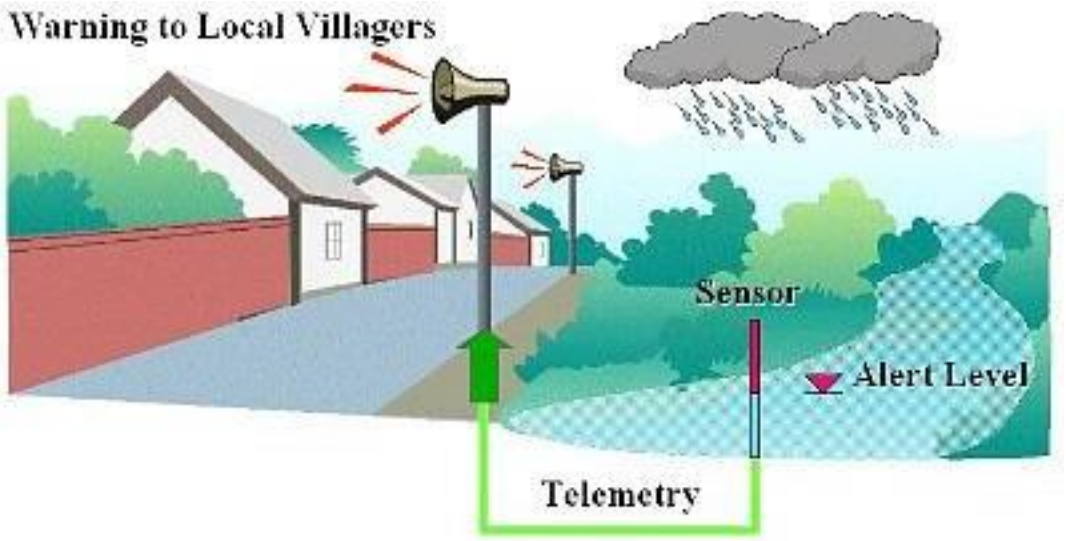
FLOOD MONITORING AND EARLY WARNING SYSTEM

ABISHA RAJ. D. S

Phase 2 Submission Document

Project: Flood Monitoring system



Introduction:

- Floods are natural disasters that can have devastating consequences, including loss of life, property damage, and significant economic impacts.
- To mitigate these risks and provide early warnings to affected communities, flood monitoring systems play a vital role.
- These systems are a combination of technology, data collection methods, and communication networks designed to continuously track and predict flood events.
- In this introduction to flood monitoring systems, we will explore their importance, key components, and their role in safeguarding communities.
- Incorporating predictive modeling into flood monitoring and early warning systems is crucial for improving the accuracy and timeliness of flood predictions, as well as for mitigating the impacts of floods

- Historical flood data refers to records, information, and data collected over time that document past instances of flooding events.
- This data is crucial for various purposes, including flood risk assessment, floodplain management, infrastructure planning, and the development of predictive models for flood monitoring and early warning systems

Content for Project Phase 2 :

Consider incorporating predictive modeling and historical flood data to improve the accuracy of early warnings.

Data Source

A good data source for flood monitoring system should beAccurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: https://www.kaggle.com/code/mukulthakur177/flood-prediction-model/input

SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FLOO
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
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
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
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
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

Data Collection and Preprocessing:

✓ Maintain a historical data archive for reference and analysis, which can be used for model calibration, research, and retrospective flood assessments

✓ Document metadata for collected data, including sensor specifications, data sources, and collection timestamps.

✓ Create documentation that describes data collection processes and data sources for future reference.

Exploratory Data Analysis (EDA):

✓ Gather the relevant data from sensors, gauges, and other sources used in the flood monitoring system.

✓ Calculate summary statistics for each variable, including mean, median, standard deviation, minimum, maximum, and quartiles.

✓ Calculate correlation coefficients (e.g., Pearson, Spearman) to measure the strength and direction of linear relationships between variables.

✓ If the data involves temporal components (e.g., time series of river levels), perform time series analysis to identify trends, seasonality, and cyclic patterns.

Feature Engineering:

- Evaluate the importance of input features and consider feature selection or engineering techniques to improve model performance.
- Techniques like recursive feature elimination or feature importance analysis can help identify critical variables.
- Based on EDA insights, consider creating new features or transformations of existing variables that may be more informative for flood prediction models.

Advanced Regression Techniques:

- Multiple Linear Regression (MLR): MLR is a basic regression technique that models the relationship between multiple independent variables

- Nonlinear Regression: In cases where the relationship between variables is not linear, nonlinear regression techniques such as polynomial regression or exponential regression can be applied

- Random Forest Regression: Random Forest is an ensemble machine learning method that can handle both linear and nonlinear relationships.

- Ridge and Lasso Regression: Ridge and Lasso regression techniques are used for feature selection and regularization.

Model Evaluation and Selection:

- Divide the available dataset into three subsets: training, validation, and testing.
- The training set is used to train the models, the validation set to tune hyperparameters, and the testing set to assess model performance.
- Choose appropriate evaluation metrics that align with the goals of the flood monitoring system.
- Experiment with different modeling techniques suitable for flood prediction, such as linear regression, machine learning algorithms (e.g., Random Forest, Gradient Boosting, Support Vector Machines), or deep learning approaches (e.g., neural networks)

Model Interpretability:

- Identify and rank the importance of input features (e.g., rainfall, river levels, soil moisture) in influencing flood predictions
- PDPs (Partial Dependence Plots) illustrate the relationship between a specific input variable and the model's predictions while keeping other variables constant.
- Create visualizations of the model's inner workings, such as decision trees or neural network activations, to illustrate how information is processed.

Deployment and Prediction:

- Deploy the selected flood prediction model(s) to the central monitoring station.
- Develop user-friendly dashboards and visualization tools for monitoring real-time data and model outputs.

Program:

Flood Prediction Model

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Sample dataset (replace with your own dataset)
data = pd.read_csv("flood_data.csv")

# Data preprocessing
# - Handle missing values, outliers, and feature engineering if needed
```

```
# Split the data into training and testing sets
X = data[['Rainfall', 'RiverLevel']] # Features (rainfall and river level)
y = data['Flood']                    # Target variable (binary: 0 = no flood, 1 = flood)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model training
model = LinearRegression()
model.fit(X_train, y_train)

# Model evaluation
y_pred = model.predict(X_test)

# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

# Visualize predictions
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Flood Status")
plt.ylabel("Predicted Flood Probability")
plt.title("Actual vs. Predicted Flood Status")
plt.show()

# Make predictions for new data (replace with your own input data)
new_data = pd.DataFrame({'Rainfall': [50.0], 'RiverLevel': [3.0]})
prediction = model.predict(new_data)
if prediction[0] > 0.5:
    print("Flood is predicted.")
else:
    print("No flood is predicted.")
```

Prediction Algorithms

1.KNN Classifier

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

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

In[]:#Actual chances of flood
     print("actual values of floods:")
     print(y_test)
     from sklearn.model_selection
               import cross_val_score
In[]:knn_accuracy =cross
_ val_score (knn_clf,x_test,y_test,cv
=3,scoring='accuracy',n_jobs=-1)

In[]:knn_accuracy.mean()

Out[]:0.7083333333333334
```

2.Logistic Regression

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

In []: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[]:lr_accuracy.mean()

Out[]:0.625
```

In[]: y_predict = lr_clf.predict(x_test_std)
print('Predicted chances of flood')
print(y_predict)
Predicted chances of flood
[1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 1 0]
In[]: print('Actual chances of flood')
print(y_test.values)
In[]: Actual chances of flood
[1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0]
In[]: 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))
Out[]: accuracy score: 83.333333
recall score: 88.888889
roc score: 84.444444

3.Decision Tree Classification

In[]: 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[]: array([0.71875 , 0.64516129, 0.61290323**])**
In[]: #Predicted flood chances
y_pred = dtc_clf.predict(x_test)
print(y_pred)
[1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0]

In []:#Actual flood chances

```
print("actual  values:")
print(y_test.values)
```

actual values:

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

In[]: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:70.833333

recall score:77.777778

roc score:72.222222

4.Random Forest Classification:

In[]:from sklearn.ensemble import RandomForestClassifier

```
rmf = RandomForestClassifier(max_depth=3,random_state=0)
rmf_clf = rmf.fit(x_train,y_train)
rmf_clf
```

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

```
rmf_clf_acc =
```

cross_val_score(rmf_clf,x_train_std,y_train,cv=3,scoring="accuracy",n_jobs=-1)

In[]:#rmf_proba =

cross_val_predict(rmf_clf,x_train_std,y_train,cv=3,method='predict_proba')

In[]:rmf_clf_acc

Out[]:array([0.8125 , 0.67741935, 0.87096774])

In[]:y_pred = rmf_clf.predict(x_test)

In[]: 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:100.000000

roc score:83.333333

5.Ensemble Learning

In[]: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[]:voting_clf = voting.fit(x_train, y_train)
from sklearn.metrics import accuracy_score

```
In[]: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 0.9583333333333334
RandomForestClassifier 0.7083333333333334
KNeighborsClassifier 0.9166666666666666
VotingClassifier 0.9166666666666666

Comparing All Prediction Models

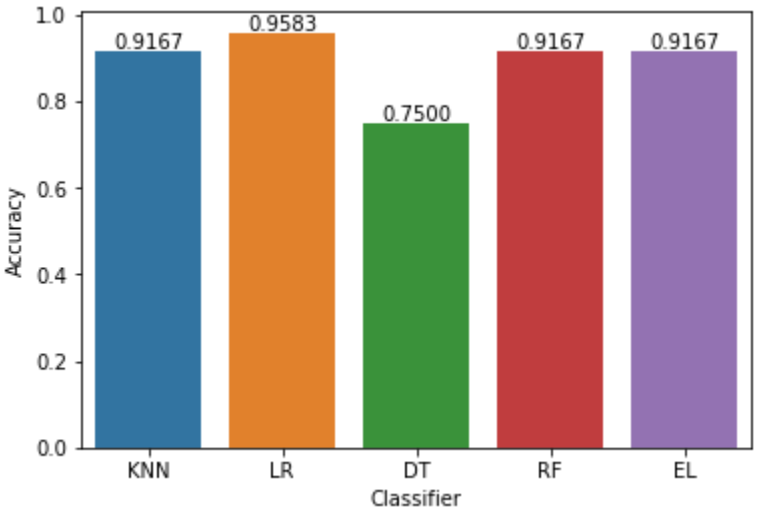
In[]:
```
models = []
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
models.append(('KNN', KNeighborsClassifier()))
models.append(('LR', LogisticRegression()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier()))
models.append(('EL', 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  0.916667
1   LR  0.958333
2   DT  0.750000
3   RF  0.916667
4   EL  0.916667
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:764:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

In[]: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()

Output:



Conclusion and future Work(Phase 2):

Project Conclusion:

- Flood monitoring systems are indispensable tools for mitigating the devastating impacts of floods.
- By continuously collecting and analyzing data, issuing early warnings, and facilitating coordinated responses, these systems play a crucial role in safeguarding lives and property.
- Future Work: Future work in flood monitoring systems should aim to provide more accurate and timely information, improve the resilience of communities and infrastructure, and enhance our ability to mitigate the impact of floods in an era of changing climate patterns.