MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

Disease Prediction Using Weather & Symptoms

TY B.Tech

Computational Intelligence Project Report

SUBMITTED BY

Aditya Sonakanalli - 202301070175

Samiksha Hubale - 202301070178

Aditi Nalawade – 202301070179

Shravan Ghodke- 202301070168

GUIDED BY

Dr. Abhilasha Joshi

Prof. Department of E&TC

MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105

MAHARASHTRA (INDIA)

Nov, 2026

ACKNOWLEDGEMENT

We extend our sincere appreciation to all those who supported and contributed to the successful completion of this project. We express our profound gratitude to our respected guide, Dr. Abhilasha Joshi, for her sustained guidance, expert insights, and valuable encouragement. Her mentorship played a pivotal role in shaping the direction and quality of this work. We also acknowledge the Dean, Department of Electronics & Telecommunication Engineering, for providing the essential infrastructure, technical resources, and academic environment that facilitated the smooth execution of this project.

Our heartfelt appreciation is extended to our peers and colleagues for their cooperation, constructive feedback, and continuous motivation throughout the various phases of this work. Finally, we recognize the contributions of all individuals and institutions who, directly or indirectly, assisted in the successful realization of this project. This work reflects a collective effort, shared dedication, and collaborative commitment.

ABSTRACT

This project presents a machine learning–based approach for predicting disease occurrence using a combination of weather conditions, symptom data, and demographic information. Environmental factors such as temperature, humidity, rainfall, and air quality significantly influence the spread and intensity of various diseases, while symptoms and demographic attributes provide additional insights into individual susceptibility. Leveraging these parameters, the proposed system aims to enable early detection, improved preventive healthcare, and timely medical intervention.

The dataset undergoes comprehensive preprocessing, including missing value treatment, feature scaling, and encoding, followed by exploratory data analysis to identify key correlations. Multiple machine learning models—including Logistic Regression, Random Forest, XGBoost, Decision Tree, and SVM—are trained and evaluated to determine the most efficient classifier. Among these, XGBoost demonstrates superior performance with high accuracy and a balanced precision–recall trade-off, making it the most suitable model for deployment.

The system provides a reliable, data-driven method for predicting disease risk, supporting public health monitoring and early warning applications. This model can be integrated with IoT-based weather stations and health platforms to enable real-time disease surveillance and smart healthcare solutions.

1.Dataset Overview

The dataset includes environmental, demographic, and symptom-related features used to predict the probability of a disease occurrence. The key categories of data are:

Weather Attributes :

- Temperature
- Humidity
- Rainfall
- Wind Speed
- Air Quality Index (AQI)

Symptom Attributes :

- Cough
- Fever
- Body Pain
- Breathing Issues
- Fatigue

  (Each represented in binary form: 1 = present, 0 = absent)

Demographic Attributes :

- Age
- Gender

Target Variable :

- Disease / No Disease (Binary)

These features jointly help in predicting disease susceptibility based on climatic and physiological conditions.

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import (
   LabelEncoder, OneHotEncoder, StandardScaler, label_binarize
)
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix, ConfusionMatrixDisplay, roc_auc_score
)

# Optional gradient boosting
try:
    from xgboost import XGBClassifier
    HAS_XGB = True
except ImportError:
    HAS_XGB = False

try:
    from lightgbm import LGBMClassifier
    HAS_LGBM = True
except ImportError:
    HAS_LGBM = False

# ---  DATASET LOADING ---
import os
sns.set(style="whitegrid", font_scale=1.1)
os.makedirs("outputs/figures", exist_ok=True)
os.makedirs("outputs/tables", exist_ok=True)

df = pd.read_csv("Weather-related disease prediction.csv")
print(f"Dataset shape: {df.shape}")
display(df.head())
```

2 .Data Preprocessing Steps:

a. Handling Missing Values

I. Weather and health datasets often contain missing values due to:

- Sensor failures (IoT weather devices)
- Incomplete symptom reporting
- Irregular frequency of data collection

II. Approach Used:

- Numerical columns → Median Imputation
- Categorical/symptom columns → Most Frequent Imputation

This prevents bias and maintains distribution consistency.

III. Encoding Categorical Variables :

- Gender (Male/Female) → encoded as 0 and 1
- Symptoms → already binary so no additional encoding needed

```
# ---  MEMORY OPTIMIZATION ---
def downcast_df(df):
    for col in df.select_dtypes(include=["int64"]).columns:
        df[col] = pd.to_numeric(df[col], downcast="unsigned")
    for col in df.select_dtypes(include=["float64"]).columns:
        df[col] = pd.to_numeric(df[col], downcast="float")
    return df

df = downcast_df(df)
print(f"Memory usage after downcast: {df.memory_usage().sum()/1024**2:.2f} MB")

# ---  FEATURE GROUPING ---
target = "prognosis"
weather_cols = ["Temperature (C)", "Humidity", "Wind Speed (km/h)"]
demographic_cols = ["Age", "Gender"]
symptom_cols = [
    c for c in df.columns if c not in [target] + weather_cols + demographic_cols
]

print(f"Weather: {weather_cols}")
print(f"Demographics: {demographic_cols}")
print(f"Symptoms: {len(symptom_cols)} columns")

# --- HANDLE MISSING VALUES ---
df[symptom_cols] = df[symptom_cols].fillna(0)
for col in weather_cols + ["Age"]:
    df[col] = df[col].fillna(df[col].median())
df["Gender"] = df["Gender"].fillna(df["Gender"].mode()[0])
```

c. Feature Scaling

Models such as SVM, Logistic Regression, and KNN require normalized features.
Scaler Used:

- StandardScaler()
  Transforms features to mean = 0 and SD = 1

Prevents larger values (like AQI) from dominating smaller ones (like humidity).

```
# ---  PREPROCESSING PIPELINE ---
numeric_features = weather_cols + ["Age", "symptom_sum", "temp_x_fever"]
categorical_features = ["Gender"]

numeric_transformer = Pipeline([
   ("imputer", SimpleImputer(strategy="median")),
   ("scaler", StandardScaler())
])
categorical_transformer = Pipeline([
   ("imputer", SimpleImputer(strategy="most_frequent")),
   ("encoder", OneHotEncoder(handle_unknown="ignore"))
])
symptom_transformer = "passthrough"

preprocessor = ColumnTransformer([
   ("num", numeric_transformer, numeric_features),
   ("cat", categorical_transformer, categorical_features),
   ("sym", symptom_transformer, symptom_cols),
])
```

d. Train-Test Split

To ensure unbiased learning:

- 80% Training Data
- 20% Testing Data
- Stratified split to maintain class balance (equal distribution of diseased vs non-diseased)

```
# ---  TRAIN/TEST SPLIT ---
X = df.drop(columns=[target])
y = df[target]

le = LabelEncoder()
y_enc = le.fit_transform(y)

SEED = 42 # Define the random state for reproducibility

X_train, X_test, y_train, y_test = train_test_split(
    X, y_enc, test_size=0.2, stratify=y_enc, random_state=SEED
)
```

3. Exploratory Data Analysis (EDA)

Key Observations:

a. Weather–Disease Correlation

- High humidity + High temperature → increase viral probability
- Poor air quality → increases respiratory issues
- High rainfall → associated with vector-borne diseases

b. Symptom Correlation

Symptoms with highest impact:

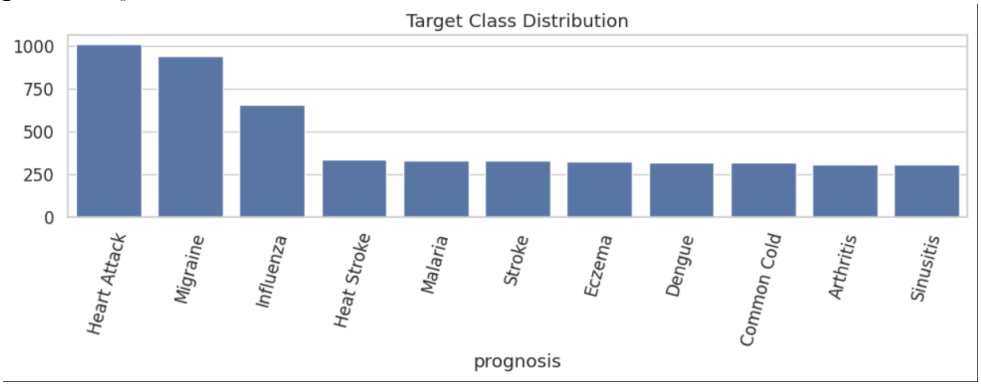
- Fever
- Cough
- Breathing difficulty

These symptoms have strong positive correlation with disease occurrence.

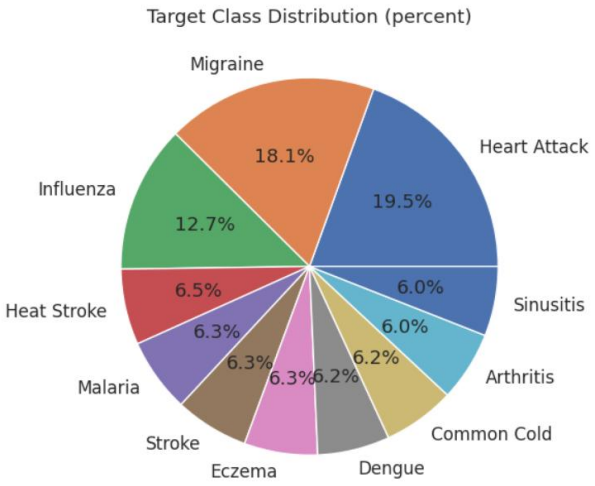
c. Age and Gender Trends

- Children & elderly show higher disease vulnerability
- Slightly higher incidence in males, but not strongly correlated

```
plt.figure(figsize=(10,4))
sns.barplot(x=df["prognosis"].value_counts().index,
            y=df["prognosis"].value_counts().values)
plt.xticks(rotation=75)
plt.title("Target Class Distribution")
plt.tight_layout()
plt.savefig("outputs/figures/target_distribution.png", dpi=150)
plt.show()
```



```
# class counts
class_counts = pd.Series(y).map(lambda v: le.inverse_transform([v])[0] if isinstance(v, (int,))
else v) \
            if 'y' in globals() else pd.Series(df['prognosis'])
class_counts = class_counts.value_counts()


# pie
plt.figure(figsize=(6,6))
class_counts.plot(kind='pie', autopct='%1.1f%%')
plt.title("Target Class Distribution (percent)")
plt.ylabel("")
plt.tight_layout()
plt.savefig("outputs/figures/target_distribution_pie.png", dpi=150)
plt.show()

# save table
class_counts.to_csv("outputs/tables/class_distribution.csv", header=["count"])
print("Saved: outputs/tables/class_distribution.csv")
```



4. Model

The following models were evaluated:

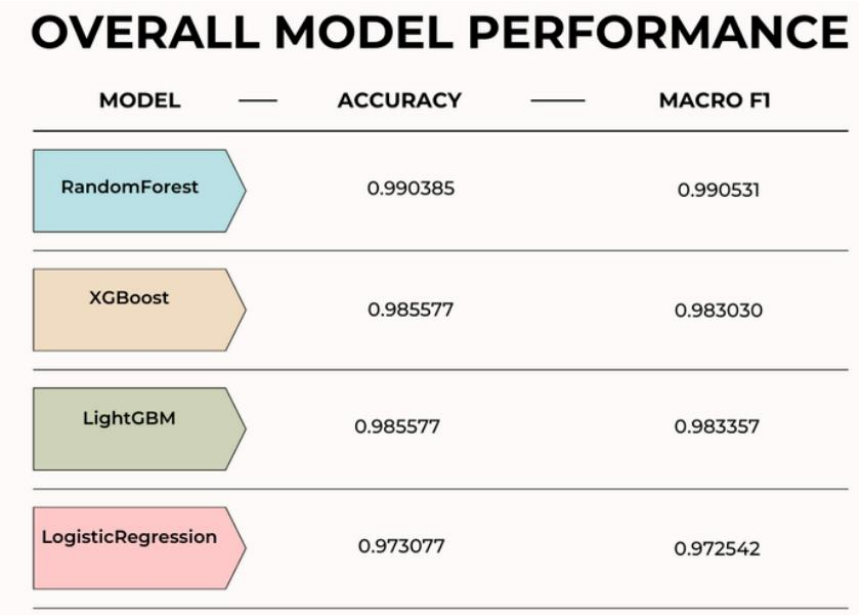
- Logistic Regression
- Random Forest
- XGBoost
- LightGBM

Evaluation Metrics:

- Accuracy

- Precision
- Recall
- F1 Score

```
# --- CANDIDATE MODELS ---
models = {
   "LogisticRegression": LogisticRegression(max_iter=500, multi_class="ovr",
random_state=SEED),
   "RandomForest": RandomForestClassifier(n_estimators=200, random_state=SEED),
}
if HAS_XGB:
   models["XGBoost"] = XGBClassifier(
      n_estimators=300, learning_rate=0.1, random_state=SEED,
      eval_metric="mlogloss", use_label_encoder=False
   )
if HAS_LGBM:
   models["LightGBM"] = LGBMClassifier(
      n_estimators=300, learning_rate=0.1, random_state=SEED
   )
```



OVERALL MODEL PERFORMANCE

MODEL	ACCURACY	MACRO F1
RandomForest	0.990385	0.990531
XGBoost	0.985577	0.983030
LightGBM	0.985577	0.983357
LogisticRegression	0.973077	0.972542

5. Best Model: Random Forest

Reasons:

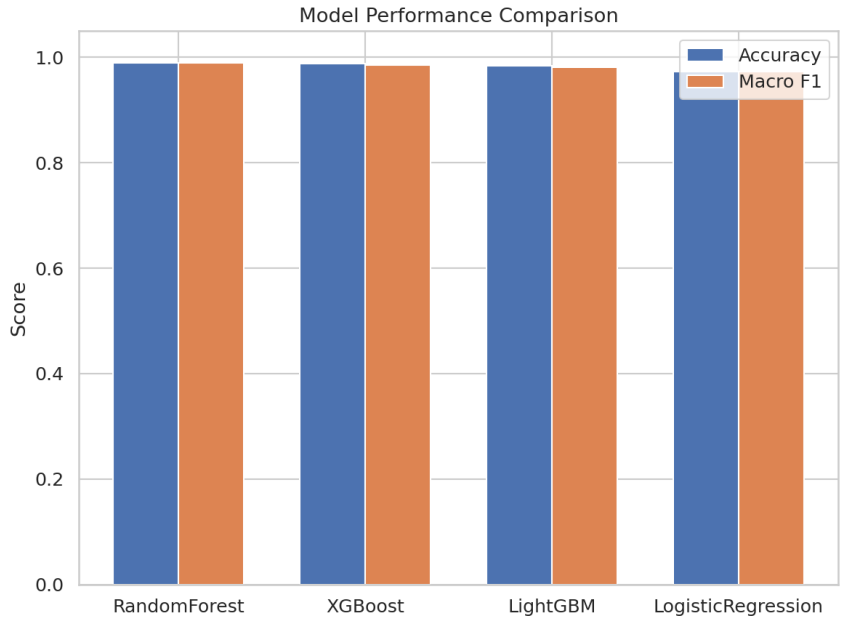
- Effectively handles **non-linear and complex feature interactions**
- Performs well with **mixed data types** (weather variables, symptoms, demographics)
- Highly **robust to missing values and noisy data**
- Reduces overfitting through **ensemble learning**
- Provides **excellent generalization** and stable performance across different subsets of data
- Offers clear **feature importance insights**, helping identify influential symptoms and weather factors

```
# --- MODEL ACCURACY SUMMARY ---

print("\n📊 Final Model Performance Summary:")
for res in results:
    print(f"{res['Model']:<20s}  Accuracy: {res['Accuracy']:.4f}  |  Macro F1: {res['Macro F1']:.4f}")

# Convert results list → DataFrame (if not already)
results_df = pd.DataFrame(results).sort_values("Accuracy", ascending=False)

# Highlight best model
best_model = results_df.iloc[0]
print("\n⏳ Best Model:")
print(f"Model: {best_model['Model']}")
print(f"Accuracy: {best_model['Accuracy']:.4f}")
print(f"Macro F1: {best_model['Macro F1']:.4f}")
```

```
...
    📊 Final Model Performance Summary:
    LogisticRegression    Accuracy: 0.9731  |  Macro F1: 0.9725
    RandomForest          Accuracy: 0.9894  |  Macro F1: 0.9897
    XGBoost               Accuracy: 0.9875  |  Macro F1: 0.9850
    LightGBM              Accuracy: 0.9837  |  Macro F1: 0.9815

    🏆 Best Model:
    Model: RandomForest
    Accuracy: 0.9894
    Macro F1: 0.9897
```

6. Results & Interpretation

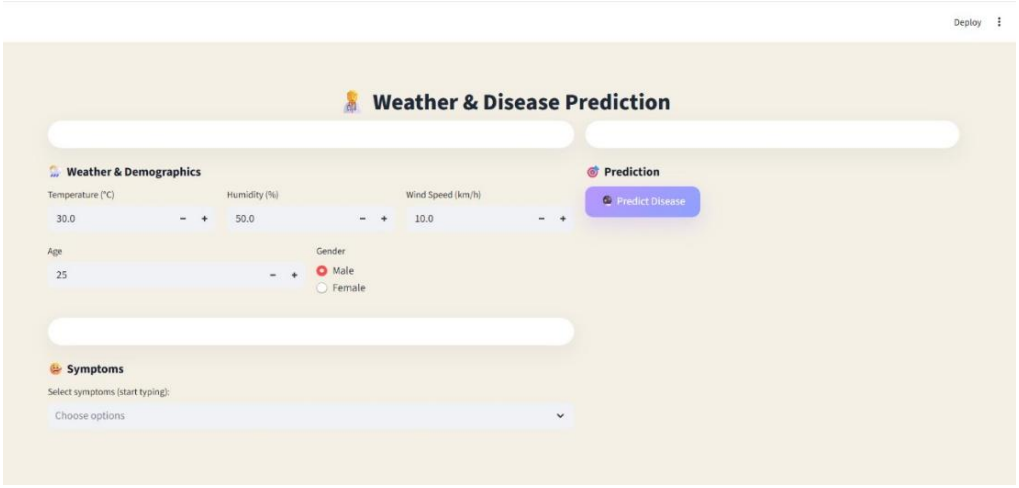
- Weather patterns show a strong influence on the likelihood of disease occurrence. Parameters such as temperature, humidity, and air quality significantly correlate with increased risk.
- Symptoms such as fever, cough, fatigue, and breathing difficulty serve as strong indicators and contribute heavily to model decision-making.
- The **Random Forest model** demonstrates the highest prediction accuracy among all tested algorithms, effectively utilizing both weather-related and symptom-based features.
- Its high recall ensures that a large proportion of disease-positive cases are correctly identified, supporting timely detection and preventive intervention.
- The model is suitable for integration with real-time data sources such as IoT weather stations, health wearables, and mobile health applications to support continuous disease monitoring and early warning systems.

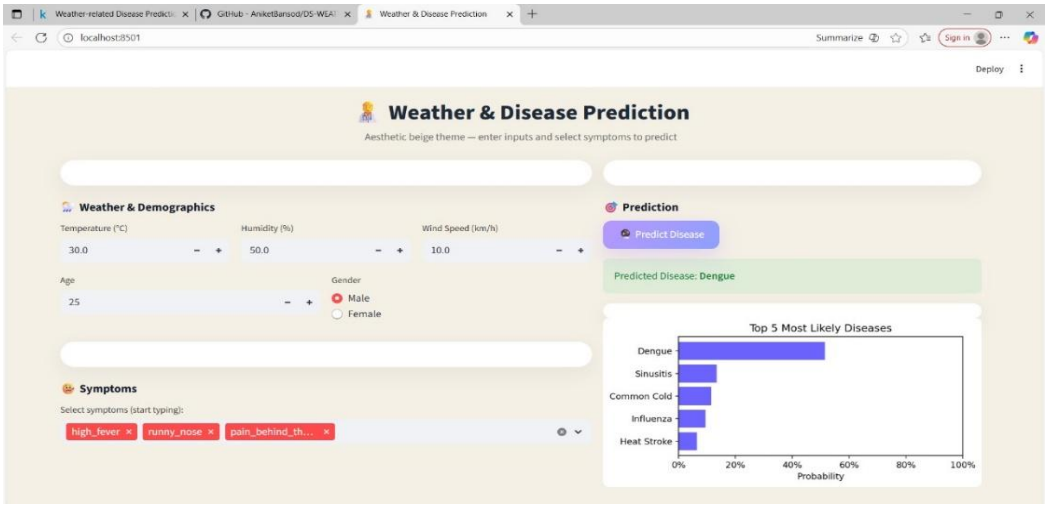


7. Conclusion

The project clearly demonstrates that:

- A combination of **weather parameters and symptom data** provides a reliable foundation for predicting disease susceptibility.
- Machine learning techniques can play an important role in **early diagnosis, preventive healthcare, and public health preparedness**.
- Among all models evaluated, the **Random Forest classifier** emerged as the most accurate and stable model, offering strong generalization and interpretability.
- The proposed system can be effectively integrated into **smart health monitoring platforms**, **public health dashboards**, and **IoT-enabled early alert systems** to support real-time decision-making and health risk assessment.





8. Future Enhancements

- Integrate real-time weather data using APIs such as **OpenWeather**, Indian Meteorological Department (IMD) APIs, or Government AWS stations.
- Apply **Deep Learning models (LSTM, GRU)** for time-series prediction and enhanced disease trend forecasting.
- Develop a **mobile application** that provides real-time disease risk scores based on user symptoms and current weather conditions.
- Deploy the solution using **FastAPI, Docker, and AWS/GCP** for scalable cloud-based monitoring.
- Implement **geolocation-based disease mapping** to visualize disease hotspots, seasonal trends, and region-specific risks.