | KERALA | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 158.3 | 121.5 | 3326.6 | 1 |
| 2 | KERALA | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 157.0 | 59.0 | 3271.2 | 1 |
| 3 | KERALA | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 33.9 | 3.3 | 3129.7 | 1 |
| 4 | KERALA | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 74.4 | 0.2 | 2741.6 | 0 |

SCR03: Converting Boolean values to integer

Step4: visualization of data: After preprocessing the data, we used a visualization technique to identify floodsin the months of June, July, August, and September. We discovered that the June rainfall ranges from 400 mm to 800 mm. The range of rainfall in July is 400 to 1000 mm, which is greater than in June, and the range of rainfall in August is less than in the two preceding months. In SCR04 The graph's overall observation led us to the conclusion that July was the month with the most rainfall.

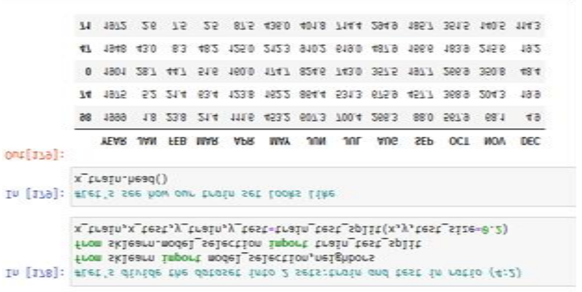


```
In [175]: # Histograom Plot for monthy Analysis

In [176]: #Let's see hoe the rainfall index vary during rainy season
          import matplotlib.pyplot as plt
          %matplotlib inline
          c = data[['JUN','JUL','AUG','SEP']]
          c.hist()
          plt.show()
```



SCR04: Analysing Dataset by Visualization

Step5: Models for testing and training: After data visualisation, we used training and testing techniques to analyse the dataset to identify final accuracy and improve model performance using training dataset. In order to do that, we divided the data into 20% for testing and 80% for training, as shown inSCR05. Then, in order to predict the flood, we used the KNN algorithm, Decision tree algorithm, Logistic regression algorithm based on SVM, random Forest classifier, and Naive Bayesian classifier.

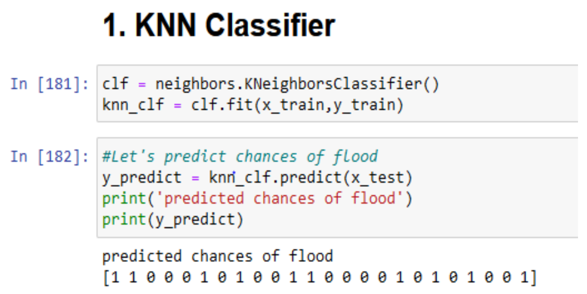


```
In [118]: #Let's divide the dataset into 2 sets:train and test in ratio (4:2)
          from sklearn import model_selection,neighbors
          from sklearn.model_selection import train_test_split
          x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

In [119]: #Let's see how our train set looks like
          x_train.head()
Out[119]:
```

| | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

SCR05: Divide the dataset in two set training and testing

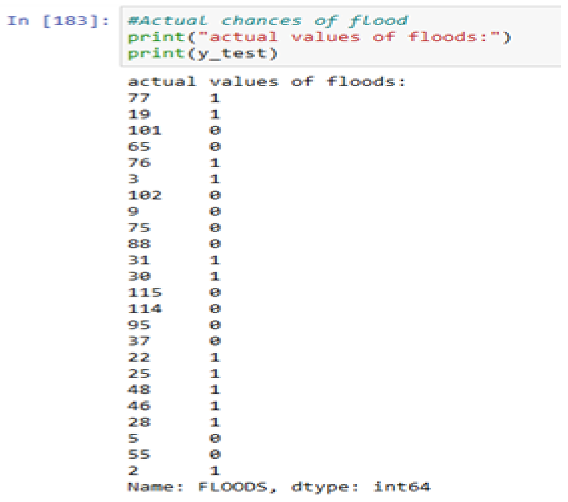
Step6: Applying KNN classifier: Then, using the training dataset, we applied the KNN classifier to determine the actual likelihood of a flood. By using a KNN classifier, we were able to determine that the likelihood of a flood occurring is greater than its likelihood of not occurring, as shown in SCR06 and SCR07. Flood will occur if 1 is present, while flood won't occur if 0 is present. We also discovered that the KNN classifier has an accuracy rate of 83.33% for predicting floods. As shown in SCR08 below.



**1. KNN Classifier**

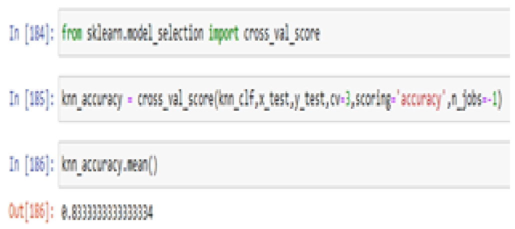
```
In [181]: clf = neighbors.KNeighborsClassifier()
          knn_clf = clf.fit(x_train,y_train)

In [182]: #Let's predict chances of flood
          y_predict = knn_clf.predict(x_test)
          print('predicted chances of flood')
          print(y_predict)

          predicted chances of flood
          [1 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1]
```

SCR06: Predicted chances of flood by using KNN algorithm



```
In [183]: #Actual chances of flood
          print("actual values of floods:")
          print(y_test)

          actual values of floods:
          77     1
          19     1
          101    0
          65     0
          76     1
          3      1
          102    0
          9      0
          75     0
          88     0
          31     1
          30     1
          115    0
          114    0
          95     0
          37     0
          22     1
          25     1
          48     1
          46     1
          28     1
          5      0
          55     0
          2      1
          Name: FLOODS, dtype: int64
```

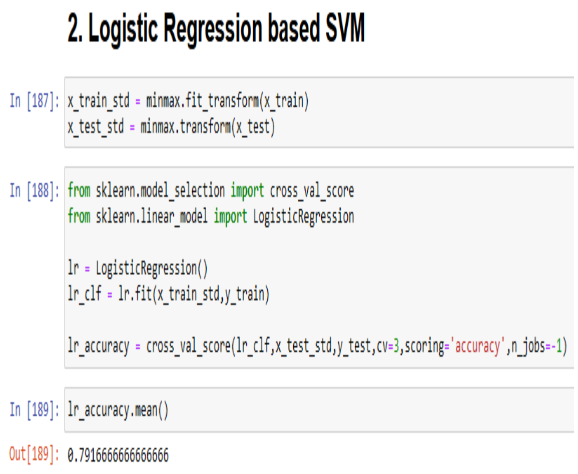
FigSCR07: Actual chances of flood by using KNN Algorithm



```
In [184]: from sklearn.model_selection import cross_val_score

In [185]: knn_accuracy = cross_val_score(knn_clf,x_test,y_test,cv=3,scoring='accuracy',n_jobs=-1)

In [186]: knn_accuracy.mean()

Out[186]: 0.8333333333333334
```

SCR08: Accuracy of flood by using KNN Algorithm

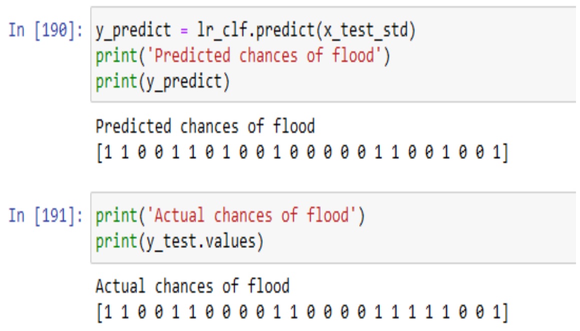
Step7: Applying logistic regression algorithm: After applying KNN algorithm for training phase, we applied logistic regression algorithm based on svm for training purpose as shown in following SCR09. After that we compared this logistic algorithm with other algorithm by calculating their accuracy for identifying prediction of flood . We identified that svm algorithm has 79.16% accuracy for tarining dataset. After we applied this LR svm algorithm for prediction of floodas shown in SCR10 i.e ‘ 1’ indicates flood happned and ‘ 0’ indicates flood not happen. Also we identified for testing purpose, accuracy is 83.33%, recall score is 75.00% and roc score is 83.33% as shown in SCR11. For prediction of flood , accuracy of this algorithm is 91.67% identified.



**2. Logistic Regression based SVM**

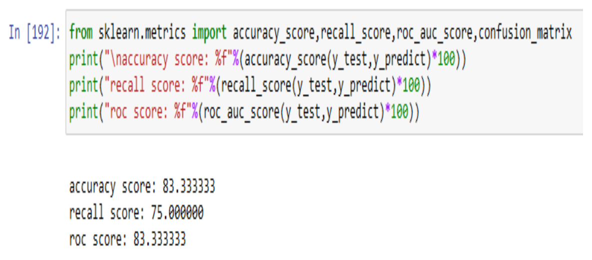
```
In [187]: x_train_std = minmax.fit_transform(x_train)
          x_test_std = minmax.transform(x_test)

In [188]: from sklearn.model_selection import cross_val_score
          from sklearn.linear_model import LogisticRegression

          lr = LogisticRegression()
          lr_clf = lr.fit(x_train_std,y_train)

          lr_accuracy = cross_val_score(lr_clf,x_test_std,y_test,cv=3,scoring='accuracy',n_jobs=-1)

In [189]: lr_accuracy.mean()

Out[189]: 0.7916666666666666
```

SCR09:Logistic regression based on SVM



```
In [190]: y_predict = lr_clf.predict(x_test_std)
          print('Predicted chances of flood')
          print(y_predict)

          Predicted chances of flood
          [1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1]

In [191]: print('Actual chances of flood')
          print(y_test.values)

          Actual chances of flood
          [1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1]
```

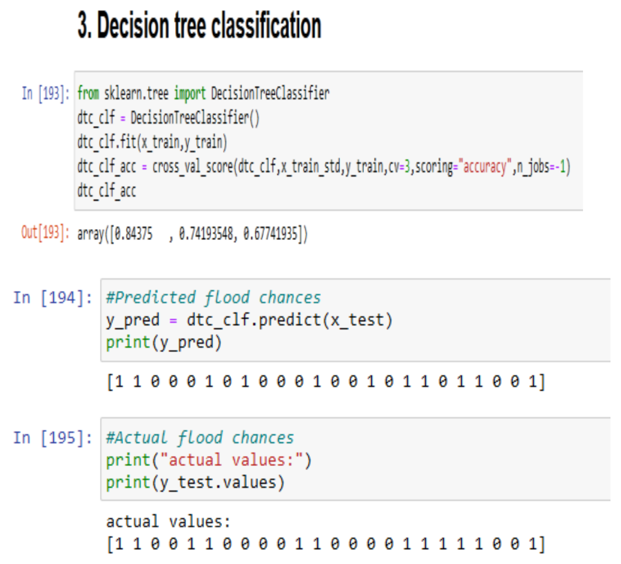
SCR10: Predicted chances of flood and Actual chances of flood by using logistic regression algorithm



```
In [192]: from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
          print("\naccuracy score: %f"%(accuracy_score(y_test,y_predict)*100))
          print("recall score: %f"%(recall_score(y_test,y_predict)*100))
          print("roc score: %f"%(roc_auc_score(y_test,y_predict)*100))

          accuracy score: 83.333333
          recall score: 75.000000
          roc score: 83.333333
```

SCR11: Accuracy of flood by using logistic regression Algorithm

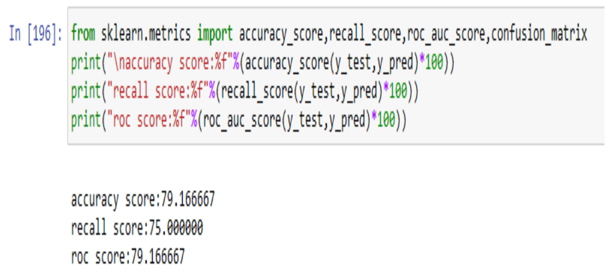
Step8: Applying Decision tree algorithm: After applying logistic regression algorithm for training phase, we applied Decision tree algorithm for training phase as well as testing phase as shown in following SCR12. After that we compared this Decision tree algorithm with other algorithm by calculating their accuracy for identifying prediction of flood. We identified that Decision tree algorithm has 74.53% accuracy for training dataset. and 79.16% accuracy of testing dataset. After we applied this Decision tree algorithm for prediction of flood as shown in SCR12 i.e ‘ 1’ indicates flood happned and ‘ 0’ indicates flood not happen. Also we identified for recall score is 75.00% and roc score is 79.16% as shown in SCR13 . For prediction of flood , accuracy of this algorithm is 58.33% identified.



**3. Decision tree classification**

```
In [193]: from sklearn.tree import DecisionTreeClassifier
          dtc_clf = DecisionTreeClassifier()
          dtc_clf.fit(x_train,y_train)
          dtc_clf_acc = cross_val_score(dtc_clf,x_train_std,y_train,cv=3,scoring="accuracy",n_jobs=-1)
          dtc_clf_acc

Out[193]: array([0.84375   , 0.74193548, 0.67741935])

In [194]: #Predicted flood chances
          y_pred = dtc_clf.predict(x_test)
          print(y_pred)

          [1 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1]

In [195]: #Actual flood chances
          print("actual values:")
          print(y_test.values)

          actual values:
          [1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1]
```

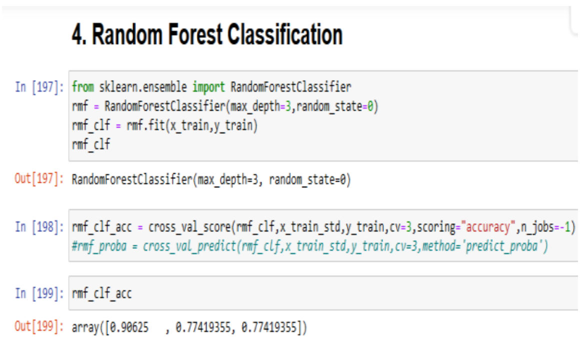
SCR12: Predicted chances of flood and Actual chances of flood by using Decision tree algorithm



```
In [196]: from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
          print("\naccuracy score:%f"%(accuracy_score(y_test,y_pred)*100))
          print("recall score:%f"%(recall_score(y_test,y_pred)*100))
          print("roc score:%f"%(roc_auc_score(y_test,y_pred)*100))

          accuracy score:79.166667
          recall score:75.000000
          roc score:79.166667
```

SCR13: Accuracy of flood by using Decision tree Algorithm

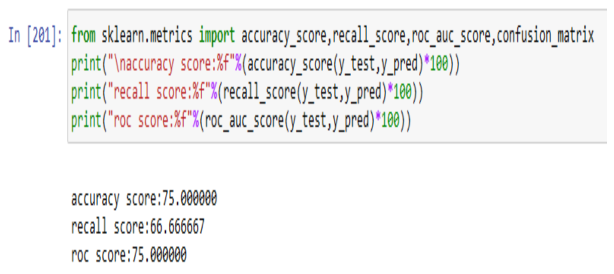
Step9: Applying random forest algorithm: After applying logistic regression algorithm for training phase, we applied Random forest algorithm for training phase as well as testing phase as shown in followingSCR14. After that we compared this Random forest algorithm with other algorithm by calculating their accuracy for identifying prediction of flood. We identified that Decision tree algorithm has 81.82% accuracy for training dataset. and 75.00% accuracy of testing dataset. After we applied this Decision tree algorithm for prediction of flood as shown in SCR15 i.e ‘ 1’ indicates flood happned and ‘ 0’ indicates flood not happen. Also we identified for recall score is 66.66% and roc score is 75.00% as shown inSCR15. For prediction of flood , accuracy of this algorithm is 75.00% identified.



**4. Random Forest Classification**

```
In [197]: from sklearn.ensemble import RandomForestClassifier
          rmf = RandomForestClassifier(max_depth=3,random_state=0)
          rmf_clf = rmf.fit(x_train,y_train)
          rmf_clf

Out[197]: RandomForestClassifier(max_depth=3, random_state=0)

In [198]: rmf_clf_acc = cross_val_score(rmf_clf,x_train_std,y_train,cv=3,scoring="accuracy",n_jobs=-1)
          #rmf_proba = cross_val_predict(rmf_clf,x_train_std,y_train,cv=3,method='predict_proba')

In [199]: rmf_clf_acc

Out[199]: array([0.90625   , 0.77419355, 0.77419355])
```

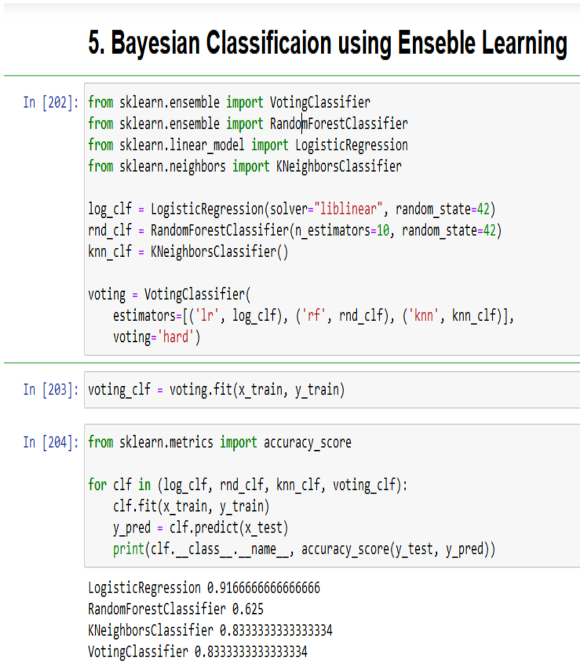
SCR14: Predicted chances of flood and Actual chances of flood by using Random forest classification



```
In [201]: from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
          print("\naccuracy score:%f"%(accuracy_score(y_test,y_pred)*100))
          print("recall score:%f"%(recall_score(y_test,y_pred)*100))
          print("roc score:%f"%(roc_auc_score(y_test,y_pred)*100))

          accuracy score:75.000000
          recall score:66.666667
          roc score:75.000000
```

SCR15: Accuracy of flood by using Random forest classification

Step10: Applying Naïve Bayes algorithm: After applying Random forest algorithm for training phase, we applied Naïve Bayes algorithm for training phase as well as testing phase as shown in following fig17. After that we compared this Naïve Bayes algorithm with other algorithm by calculating their accuracy for identifying prediction of flood. We identified that Decision tree algorithm has 83.33% accuracy for training dataset. and 79.16% accuracy of testing dataset. After we applied this Decision tree algorithm for prediction of flood as shown in SCR16 i.e ‘ 1’ indicates flood happned and ‘ 0’ indicates flood not happen. Also we identified for recall score is 75.00% and roc score is 79.16% as shown in SCR16. For prediction of flood , accuracy of this algorithm is 83.83% identified.

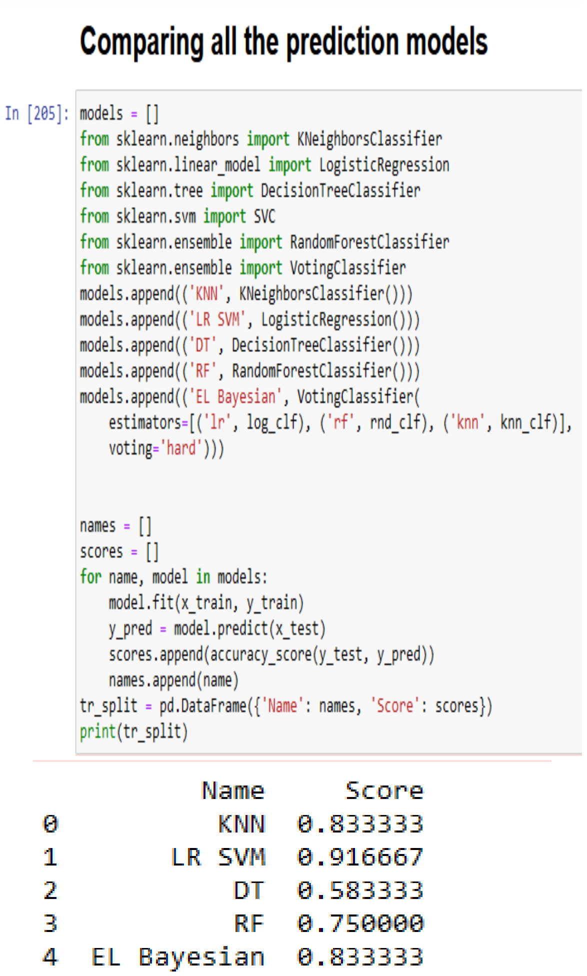


5. Bayesian Classificaion using Enseble Learning

```
In [202]: from sklearn.ensemble import VotingClassifier
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.neighbors import KNeighborsClassifier

          log_clf = LogisticRegression(solver="liblinear", random_state=42)
          rnd_clf = RandomForestClassifier(n_estimators=10, random_state=42)
          knn_clf = KNeighborsClassifier()

          voting = VotingClassifier(
              estimators=[('lr', log_clf), ('rf', rnd_clf), ('knn', knn_clf)],
              voting='hard')

In [203]: voting_clf = voting.fit(x_train, y_train)

In [204]: from sklearn.metrics import accuracy_score

          for clf in (log_clf, rnd_clf, knn_clf, voting_clf):
              clf.fit(x_train, y_train)
              y_pred = clf.predict(x_test)
              print(clf.__class__.__name__, accuracy_score(y_test, y_pred))

          LogisticRegression 0.9166666666666666
          RandomForestClassifier 0.625
          KNeighborsClassifier 0.8333333333333334
          VotingClassifier 0.8333333333333334
```

SCR16: Predicted chances of flood and Actual chances of flood by using Naïve Bayes classifier

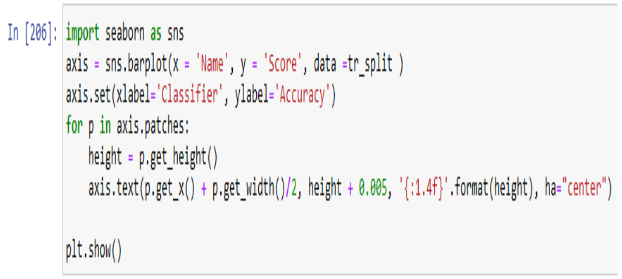
Step11: Comparing prediction of models: After that we compared all the algorithms i.e, KNN Classifier, Logistic Regression based on SVM algorithm, Random Forest algorithm, Naïve Bayes classifier, Decision tree classification algorithm for prediction of flood. After comparing all algorithm, we identified KNN algorithm has 83.33% accuracy for prediction of flood. Logistic Regression SVM algorithm has 91.61% accuracy for prediction of flood. Decision tree classification algorithm has 58.33% accuracy for prediction of flood. Random forest algorithm has 75.00% accuracy for prediction of flood. Naïve Bayes algorithm has 83.33% accuracy for prediction of flood. Then we identified that Logistic Regression based on SVM algorithm has highest accuracy than other algorithm for prediction of flood as shown in the below SCR17 and SCR18. Also we showed comparision of all the algoritms which is used for flood prediction in the form of graph as shown in the below fig. it is conluded that SVM algorim is best for prediction of flood.



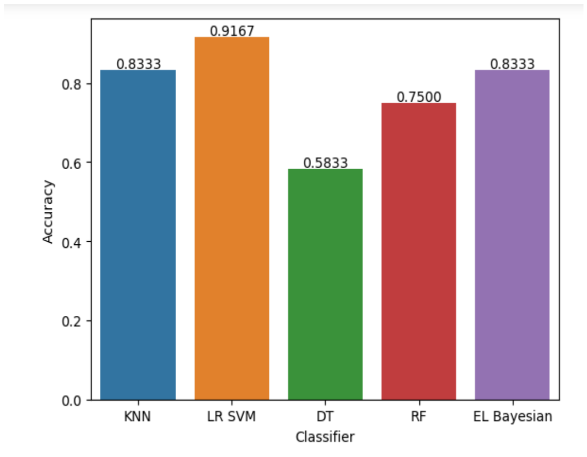
Comparing all the prediction models

```
In [205]: models = []
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.svm import SVC
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.ensemble import VotingClassifier
          models.append(('KNN', KNeighborsClassifier()))
          models.append(('LR SVM', LogisticRegression()))
          models.append(('DT', DecisionTreeClassifier()))
          models.append(('RF', RandomForestClassifier()))
          models.append(('EL Bayesian', VotingClassifier(
              estimators=[('lr', log_clf), ('rf', rnd_clf), ('knn', knn_clf)],
              voting='hard')))


          names = []
          scores = []
          for name, model in models:
              model.fit(x_train, y_train)
              y_pred = model.predict(x_test)
              scores.append(accuracy_score(y_test, y_pred))
              names.append(name)
          tr_split = pd.DataFrame({'Name': names, 'Score': scores})
          print(tr_split)

          Name     Score
0          KNN  0.833333
1       LR SVM  0.916667
2           DT  0.583333
3           RF  0.750000
4  EL Bayesian  0.833333
```

SCR17: Comparision of prediction models



```
In [206]: import seaborn as sns
          axis = sns.barplot(x = 'Name', y = 'Score', data =tr_split )
          axis.set(xlabel='Classifier', ylabel='Accuracy')
          for p in axis.patches:
              height = p.get_height()
              axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")


          plt.show()
```



Type *Markdown* and LaTeX: $\alpha^2$

Type *Markdown* and LaTeX: $\alpha^2$

SCR 18: Accuracy of Prediction of model

## 4. CONCLUSION

The systematic process, it is concluded, will start with data cleaning, address missing values, carry out exploratory analysis, and then produce the model for evaluation. The model with the highest performance and accuracy on test data will be used in the machine learning model after being taken into account. Precision, recall, F1 score, sensitivity, and specificity will all be calculated for each method. The confusion matrices for TP, TN, FP, and FN will also be computed.

## REFERENCE

[1] Ajay Katti, K.V Ashish, Aditya Loke, Kranti Bade, A Pluvial Flood Detection Model Using Machine Learning Techniques and Simulate The flow of Water, Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020) IEEE.
[2] Chinmayee Kinage, Abhishek Kalgutkar, Amruta Parab[7], Sejal Mandora, Sunita Sahu, Machine Learning Based Algorithms for Flood Prediction and Model for Real Time Flood Prediction, 2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA) IEEE.
[3] D.P. Lettenmaier and E.F. Wood, 1993, Hydrological Forecasting, Chapter 26 in Handbook of Hydrology. (D. Maidment, ed.), McGraw-Hill. D.P.
[4] Adeli, H. (2001). Neural networks in civil engineering: 1989–2000. Computer-Aided Civil and Infrastructure Engineering, 16(2), 126-142.
[5] Benoudjit, A., & Guida, R. (2019). A Novel Fully Automated Mapping of the Flood Extent on SAR Images Using a Supervised Classifier. Remote Sensing, 11(7), 779.
[6] Zhao, G., Pang, B., Xu, Z., Peng, D., & Xu, L. (2019). Assessment of urban flood susceptibility using semi-supervised machine learning model. Science of The Total Environment, 659, 940-949.
[7] Bauer, B., & Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. The Annals of Statistics, 47(4), 2261-2285.
[8] Engle, R. F., D. F. Hendry, and J. F. Richard. 1983. Exogeneity. Econometrica 51:277-304
[9] Ruslan, F. A., Samad, A. M., Zain, Z. M., & Adnan, R. (2013, August). Flood prediction using NARX neural network and EKF prediction technique: A comparative study. In 2013 IEEE 3rd International Conference on System Engineering and Technology (pp. 203-208). IEEE.