In [2]:
```
pip install pandas numpy matplotlib seaborn scikit-learn xgboost joblib
```

```
Requirement already satisfied: pandas in c:\users\admin\anaconda3\lib\site-packages
(2.2.2)
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-packages
(1.26.4)
Requirement already satisfied: matplotlib in c:\users\admin\anaconda3\lib\site-packa
ges (3.9.2)
Requirement already satisfied: seaborn in c:\users\admin\anaconda3\lib\site-packages
(0.13.2)
Requirement already satisfied: scikit-learn in c:\users\admin\anaconda3\lib\site-pac
kages (1.5.1)
Collecting xgboost
  Downloading xgboost-3.0.2-py3-none-win_amd64.whl.metadata (2.1 kB)
Requirement already satisfied: joblib in c:\users\admin\anaconda3\lib\site-packages
(1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\anaconda3\li
b\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\anaconda3\lib\site-pac
kages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\anaconda3\lib\site-p
ackages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\anaconda3\lib\site
-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\anaconda3\lib\site-pac
kages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\anaconda3\lib\sit
e-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\admin\anaconda3\lib\sit
e-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\anaconda3\lib\site-
packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\admin\anaconda3\lib\site-packag
es (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\anaconda3\lib\site
-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\admin\anaconda3\lib\site-pac
kages (from scikit-learn) (1.13.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\admin\anaconda3\lib
\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-package
s (from python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading xgboost-3.0.2-py3-none-win_amd64.whl (150.0 MB)
   ---------------------------------------- 0.0/150.0 MB ? eta -:--:--
   ---------------------------------------- 0.0/150.0 MB ? eta -:--:--
   ---------------------------------------- 0.3/150.0 MB ? eta -:--:--
   ---------------------------------------- 0.8/150.0 MB 1.5 MB/s eta 0:01:43
   ---------------------------------------- 1.0/150.0 MB 1.5 MB/s eta 0:01:41
   ---------------------------------------- 1.3/150.0 MB 1.6 MB/s eta 0:01:34
   ---------------------------------------- 1.8/150.0 MB 1.7 MB/s eta 0:01:30
    --------------------------------------- 2.4/150.0 MB 1.8 MB/s eta 0:01:22
    --------------------------------------- 2.9/150.0 MB 1.9 MB/s eta 0:01:18
    --------------------------------------- 3.4/150.0 MB 2.0 MB/s eta 0:01:13
   - -------------------------------------- 4.2/150.0 MB 2.1 MB/s eta 0:01:09
   - -------------------------------------- 4.7/150.0 MB 2.2 MB/s eta 0:01:07
   - -------------------------------------- 5.5/150.0 MB 2.3 MB/s eta 0:01:03
   - -------------------------------------- 6.0/150.0 MB 2.4 MB/s eta 0:01:02
   - -------------------------------------- 6.8/150.0 MB 2.4 MB/s eta 0:01:00
```

```
- ------------------------------------ 7.3/150.0 MB 2.4 MB/s eta 0:00:59
-- ------------------------------------ 7.9/150.0 MB 2.4 MB/s eta 0:00:59
-- ------------------------------------ 8.7/150.0 MB 2.5 MB/s eta 0:00:57
-- ------------------------------------ 9.2/150.0 MB 2.5 MB/s eta 0:00:57
-- ------------------------------------ 9.7/150.0 MB 2.5 MB/s eta 0:00:56
-- ------------------------------------ 10.5/150.0 MB 2.5 MB/s eta 0:00:55
-- ------------------------------------ 11.0/150.0 MB 2.6 MB/s eta 0:00:55
--- ----------------------------------- 11.8/150.0 MB 2.6 MB/s eta 0:00:54
--- ----------------------------------- 12.1/150.0 MB 2.6 MB/s eta 0:00:53
--- ----------------------------------- 12.6/150.0 MB 2.5 MB/s eta 0:00:55
--- ----------------------------------- 13.1/150.0 MB 2.6 MB/s eta 0:00:54
--- ----------------------------------- 13.9/150.0 MB 2.6 MB/s eta 0:00:53
--- ----------------------------------- 14.4/150.0 MB 2.6 MB/s eta 0:00:53
---- ---------------------------------- 15.2/150.0 MB 2.6 MB/s eta 0:00:52
---- ---------------------------------- 16.0/150.0 MB 2.7 MB/s eta 0:00:51
---- ---------------------------------- 16.5/150.0 MB 2.6 MB/s eta 0:00:51
---- ---------------------------------- 17.3/150.0 MB 2.7 MB/s eta 0:00:50
---- ---------------------------------- 17.6/150.0 MB 2.7 MB/s eta 0:00:50
---- ---------------------------------- 18.4/150.0 MB 2.7 MB/s eta 0:00:50
----- --------------------------------- 18.9/150.0 MB 2.7 MB/s eta 0:00:50
----- --------------------------------- 19.7/150.0 MB 2.7 MB/s eta 0:00:49
----- --------------------------------- 20.4/150.0 MB 2.7 MB/s eta 0:00:48
----- --------------------------------- 21.2/150.0 MB 2.8 MB/s eta 0:00:47
----- --------------------------------- 22.3/150.0 MB 2.8 MB/s eta 0:00:46
------ -------------------------------- 23.1/150.0 MB 2.8 MB/s eta 0:00:45
------ -------------------------------- 23.9/150.0 MB 2.9 MB/s eta 0:00:45
------ -------------------------------- 24.9/150.0 MB 2.9 MB/s eta 0:00:44
------ -------------------------------- 25.7/150.0 MB 2.9 MB/s eta 0:00:43
------- ------------------------------- 26.7/150.0 MB 3.0 MB/s eta 0:00:42
------- ------------------------------- 27.8/150.0 MB 3.0 MB/s eta 0:00:41
------- ------------------------------- 28.8/150.0 MB 3.1 MB/s eta 0:00:40
------- ------------------------------- 29.9/150.0 MB 3.1 MB/s eta 0:00:39
-------- ------------------------------ 30.9/150.0 MB 3.1 MB/s eta 0:00:38
-------- ------------------------------ 31.7/150.0 MB 3.2 MB/s eta 0:00:38
-------- ------------------------------ 32.5/150.0 MB 3.2 MB/s eta 0:00:38
--------- ----------------------------- 33.8/150.0 MB 3.2 MB/s eta 0:00:37
--------- ----------------------------- 34.6/150.0 MB 3.2 MB/s eta 0:00:36
--------- ----------------------------- 35.9/150.0 MB 3.3 MB/s eta 0:00:35
--------- ----------------------------- 37.2/150.0 MB 3.4 MB/s eta 0:00:34
---------- ---------------------------- 38.5/150.0 MB 3.4 MB/s eta 0:00:33
---------- ---------------------------- 40.1/150.0 MB 3.5 MB/s eta 0:00:32
----------- ---------------------------- 41.4/150.0 MB 3.5 MB/s eta 0:00:31
----------- ---------------------------- 42.7/150.0 MB 3.6 MB/s eta 0:00:31
----------- ---------------------------- 44.0/150.0 MB 3.6 MB/s eta 0:00:30
------------ --------------------------- 45.4/150.0 MB 3.7 MB/s eta 0:00:29
------------ --------------------------- 46.7/150.0 MB 3.7 MB/s eta 0:00:28
------------ --------------------------- 48.2/150.0 MB 3.8 MB/s eta 0:00:28
------------- -------------------------- 49.8/150.0 MB 3.8 MB/s eta 0:00:27
------------- -------------------------- 51.4/150.0 MB 3.9 MB/s eta 0:00:26
-------------- ------------------------- 53.0/150.0 MB 3.9 MB/s eta 0:00:25
-------------- ------------------------- 54.5/150.0 MB 4.0 MB/s eta 0:00:24
-------------- ------------------------- 55.8/150.0 MB 4.0 MB/s eta 0:00:24
--------------- ------------------------ 56.9/150.0 MB 4.0 MB/s eta 0:00:24
--------------- ------------------------ 58.2/150.0 MB 4.1 MB/s eta 0:00:23
--------------- ------------------------ 59.0/150.0 MB 4.1 MB/s eta 0:00:23
---------------- ----------------------- 60.3/150.0 MB 4.1 MB/s eta 0:00:22
```

```
--------------- ---------------------- 61.1/150.0 MB 4.1 MB/s eta 0:00:22
--------------- ---------------------- 62.1/150.0 MB 4.1 MB/s eta 0:00:22
--------------- ---------------------- 63.4/150.0 MB 4.1 MB/s eta 0:00:21
---------------- ---------------------- 64.2/150.0 MB 4.1 MB/s eta 0:00:21
---------------- ---------------------- 65.3/150.0 MB 4.1 MB/s eta 0:00:21
---------------- ---------------------- 66.6/150.0 MB 4.2 MB/s eta 0:00:21
----------------- --------------------- 67.6/150.0 MB 4.2 MB/s eta 0:00:20
----------------- --------------------- 68.9/150.0 MB 4.2 MB/s eta 0:00:20
----------------- --------------------- 70.0/150.0 MB 4.2 MB/s eta 0:00:20
----------------- --------------------- 71.0/150.0 MB 4.2 MB/s eta 0:00:19
------------------ -------------------- 72.4/150.0 MB 4.2 MB/s eta 0:00:19
------------------ -------------------- 73.7/150.0 MB 4.3 MB/s eta 0:00:18
------------------ -------------------- 74.7/150.0 MB 4.3 MB/s eta 0:00:18
------------------- ------------------- 76.0/150.0 MB 4.3 MB/s eta 0:00:18
------------------- ------------------- 77.3/150.0 MB 4.3 MB/s eta 0:00:17
------------------- ------------------- 78.6/150.0 MB 4.3 MB/s eta 0:00:17
-------------------- ------------------ 80.0/150.0 MB 4.3 MB/s eta 0:00:17
-------------------- ------------------ 81.0/150.0 MB 4.4 MB/s eta 0:00:16
-------------------- ------------------ 82.3/150.0 MB 4.4 MB/s eta 0:00:16
--------------------- ----------------- 83.6/150.0 MB 4.4 MB/s eta 0:00:16
--------------------- ----------------- 84.7/150.0 MB 4.4 MB/s eta 0:00:15
--------------------- ----------------- 85.5/150.0 MB 4.4 MB/s eta 0:00:15
---------------------- ---------------- 86.2/150.0 MB 4.4 MB/s eta 0:00:15
---------------------- ---------------- 87.3/150.0 MB 4.4 MB/s eta 0:00:15
---------------------- ---------------- 87.8/150.0 MB 4.4 MB/s eta 0:00:15
---------------------- ---------------- 88.3/150.0 MB 4.3 MB/s eta 0:00:15
---------------------- ---------------- 89.7/150.0 MB 4.4 MB/s eta 0:00:14
---------------------- ---------------- 89.7/150.0 MB 4.4 MB/s eta 0:00:14
---------------------- ---------------- 89.9/150.0 MB 4.3 MB/s eta 0:00:14
----------------------- --------------- 90.7/150.0 MB 4.3 MB/s eta 0:00:14
----------------------- --------------- 91.2/150.0 MB 4.3 MB/s eta 0:00:14
----------------------- --------------- 92.0/150.0 MB 4.3 MB/s eta 0:00:14
----------------------- --------------- 92.5/150.0 MB 4.2 MB/s eta 0:00:14
----------------------- --------------- 93.3/150.0 MB 4.2 MB/s eta 0:00:14
------------------------ -------------- 94.4/150.0 MB 4.2 MB/s eta 0:00:14
------------------------ -------------- 94.9/150.0 MB 4.2 MB/s eta 0:00:14
------------------------ -------------- 95.4/150.0 MB 4.2 MB/s eta 0:00:13
------------------------ -------------- 95.9/150.0 MB 4.2 MB/s eta 0:00:13
------------------------ -------------- 96.5/150.0 MB 4.2 MB/s eta 0:00:13
------------------------ -------------- 97.3/150.0 MB 4.2 MB/s eta 0:00:13
------------------------- ------------- 98.0/150.0 MB 4.2 MB/s eta 0:00:13
------------------------- ------------- 98.6/150.0 MB 4.2 MB/s eta 0:00:13
------------------------- ------------- 98.8/150.0 MB 4.1 MB/s eta 0:00:13
------------------------- ------------- 99.9/150.0 MB 4.1 MB/s eta 0:00:13
------------------------- ------------- 100.7/150.0 MB 4.1 MB/s eta 0:00:12
------------------------- ------------- 100.9/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 101.4/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 102.2/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 103.0/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 103.0/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 103.8/150.0 MB 4.1 MB/s eta 0:00:12
-------------------------- ------------ 104.1/150.0 MB 4.0 MB/s eta 0:00:12
-------------------------- ------------ 104.6/150.0 MB 4.0 MB/s eta 0:00:12
--------------------------- ----------- 105.1/150.0 MB 4.0 MB/s eta 0:00:12
--------------------------- ----------- 105.9/150.0 MB 4.0 MB/s eta 0:00:12
--------------------------- ----------- 106.4/150.0 MB 4.0 MB/s eta 0:00:11
```

```
--------------------------- ----------- 107.2/150.0 MB 4.0 MB/s eta 0:00:11
--------------------------- ----------- 108.3/150.0 MB 4.0 MB/s eta 0:00:11
--------------------------- ---------- 109.3/150.0 MB 4.0 MB/s eta 0:00:11
--------------------------- ---------- 109.8/150.0 MB 4.0 MB/s eta 0:00:11
--------------------------- ---------- 110.9/150.0 MB 4.0 MB/s eta 0:00:10
--------------------------- ---------- 111.7/150.0 MB 4.0 MB/s eta 0:00:10
--------------------------- ---------- 112.5/150.0 MB 4.0 MB/s eta 0:00:10
-------------------------- --------- 113.5/150.0 MB 4.0 MB/s eta 0:00:10
-------------------------- --------- 114.6/150.0 MB 4.0 MB/s eta 0:00:09
-------------------------- --------- 115.3/150.0 MB 4.0 MB/s eta 0:00:09
-------------------------- --------- 116.1/150.0 MB 4.0 MB/s eta 0:00:09
-------------------------- -------- 116.9/150.0 MB 4.0 MB/s eta 0:00:09
-------------------------- -------- 117.7/150.0 MB 4.0 MB/s eta 0:00:09
-------------------------- -------- 118.2/150.0 MB 4.0 MB/s eta 0:00:08
-------------------------- -------- 119.0/150.0 MB 4.0 MB/s eta 0:00:08
------------------------- ------- 120.3/150.0 MB 4.0 MB/s eta 0:00:08
------------------------- ------- 121.1/150.0 MB 4.0 MB/s eta 0:00:08
------------------------- ------- 121.6/150.0 MB 4.0 MB/s eta 0:00:08
------------------------- ------- 122.7/150.0 MB 4.0 MB/s eta 0:00:07
------------------------- ------- 123.5/150.0 MB 4.1 MB/s eta 0:00:07
------------------------- ------ 124.3/150.0 MB 4.1 MB/s eta 0:00:07
------------------------- ------ 125.6/150.0 MB 4.1 MB/s eta 0:00:06
------------------------- ------ 126.9/150.0 MB 4.1 MB/s eta 0:00:06
------------------------- ------ 127.1/150.0 MB 4.1 MB/s eta 0:00:06
------------------------- ----- 128.2/150.0 MB 4.1 MB/s eta 0:00:06
------------------------- ----- 129.2/150.0 MB 4.1 MB/s eta 0:00:06
------------------------- ----- 130.0/150.0 MB 4.1 MB/s eta 0:00:05
------------------------- ----- 130.5/150.0 MB 4.1 MB/s eta 0:00:05
------------------------- ---- 131.6/150.0 MB 4.1 MB/s eta 0:00:05
------------------------- ---- 132.4/150.0 MB 4.2 MB/s eta 0:00:05
------------------------- ---- 132.9/150.0 MB 4.2 MB/s eta 0:00:05
------------------------- ---- 133.4/150.0 MB 4.2 MB/s eta 0:00:04
------------------------- ---- 134.2/150.0 MB 4.2 MB/s eta 0:00:04
------------------------- --- 135.0/150.0 MB 4.2 MB/s eta 0:00:04
------------------------- --- 135.5/150.0 MB 4.2 MB/s eta 0:00:04
------------------------- --- 136.8/150.0 MB 4.2 MB/s eta 0:00:04
------------------------- --- 137.9/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- --- 138.7/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 139.7/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 140.5/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 141.0/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 141.3/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 141.6/150.0 MB 4.2 MB/s eta 0:00:03
------------------------- -- 142.3/150.0 MB 4.2 MB/s eta 0:00:02
------------------------- - 143.1/150.0 MB 4.2 MB/s eta 0:00:02
------------------------- - 143.7/150.0 MB 4.2 MB/s eta 0:00:02
------------------------- - 144.7/150.0 MB 4.2 MB/s eta 0:00:02
------------------------- - 145.5/150.0 MB 4.2 MB/s eta 0:00:02
------------------------  146.8/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  147.3/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  148.4/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  149.2/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  149.9/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  149.9/150.0 MB 4.2 MB/s eta 0:00:01
------------------------  149.9/150.0 MB 4.2 MB/s eta 0:00:01
------------------------ 150.0/150.0 MB 4.2 MB/s eta 0:00:00
```

```
Installing collected packages: xgboost
Successfully installed xgboost-3.0.2
Note: you may need to restart the kernel to use updated packages.
```

In [11]:
```python
# Part 1: Imports and Setup
# Flood Prediction ML System with Parallel Computing

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.pipeline import Pipeline
import xgboost as xgb
import time
import warnings
warnings.filterwarnings('ignore')

# For parallel processing
from joblib import Parallel, delayed
import multiprocessing

print("="*60)
print("FLOOD PREDICTION SYSTEM - DISASTER RESPONSE ML")
print("="*60)
print("Part 1: All imports loaded successfully!")
print(f"Available CPU cores: {multiprocessing.cpu_count()}")
```

```
============================================================
FLOOD PREDICTION SYSTEM - DISASTER RESPONSE ML
============================================================
Part 1: All imports loaded successfully!
Available CPU cores: 12
```

In [12]:
```python
# Part 2: Data Loading and Exploration
# Run this after Part 1

def load_and_explore_data():
    """Load and explore the flood prediction dataset"""
    print("\n2. LOADING AND EXPLORING DATASET")
    print("-" * 40)

    # Load dataset
    df = pd.read_csv('flood.csv')

    print(f"Dataset shape: {df.shape}")
    print(f"Dataset columns: {list(df.columns)}")
    print(f"\nFirst few rows:")
    print(df.head())

    print(f"\nDataset info:")
    print(df.info())
```

```
    print(f"\nMissing values:")
    print(df.isnull().sum())

    print(f"\nStatistical summary:")
    print(df.describe())

    return df

# Run the data loading function
df = load_and_explore_data()
print("\nPart 2: Data loading completed successfully!")
```

```
2. LOADING AND EXPLORING DATASET
----------------------------------------
Dataset shape: (50000, 21)
Dataset columns: ['MonsoonIntensity', 'TopographyDrainage', 'RiverManagement', 'Defo
restation', 'Urbanization', 'ClimateChange', 'DamsQuality', 'Siltation', 'Agricultur
alPractices', 'Encroachments', 'IneffectiveDisasterPreparedness', 'DrainageSystems',
'CoastalVulnerability', 'Landslides', 'Watersheds', 'DeterioratingInfrastructure',
'PopulationScore', 'WetlandLoss', 'InadequatePlanning', 'PoliticalFactors', 'FloodPr
obability']

First few rows:
   MonsoonIntensity  TopographyDrainage  RiverManagement  Deforestation  \
0                 3                   8                6              6
1                 8                   4                5              7
2                 3                  10                4              1
3                 4                   4                2              7
4                 3                   7                5              2

   Urbanization  ClimateChange  DamsQuality  Siltation  AgriculturalPractices  \
0             4              4            6          2                      3
1             7              9            1          5                      5
2             7              5            4          7                      4
3             3              4            1          4                      6
4             5              8            5          2                      7

   Encroachments  ...  DrainageSystems  CoastalVulnerability  Landslides  \
0              2  ...               10                     7           4
1              4  ...                9                     2           6
2              9  ...                7                     4           4
3              4  ...                4                     2           6
4              5  ...                7                     6           5

   Watersheds  DeterioratingInfrastructure  PopulationScore  WetlandLoss  \
0           2                            3                4            3
1           2                            1                1            9
2           8                            6                1            8
3           6                            8                8            6
4           3                            3                4            4

   InadequatePlanning  PoliticalFactors  FloodProbability
0                   2                 6             0.450
1                   1                 3             0.475
2                   3                 6             0.515
3                   6                10             0.520
4                   3                 4             0.475

[5 rows x 21 columns]

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   MonsoonIntensity                50000 non-null  int64
 1   TopographyDrainage              50000 non-null  int64
```

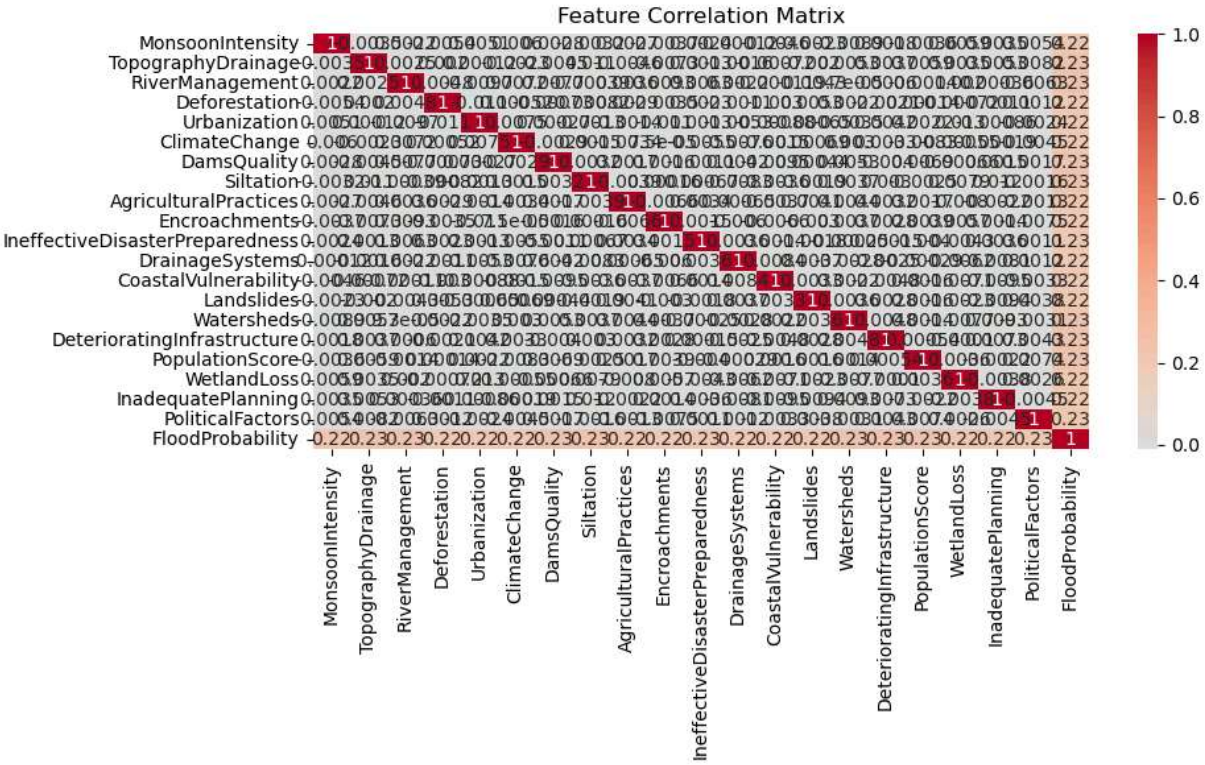
```
 2   RiverManagement                  50000 non-null  int64
 3   Deforestation                    50000 non-null  int64
 4   Urbanization                     50000 non-null  int64
 5   ClimateChange                    50000 non-null  int64
 6   DamsQuality                      50000 non-null  int64
 7   Siltation                        50000 non-null  int64
 8   AgriculturalPractices            50000 non-null  int64
 9   Encroachments                    50000 non-null  int64
 10  IneffectiveDisasterPreparedness  50000 non-null  int64
 11  DrainageSystems                  50000 non-null  int64
 12  CoastalVulnerability             50000 non-null  int64
 13  Landslides                       50000 non-null  int64
 14  Watersheds                       50000 non-null  int64
 15  DeterioratingInfrastructure      50000 non-null  int64
 16  PopulationScore                  50000 non-null  int64
 17  WetlandLoss                      50000 non-null  int64
 18  InadequatePlanning               50000 non-null  int64
 19  PoliticalFactors                 50000 non-null  int64
 20  FloodProbability                 50000 non-null  float64
dtypes: float64(1), int64(20)
memory usage: 8.0 MB
None

Missing values:
MonsoonIntensity                   0
TopographyDrainage                 0
RiverManagement                    0
Deforestation                      0
Urbanization                       0
ClimateChange                      0
DamsQuality                        0
Siltation                          0
AgriculturalPractices              0
Encroachments                      0
IneffectiveDisasterPreparedness    0
DrainageSystems                    0
CoastalVulnerability               0
Landslides                         0
Watersheds                         0
DeterioratingInfrastructure        0
PopulationScore                    0
WetlandLoss                        0
InadequatePlanning                 0
PoliticalFactors                   0
FloodProbability                   0
dtype: int64

Statistical summary:
       MonsoonIntensity  TopographyDrainage  RiverManagement  Deforestation  \
count      50000.000000        50000.000000      50000.00000   50000.000000
mean           4.991480            4.984100          5.01594       5.008480
std            2.236834            2.246488          2.23131       2.222743
min            0.000000            0.000000          0.00000       0.000000
25%            3.000000            3.000000          3.00000       3.000000
50%            5.000000            5.000000          5.00000       5.000000
75%            6.000000            6.000000          6.00000       6.000000
```

```
max           16.000000           18.000000         16.00000      17.000000

       Urbanization  ClimateChange  DamsQuality     Siltation  \
count  50000.000000   50000.000000  50000.00000  50000.000000
mean       4.989060       4.988340      5.01536      4.988600
std        2.243159       2.226761      2.24500      2.232642
min        0.000000       0.000000      0.00000      0.000000
25%        3.000000       3.000000      3.00000      3.000000
50%        5.000000       5.000000      5.00000      5.000000
75%        6.000000       6.000000      6.00000      6.000000
max       17.000000      17.000000     16.00000     16.000000

       AgriculturalPractices  Encroachments  ...  DrainageSystems  \
count           50000.000000   50000.000000  ...     50000.000000
mean                5.006120       5.006380  ...         5.006060
std                 2.234588       2.241633  ...         2.238107
min                 0.000000       0.000000  ...         0.000000
25%                 3.000000       3.000000  ...         3.000000
50%                 5.000000       5.000000  ...         5.000000
75%                 6.000000       6.000000  ...         6.000000
max                16.000000      18.000000  ...        17.000000

       CoastalVulnerability    Landslides   Watersheds  \
count          50000.000000  50000.000000  50000.00000
mean               4.999920      4.984220      4.97982
std                2.247101      2.227741      2.23219
min                0.000000      0.000000      0.00000
25%                3.000000      3.000000      3.00000
50%                5.000000      5.000000      5.00000
75%                6.000000      6.000000      6.00000
max               17.000000     16.000000     16.00000

       DeterioratingInfrastructure  PopulationScore  WetlandLoss  \
count                 50000.000000     50000.000000  50000.00000
mean                      4.988200         4.984980      5.00512
std                       2.231134         2.238279      2.23176
min                       0.000000         0.000000      0.00000
25%                       3.000000         3.000000      3.00000
50%                       5.000000         5.000000      5.00000
75%                       6.000000         6.000000      6.00000
max                      17.000000        19.000000     22.00000

       InadequatePlanning  PoliticalFactors  FloodProbability
count        50000.000000      50000.000000      50000.000000
mean             4.994360          4.990520          0.499660
std              2.230011          2.246075          0.050034
min              0.000000          0.000000          0.285000
25%              3.000000          3.000000          0.465000
50%              5.000000          5.000000          0.500000
75%              6.000000          6.000000          0.535000
max             16.000000         16.000000          0.725000

[8 rows x 21 columns]

Part 2: Data loading completed successfully!
```

In [13]:

```
# Part 3: Data Preprocessing
# Run this after Part 2

def preprocess_data(df):
    """Preprocess the flood prediction data"""
    print("\n3. DATA PREPROCESSING")
    print("-" * 40)

    # Handle missing values if any
    df = df.dropna()

    # Identify target variable (usually named 'flood' or similar)
    target_col = None
    for col in df.columns:
        if 'flood' in col.lower() or 'target' in col.lower():
            target_col = col
            break

    if target_col is None:
        # If no obvious target, assume last column is target
        target_col = df.columns[-1]

    print(f"Target variable: {target_col}")

    # Separate features and target
    X = df.drop(columns=[target_col])
    y = df[target_col]

    print(f"Features shape: {X.shape}")
    print(f"Target shape: {y.shape}")

    # Display feature importance through correlation
    plt.figure(figsize=(10, 6))
    correlation_matrix = df.corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
    plt.title('Feature Correlation Matrix')
    plt.tight_layout()
    plt.show()

    return X, y

# Run preprocessing
X, y = preprocess_data(df)

# Split data for training and testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"\nTraining set size: {X_train.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")
print("Part 3: Data preprocessing completed successfully!")
```

```
3. DATA PREPROCESSING
----------------------------------------
Target variable: FloodProbability
Features shape: (50000, 20)
Target shape: (50000,)
```




```
Training set size: 40000
Test set size: 10000
Part 3: Data preprocessing completed successfully!
```

In [14]:
```python
# Part 4: Sequential Model Training
# Run this after Part 3

def train_model_sequential(X_train, y_train):
    """Train model sequentially (non-parallel)"""
    print("\n4. SEQUENTIAL MODEL TRAINING")
    print("-" * 40)

    start_time = time.time()

    # Create pipeline with preprocessing and model
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('model', RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=1
    ])

    # Train model
    pipeline.fit(X_train, y_train)

    end_time = time.time()
    training_time = end_time - start_time

    print(f"Sequential training completed in {training_time:.2f} seconds")
```

```
    return pipeline, training_time

# Train sequential model
seq_model, seq_time = train_model_sequential(X_train, y_train)
print("Part 4: Sequential model training completed successfully!")
```

```
4. SEQUENTIAL MODEL TRAINING
----------------------------------------
Sequential training completed in 34.10 seconds
Part 4: Sequential model training completed successfully!
```

In [15]:
```
# Part 5: Parallel Model Training
# Run this after Part 4

def train_model_parallel(X_train, y_train):
    """Train model with parallel processing"""
    print("\n5. PARALLEL MODEL TRAINING")
    print("-" * 40)

    start_time = time.time()

    # Get number of CPU cores
    n_cores = multiprocessing.cpu_count()
    print(f"Using {n_cores} CPU cores for parallel processing")

    # Create pipeline with parallel processing
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('model', RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-
    ])

    # Train model with parallel processing
    pipeline.fit(X_train, y_train)

    end_time = time.time()
    training_time = end_time - start_time

    print(f"Parallel training completed in {training_time:.2f} seconds")

    return pipeline, training_time

# Train parallel model
par_model, par_time = train_model_parallel(X_train, y_train)
print("Part 5: Parallel model training completed successfully!")
```

```
5. PARALLEL MODEL TRAINING
----------------------------------------
Using 12 CPU cores for parallel processing
Parallel training completed in 6.40 seconds
Part 5: Parallel model training completed successfully!
```

In [16]:
```
# Part 6: XGBoost Parallel Training
# Run this after Part 5

def train_xgboost_parallel(X_train, y_train):
    """Train XGBoost model with parallel processing"""
    print("\n6. XGBOOST PARALLEL TRAINING")
```

```
    print("-" * 40)

    start_time = time.time()

    # Preprocess data
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)

    # Train XGBoost with parallel processing
    model = xgb.XGBRegressor(
        n_estimators=100,
        random_state=42,
        n_jobs=-1,  # Use all available cores
        tree_method='auto'
    )

    model.fit(X_train_scaled, y_train)

    end_time = time.time()
    training_time = end_time - start_time

    print(f"XGBoost parallel training completed in {training_time:.2f} seconds")

    return model, scaler, training_time

# Train XGBoost model
xgb_model, xgb_scaler, xgb_time = train_xgboost_parallel(X_train, y_train)
print("Part 6: XGBoost parallel training completed successfully!")
```

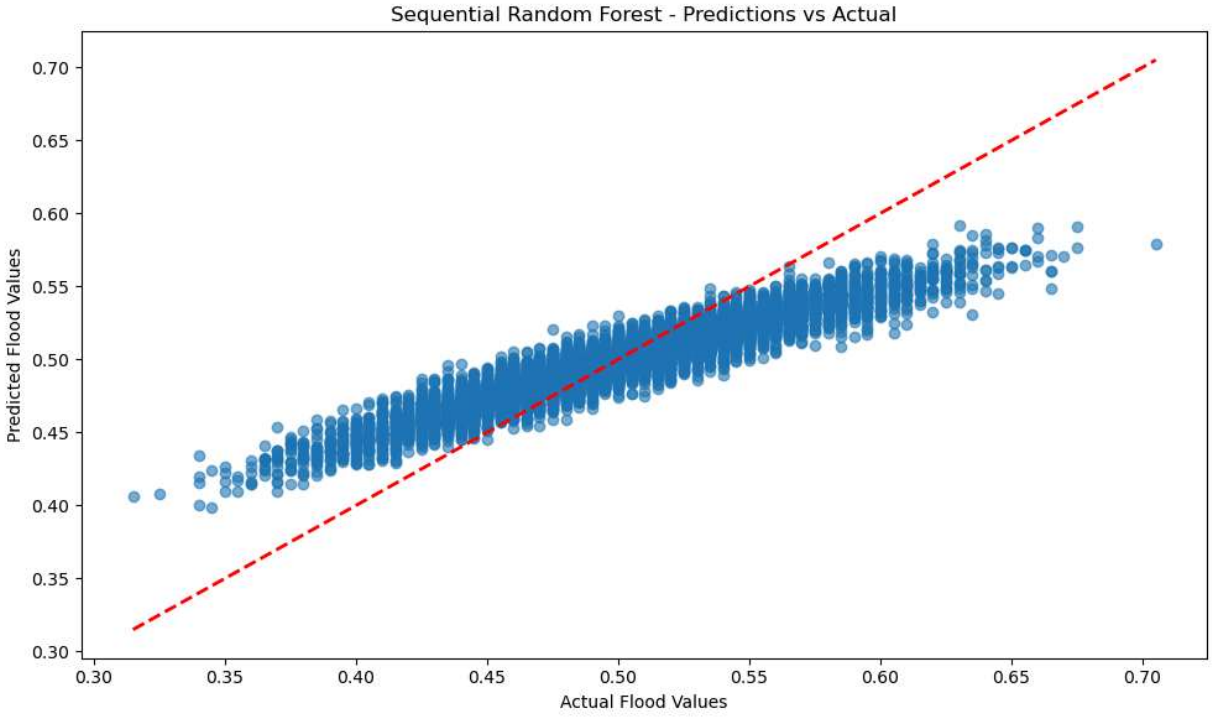
```
6. XGBOOST PARALLEL TRAINING
----------------------------------------
XGBoost parallel training completed in 0.46 seconds
Part 6: XGBoost parallel training completed successfully!
```

In [17]:

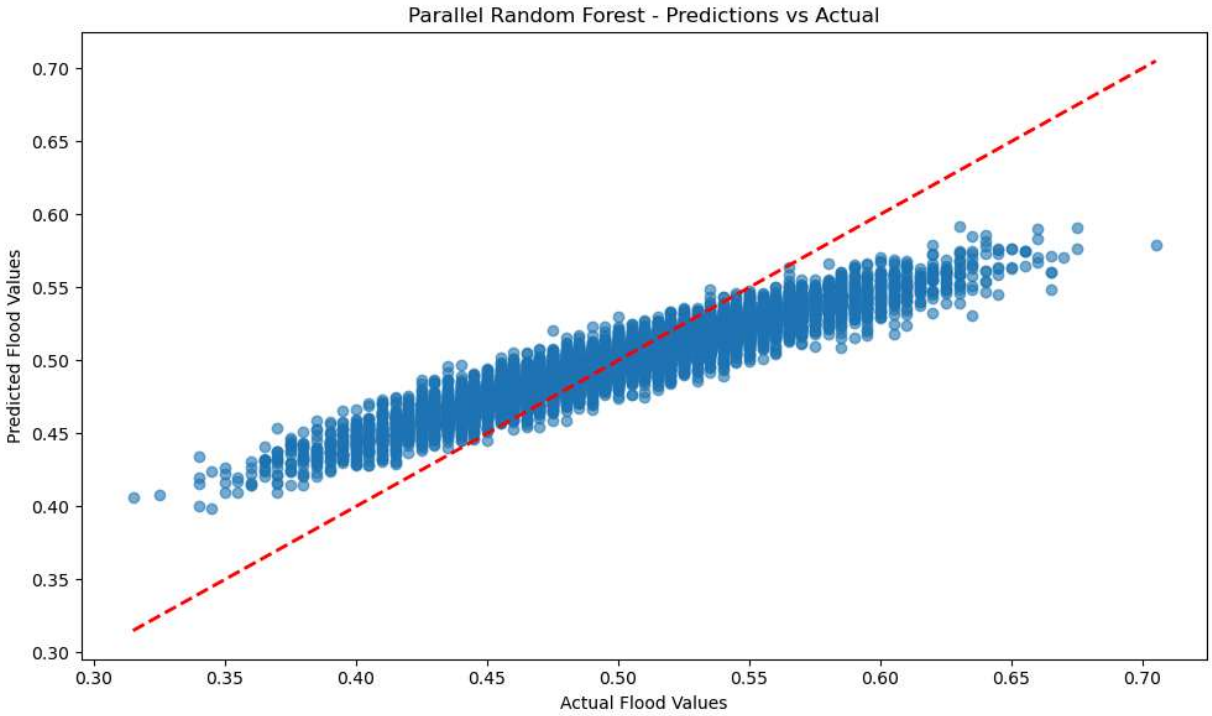
```
# Part 7: Model Evaluation
# Run this after Part 6

def evaluate_model(model, X_test, y_test, model_name, scaler=None):
    """Evaluate model performance"""
    print(f"\n7. MODEL EVALUATION - {model_name}")
    print("-" * 40)

    # Make predictions
    if scaler is not None:
        X_test_scaled = scaler.transform(X_test)
        y_pred = model.predict(X_test_scaled)
    else:
        y_pred = model.predict(X_test)

    # Calculate metrics
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error (MSE): {mse:.4f}")
```

```
    print(f"Mean Absolute Error (MAE): {mae:.4f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
    print(f"R² Score: {r2:.4f}")

    # Plot predictions vs actual
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, alpha=0.6)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=
    plt.xlabel('Actual Flood Values')
    plt.ylabel('Predicted Flood Values')
    plt.title(f'{model_name} - Predictions vs Actual')
    plt.tight_layout()
    plt.show()

    return {'MSE': mse, 'MAE': mae, 'RMSE': rmse, 'R2': r2}

# Evaluate all models
seq_results = evaluate_model(seq_model, X_test, y_test, "Sequential Random Forest")
par_results = evaluate_model(par_model, X_test, y_test, "Parallel Random Forest")
xgb_results = evaluate_model(xgb_model, X_test, y_test, "XGBoost Parallel", xgb_sca

print("Part 7: Model evaluation completed successfully!")
```

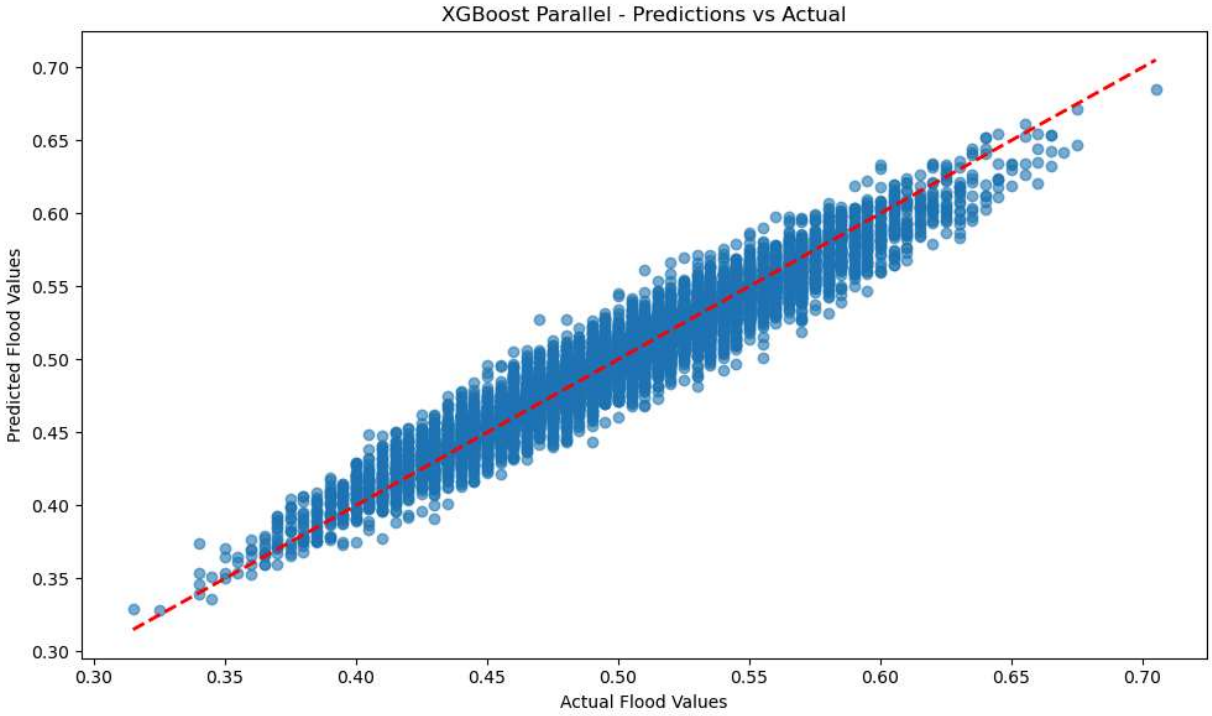
```
7. MODEL EVALUATION - Sequential Random Forest
----------------------------------------
Mean Squared Error (MSE): 0.0007
Mean Absolute Error (MAE): 0.0205
Root Mean Squared Error (RMSE): 0.0259
R² Score: 0.7305
```




```
7. MODEL EVALUATION - Parallel Random Forest
----------------------------------------
Mean Squared Error (MSE): 0.0007
Mean Absolute Error (MAE): 0.0205
Root Mean Squared Error (RMSE): 0.0259
R² Score: 0.7305
```




```
7. MODEL EVALUATION - XGBoost Parallel
----------------------------------------
Mean Squared Error (MSE): 0.0002
Mean Absolute Error (MAE): 0.0108
Root Mean Squared Error (RMSE): 0.0136
R² Score: 0.9252
```




```
Part 7: Model evaluation completed successfully!
```

In [18]:
```python
# Part 8: Parallel Cross-Validation
# Run this after Part 7

def parallel_cross_validation(X, y):
    """Perform parallel cross-validation"""
```

```
    print("\n8. PARALLEL CROSS-VALIDATION")
    print("-" * 40)

    # Create model
    model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)

    # Perform parallel cross-validation
    start_time = time.time()
    cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error'
    end_time = time.time()

    cv_time = end_time - start_time
    cv_rmse = np.sqrt(-cv_scores)

    print(f"Cross-validation completed in {cv_time:.2f} seconds")
    print(f"CV RMSE scores: {cv_rmse}")
    print(f"Mean CV RMSE: {cv_rmse.mean():.4f} (+/- {cv_rmse.std() * 2:.4f})")

    return cv_rmse.mean(), cv_time

# Perform cross-validation
cv_score, cv_time = parallel_cross_validation(X, y)
print("Part 8: Parallel cross-validation completed successfully!")
```

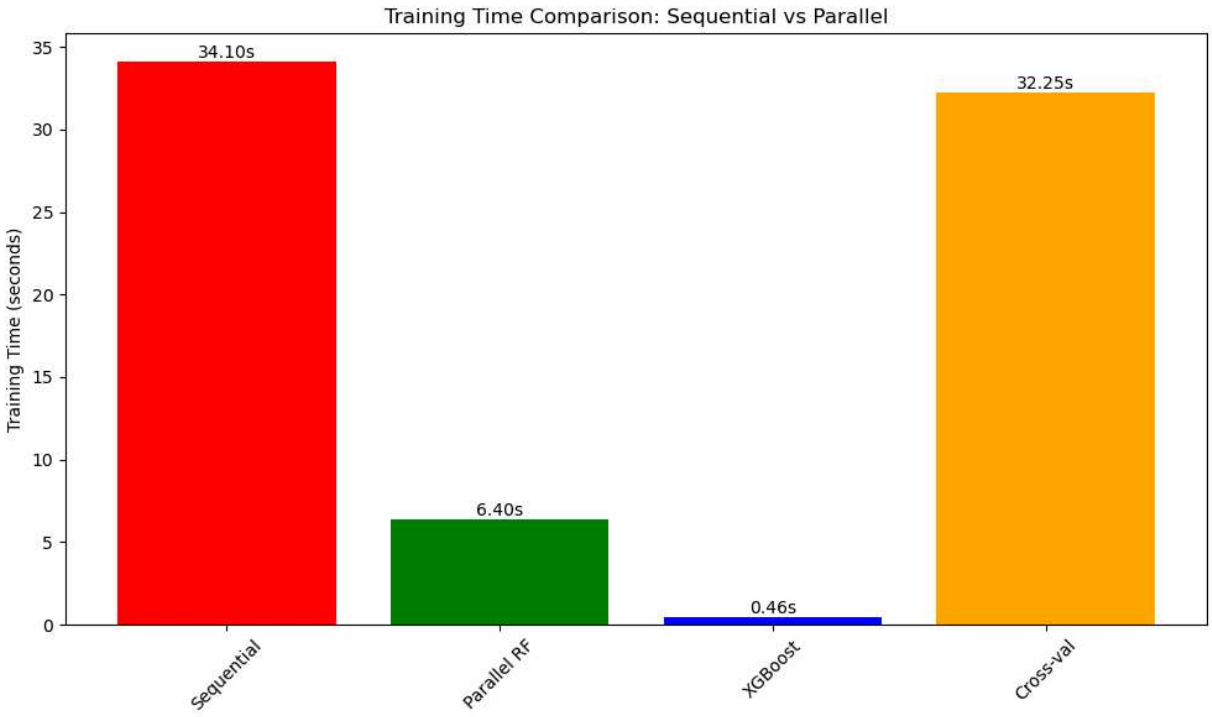
```
8. PARALLEL CROSS-VALIDATION
----------------------------------------
Cross-validation completed in 32.25 seconds
CV RMSE scores: [0.02619844 0.02542612 0.02568994 0.02604386 0.02595539]
Mean CV RMSE: 0.0259 (+/- 0.0005)
Part 8: Parallel cross-validation completed successfully!
```

In [19]:

```
# Part 9: Performance Comparison
# Run this after Part 8

def compare_performance(seq_time, par_time, xgb_time, cv_time):
    """Compare performance of different approaches"""
    print("\n9. PERFORMANCE COMPARISON")
    print("-" * 40)

    print(f"Sequential Training Time: {seq_time:.2f} seconds")
    print(f"Parallel Training Time: {par_time:.2f} seconds")
    print(f"XGBoost Parallel Time: {xgb_time:.2f} seconds")
    print(f"Cross-validation Time: {cv_time:.2f} seconds")

    speedup = seq_time / par_time
    print(f"\nParallel Speedup: {speedup:.2f}x")

    # Plot comparison
    methods = ['Sequential', 'Parallel RF', 'XGBoost', 'Cross-val']
    times = [seq_time, par_time, xgb_time, cv_time]

    plt.figure(figsize=(10, 6))
    bars = plt.bar(methods, times, color=['red', 'green', 'blue', 'orange'])
    plt.ylabel('Training Time (seconds)')
    plt.title('Training Time Comparison: Sequential vs Parallel')
    plt.xticks(rotation=45)
```

```
    # Add value labels on bars
    for bar, time_val in zip(bars, times):
        plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
                f'{time_val:.2f}s', ha='center', va='bottom')

    plt.tight_layout()
    plt.show()

# Compare performance
compare_performance(seq_time, par_time, xgb_time, cv_time)
print("Part 9: Performance comparison completed successfully!")
```

```
9. PERFORMANCE COMPARISON
----------------------------------------
Sequential Training Time: 34.10 seconds
Parallel Training Time: 6.40 seconds
XGBoost Parallel Time: 0.46 seconds
Cross-validation Time: 32.25 seconds

Parallel Speedup: 5.33x
```




```
Part 9: Performance comparison completed successfully!
```

In [20]:

```
# Part 10: Real-time Prediction
# Run this after Part 9

def real_time_prediction(model, scaler, sample_data):
    """Simulate real-time flood prediction"""
    print("\n10. REAL-TIME PREDICTION SIMULATION")
    print("-" * 40)

    # Simulate real-time prediction
    start_time = time.time()

    if scaler is not None:
```

```
        sample_scaled = scaler.transform(sample_data)
        prediction = model.predict(sample_scaled)
    else:
        prediction = model.predict(sample_data)

    end_time = time.time()
    prediction_time = end_time - start_time

    print(f"Real-time prediction: {prediction[0]:.4f}")
    print(f"Prediction latency: {prediction_time*1000:.2f} milliseconds")

    return prediction[0], prediction_time

# Real-time prediction simulation
sample_data = X_test.iloc[:1]  # Use first test sample
prediction, pred_time = real_time_prediction(par_model, None, sample_data)
print("Part 10: Real-time prediction completed successfully!")
```

```
10. REAL-TIME PREDICTION SIMULATION
----------------------------------------
Real-time prediction: 0.4643
Prediction latency: 173.69 milliseconds
Part 10: Real-time prediction completed successfully!
```

In [21]:

```
# Part 11: Final Summary
# Run this after Part 10

def print_final_summary():
    """Print final summary of all results"""
    print("\n" + "="*60)
    print("FLOOD PREDICTION SYSTEM - COMPLETED SUCCESSFULLY!")
    print("="*60)

    # Summary results
    print("\nSUMMARY RESULTS:")
    print("-" * 20)

    # Model performance comparison
    print("\nModel Performance (R² Score):")
    print(f"Sequential Random Forest: {seq_results['R2']:.4f}")
    print(f"Parallel Random Forest: {par_results['R2']:.4f}")
    print(f"XGBoost Parallel: {xgb_results['R2']:.4f}")

    # Find best model
    best_r2 = max(seq_results['R2'], par_results['R2'], xgb_results['R2'])
    print(f"\nBest Model R² Score: {best_r2:.4f}")

    # Training time comparison
    print("\nTraining Time Comparison:")
    print(f"Sequential Training: {seq_time:.2f} seconds")
    print(f"Parallel Training: {par_time:.2f} seconds")
    print(f"XGBoost Parallel: {xgb_time:.2f} seconds")
    print(f"Cross-validation: {cv_time:.2f} seconds")

    # Speedup calculation
    speedup = seq_time / par_time
```

```
    print(f"\nParallel Speedup: {speedup:.2f}x")

    # Real-time performance
    print(f"\nReal-time Prediction:")
    print(f"Sample prediction: {prediction:.4f}")
    print(f"Prediction latency: {pred_time*1000:.2f} milliseconds")

    # System specifications
    print(f"\nSystem Specifications:")
    print(f"CPU cores used: {multiprocessing.cpu_count()}")
    print(f"Dataset size: {X.shape[0]} samples, {X.shape[1]} features")
    print(f"Training set: {X_train.shape[0]} samples")
    print(f"Test set: {X_test.shape[0]} samples")

# Print final summary
print_final_summary()
```

```
============================================================
FLOOD PREDICTION SYSTEM - COMPLETED SUCCESSFULLY!
============================================================

SUMMARY RESULTS:
--------------------

Model Performance (R² Score):
Sequential Random Forest: 0.7305
Parallel Random Forest: 0.7305
XGBoost Parallel: 0.9252

Best Model R² Score: 0.9252

Training Time Comparison:
Sequential Training: 34.10 seconds
Parallel Training: 6.40 seconds
XGBoost Parallel: 0.46 seconds
Cross-validation: 32.25 seconds

Parallel Speedup: 5.33x

Real-time Prediction:
Sample prediction: 0.4643
Prediction latency: 173.69 milliseconds

System Specifications:
CPU cores used: 12
Dataset size: 50000 samples, 20 features
Training set: 40000 samples
Test set: 10000 samples
```

In [ ]: