COLLEGE CODE : 3126

COLLEGE NAME : THANGAVELU ENGINEERING COLLEGE

DEPARTMENT : B.TECH(INFORMATION TECHNOLOGY)

STUDENT NM-ID :089d7329118b7909c084fe92d019b2e

ROLL NO : 312623205006

DATE : 17/05/2025

Completed the project named as

TECHNOLOGY - PROJECT NAME :-

NATURAL DISASTER PREDICTION AND MANAGEMENT

SUBMITTED BY,

NAME : K.DHANUSH

MOBILE NO :6380104898

Phase 4: Performance of the project

Project Title: AI-Powered Natural Disaster Prediction & Management System

Objective:

To improve the system's performance in accurately predicting disasters, scaling under high usage, ensuring secure and real-time data processing, and preparing for multilingual emergency communication.

Procedure 1: AI Model Enhancement for Disaster Prediction

Step 1: Collect and analyze feedback from previous phases and identify misclassification or prediction delays.

Step 2: Expand the training dataset to include complex historical disaster data (e.g., multi-hazard events, regional patterns).

Step 3: Retrain the AI model using updated data with techniques such as transfer learning and ensemble methods.

Step 4: Apply hyperparameter tuning and model pruning to increase accuracy and reduce computation time.

Step 5: Evaluate prediction precision, false alarm rate, and response time.

Expected Output:

A more accurate and efficient AI model capable of early and reliable disaster forecasts.

Procedure 2: Communication Assistant (Chatbot) Optimization

Step 1: Analyze past user interactions to identify response lags or misinterpretations.

Step 2: Upgrade NLP components to improve understanding of diverse emergency inputs and regional language variations.

Step 3: Optimize backend response generation to ensure low-latency alerts and replies during peak traffic.

Step 4: Begin integration of multilingual support modules for key regional languages.

Step 5: Test the assistants performance under simulated high-stress scenarios.

Expected Output:

A faster, multilingual emergency assistant for real-time, accessible disaster communication.

Procedure 3: Real-Time IoT Integration for Disaster Monitoring

Step 1: Review sensor network compatibility (e.g., seismic, flood, weather stations).

Step 2: Optimize API calls and streaming protocols for real-time environmental data intake.

Step 3: Implement data preprocessing to clean and normalize incoming signals.

Step 4: Test and validate live data processing from multiple IoT sources (e.g., satellite feeds, drones, water sensors).

Step 5: Conduct latency and uptime analysis under continuous data flow.

Expected Output:

Seamless real-time integration with IoT systems for continuous disaster environment monitoring.

Procedure 4: Data Security and Privacy Reinforcement

Step 1: Implement advanced encryption (AES-256, TLS 1.3) for all data channels.

Step 2: Conduct system-wide penetration tests to identify vulnerabilities under high data loads.

Step 3: Perform stress testing to simulate high-user traffic during disaster events.

Step 4: Apply role-based access controls and audit logging for data access.

Step 5: Validate compliance with national and international disaster response data protocols.

Expected Output:

A robust and secure data infrastructure that maintains integrity under emergency conditions.

Procedure 5: System-Wide Performance Testing & FeedbacK Collection

Step 1: Simulate various disaster scenarios with different user volumes and data inputs.

Step 2: Monitor performance metrics including prediction speed, system stability, data throughput, and alert accuracy.

Step 3: Identify and resolve performance bottlenecKs using system diagnostics.

Step 4: Collect feedbacK from emergency responders and test users on system usability.

Step 5: Finalize readiness checKlist for pilot deployment.

Expected Output:

An optimized, stable, and user-tested system prepared for real-world deployment.

Key Performance Challenges

1.High user traffic during disasters:

Load balancing and cloud scalability testing

2.Data security during crises:

End-to-end encryption and role-based access control

3.IoT device diversity:

Use of standardized APIs and device testing

Phase Outcome:

By the end of Phase 4, the system will demonstrate:

Accurate and early disaster predictions

Real-time data integration from diverse sources

Multilingual, accessible emergency communication

Scalable and secure operation under emergency conditions

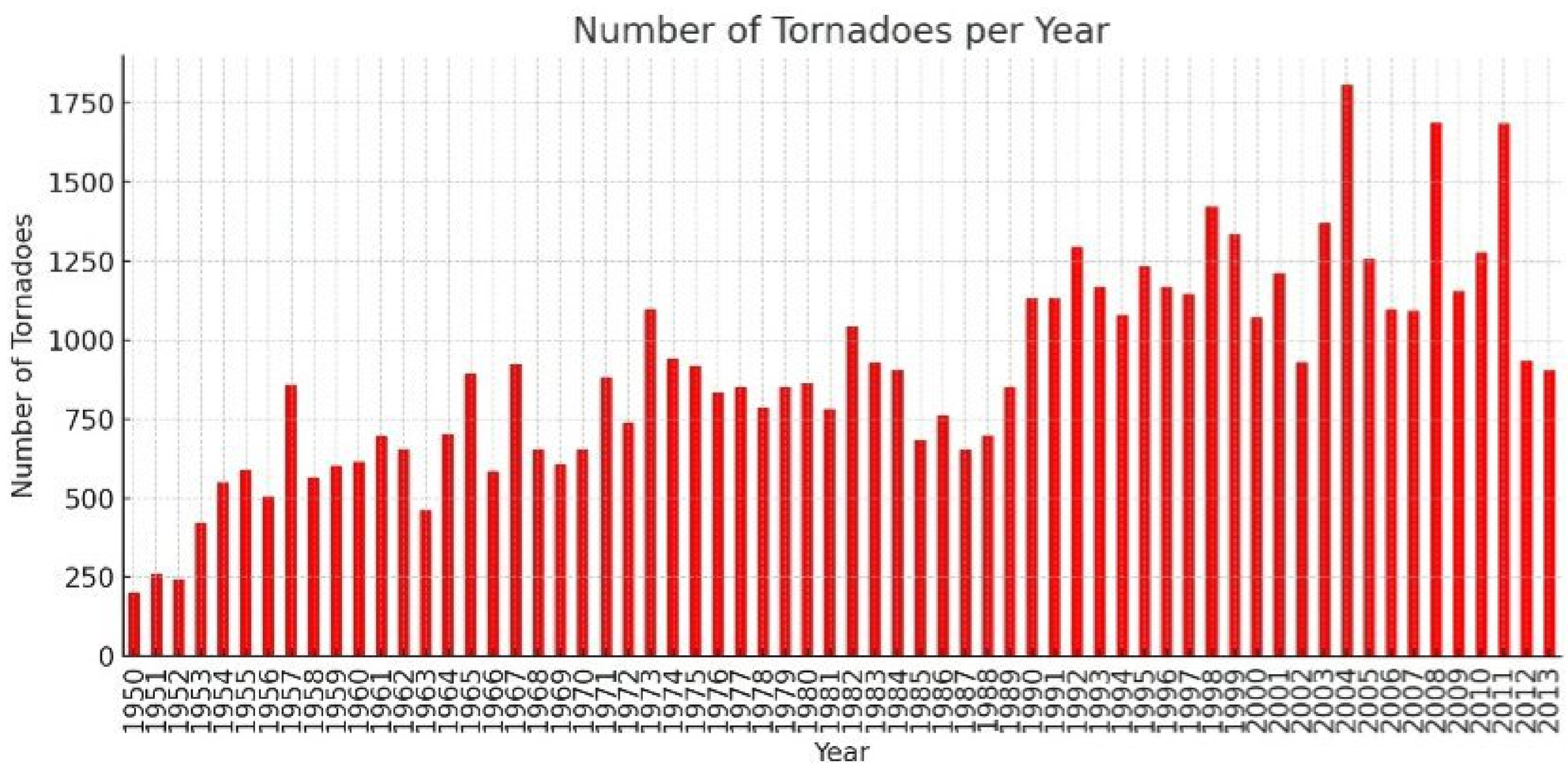
Tornado Data Visualization

Page 1: Source Code for Bar Chart and Scatter Plot

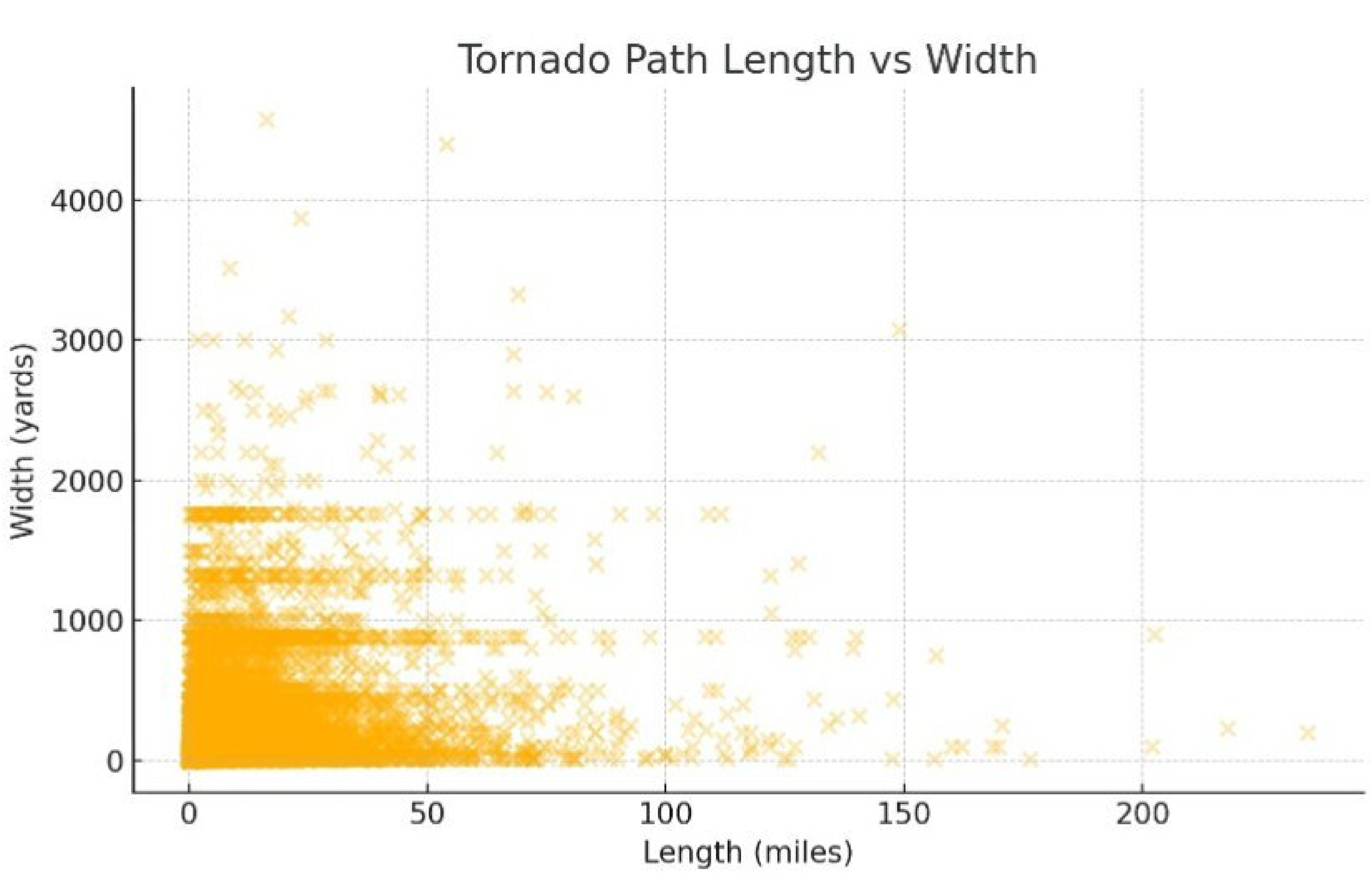
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file
df = pd.read_csv("Historical_Tornado_Tracks.csv")

# Bar Chart: Number of Tornadoes per Year
plt.figure(figsize=(10, 5))
df['YR'].value_counts().sort_index().plot(kind='bar', color='red')
plt.title('Number of Tornadoes per Year')
plt.xlabel('Year')
plt.ylabel('Number of Tornadoes')
plt.tight_layout()
plt.savefig('bar_chart.png')
plt.close()

# Scatter Plot: Length vs Width of Tornadoes
plt.figure(figsize=(8, 5))
plt.scatter(df['LEN'], df['WID'], alpha=0.3)
plt.title('Tornado Path Length vs Width')
plt.xlabel('Length (miles)')
plt.ylabel('Width (yards)')
plt.tight_layout()
plt.savefig('scatter_plot.png')
plt.close()
```

Page 2: Output - Visualizations

Bar Chart:



Scatter Plot:



Page 3: Software Requirements

1. Python 3.7 or above
2. pandas (version 1.0 or above)
3. matplotlib (version 3.0 or above)
4. Jupyter Notebook or any Python IDE
5. CSV file: Historical_Tornado_Tracks.csv

Optional:
- Microsoft Word (to view this document)