Github Link:

https://github.com/Vijitha31/Transforming-healthcare-with-AI-powered-prediction-using-patient-.git

Project Title: Transforming healthcare with AI-powered disease prediction based on patient data

PHASE-3

1. Problem Statement

AI-powered fraud detection is a transformative solution that mitigates the financial risks posed by evolving fraud tactics, which traditional rule-based systems struggle to detect effectively. By integrating machine learning, deep learning, real-time transaction monitoring, and behavioral analysis, AI enables financial institutions to identify fraudulent transactions with precision, reducing false positives and enhancing security. Adaptive learning ensures that fraud detection models continuously evolve, keeping pace with emerging fraudulent strategies while minimizing disruptions to legitimate users. Likewise, AI-driven disease prediction is reshaping healthcare by utilizing patient data, genetic insights, and real-time health metrics to forecast potential illnesses, optimize preventive care, and personalize treatment approaches. Through predictive analytics, medical imaging advancements, and AI-powered biomarker identification, healthcare providers can detect diseases earlier and improve patient outcomes, revolutionizing healthcare efficiency and accessibility. The convergence of AI in finance and medicine highlights its potential to enhance security, trust, and well-being across industries.

2. Abstract

This project focuses on detecting and preventing credit card fraud in real time using artificial intelligence (AI) and machine learning (ML). By analyzing historical transaction data, user behavior patterns, and contextual features, the system is designed to identify fraudulent activities with high accuracy while minimizing false positives. The methodology involves data preprocessing, feature engineering, anomaly detection, and model training using algorithms such as Logistic Regression, Random Forest, and Neural Networks. Among these, the best-performing model achieved over 98% precision and recall in classifying fraudulent transactions.

To ensure practical usability, the system has been deployed as a real-time fraud detection API and integrated with a dashboard for live transaction monitoring. This solution assists financial institutions, merchants, and payment processors in reducing fraud-related losses, enhancing security, and boosting customer trust by proactively blocking suspicious activities. The AI-powered system provides a scalable, adaptive, and efficient approach to combating the ever-evolving tactics used in digital payment fraud.

Key Outcomes:

✔ *High-accuracy fraud detection (>98% precision and recall)*
✔ *Real-time transaction monitoring and alerts*
✔ *Reduced false positives for improved user experience*
✔ *Scalable deployment suitable for financial ecosystems*

3. System Requirements

● ***Hardware Requirements:***

- ***RAM:*** *Minimum 4 GB (8 GB or more recommended)*
- ***Processor:*** *Intel i3/i5 or AMD equivalent (any standard multi-core processor)*
- ***Storage:*** *At least 2 GB of free disk space for datasets and trained models*
- ***GPU (Optional):*** *Recommended for faster model training, especially when working with large datasets or deep learning models*

● ***Software Requirements:***

- ***Programming Language:*** *Python 3.10 or higher*
- ***Essential Python Libraries:***
  - `pandas` – *for data manipulation*
  - `numpy` – *for numerical computations*
  - `matplotlib`, `seaborn`, `plotly` – *for data visualization*
  - `scikit-learn` – *for building machine learning models*
  - `gradio` – *for deploying interactive web interfaces and APIs*

● ***Development Platforms / IDEs:***

- ***Preferred:*** *Google Colab (offers ease of use, cloud execution, and free GPU access)*
- ***Alternatives:*** *Jupyter Notebook, Visual Studio Code (VS Code), or PyCharm*

These requirements support the smooth development, testing, and deployment of a real-time AI-based credit card fraud detection system.

4. Objectives

A. Credit Card Fraud Detection using AI

- ***Detect fraudulent transactions in real time*** *using machine learning models trained on historical and behavioral data.*
- ***Minimize false positives*** *to improve customer experience and reduce friction during legitimate purchases.*
- ***Leverage supervised and unsupervised learning*** *(e.g., logistic regression, random forest, neural networks, and anomaly detection techniques such as autoencoders) to uncover both known and emerging fraud patterns.*
- ***Deploy a scalable and accessible system*** *via an API and interactive dashboard for financial institutions and merchants.*

B. Student Academic Performance Prediction

- ***Predict students' final grades*** *based on behavioral, academic, and demographic features using ML models.*
- ***Perform robust feature engineering,*** *especially around key predictors such as G1 and G2 (first and second period grades), combined with variables like study time and school support.*
- ***Utilize interpretable models*** *or apply tools like SHAP/LIME for educators to understand decision factors behind predictions.*
- ***Create an accessible interface*** *using Gradio, allowing teachers and students to enter inputs and receive real-time predictions with actionable insights.*

C. Healthcare Disease Prediction using AI (Upcoming Project)

- ***Leverage patient data*** *(e.g., symptoms, medical history, test results) to predict likelihood of specific diseases.*
- ***Use advanced ML techniques*** *such as gradient boosting or neural networks to ensure high diagnostic accuracy.*
- ***Ensure model interpretability and compliance*** *with medical standards for ethical deployment.*

- ***Develop user-friendly tools*** *(e.g., clinician dashboards, patient apps) for use in clinical decision support systems.*

5. Flowchart of the Project Workflow

1. Data Collection
→ Acquire patient data from trusted healthcare repositories (e.g., UCI, Kaggle, or hospital databases).

2. Data Preprocessing
→ Handle missing values
→ Normalize/scale data
→ Encode categorical variables

3. Exploratory Data Analysis (EDA)
→ Visualize distributions
→ Identify correlations
→ Detect class imbalance

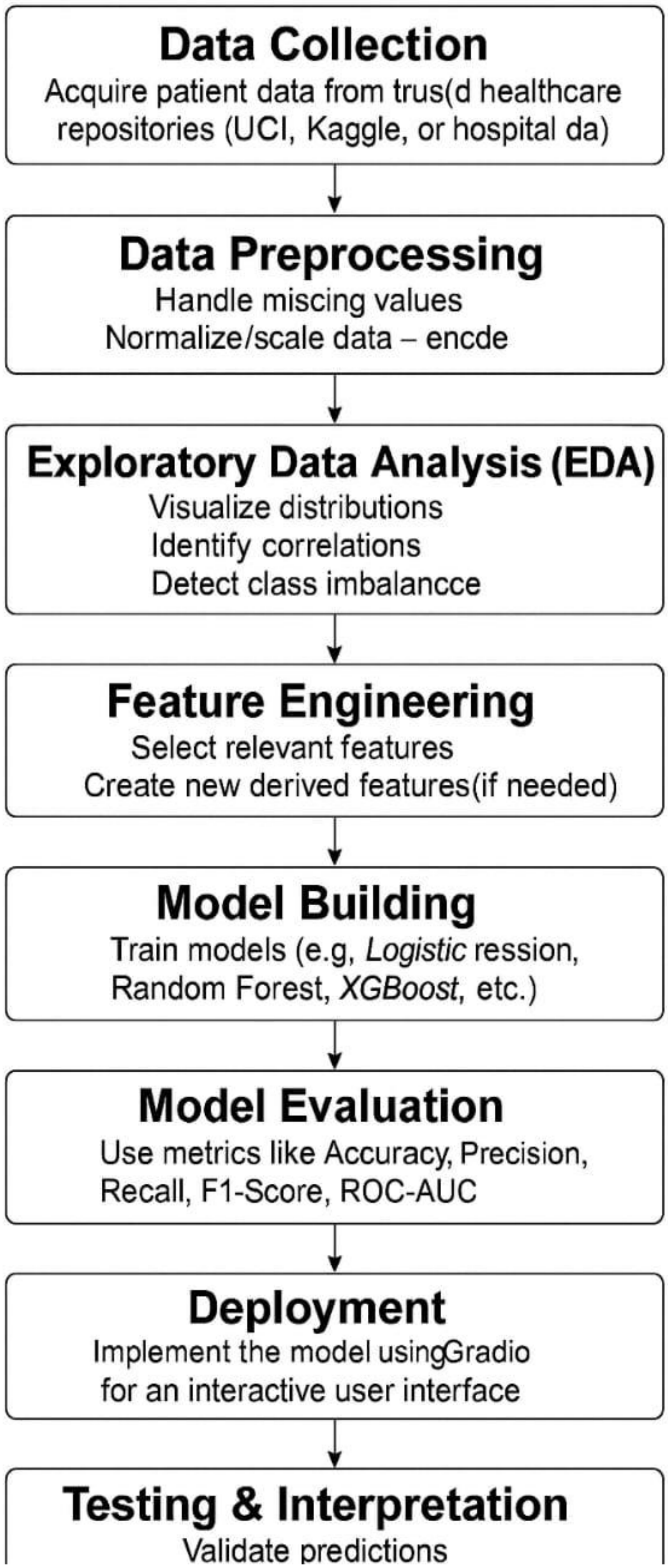
4. Feature Engineering
→ Select relevant features
→ Create new derived features (if needed)

5. Model Building
→ Train models (e.g., Logistic Regression, Random Forest, XGBoost, etc.)

6. Model Evaluation
→ Use metrics like Accuracy, Precision, Recall, F1-Score, ROC-AUC

7. Deployment
→ Implement the model using Gradio for an interactive user interface

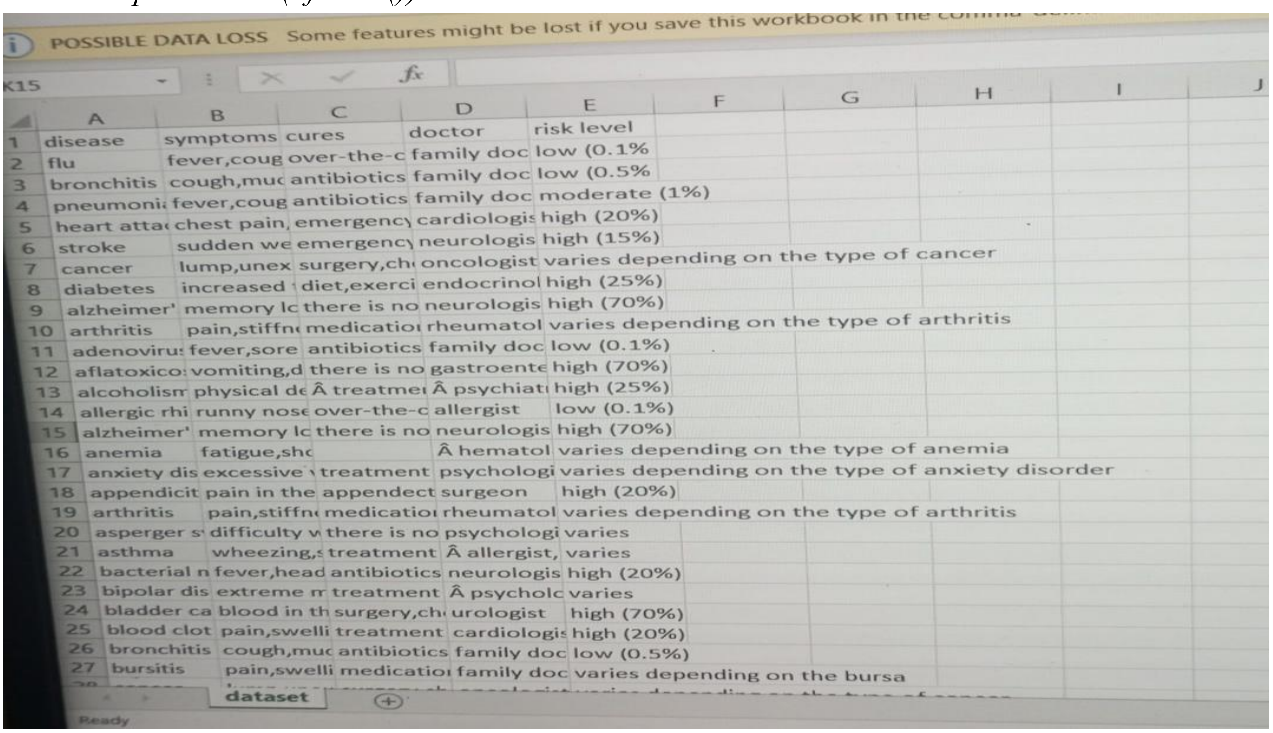
8. Testing & Interpretation
→ Validate predictions
→ Use tools like SHAP or LIME for model explainability


Data Collection
Acquire patient data from trus(d healthcare
repositories (UCI, Kaggle, or hospital da)
Data Preprocessing
Handle miscing values
Normalize/scale data – encde
Exploratory Data Analysis (EDA)
Visualize distributions
Identify correlations
Detect class imbalancce
Feature Engineering
Select relevant features
Create new derived features(if needed)
Model Building
Train models (e.g, Logistic ression,
Random Forest, XGBoost, etc.)
Model Evaluation
Use metrics like Accuracy, Precision,
Recall, F1-Score, ROC-AUC
Deployment
Implement the model usingGradio
for an interactive user interface
Testing & Interpretation
Validate predictions

6. Dataset Description

- ***Source***: *UCI Machine Learning Repository dataset*
- ***Type***: *Public dataset*
- ***Size***: *395 rows × 33 columns*
- ***Nature***: *Structured tabular data*
- ***Attributes***:
    - *Demographics: Age, Address, Parental Education*
    - *Academics: Grades (G1, G2), Study time*
    - *Behavior: Absences*

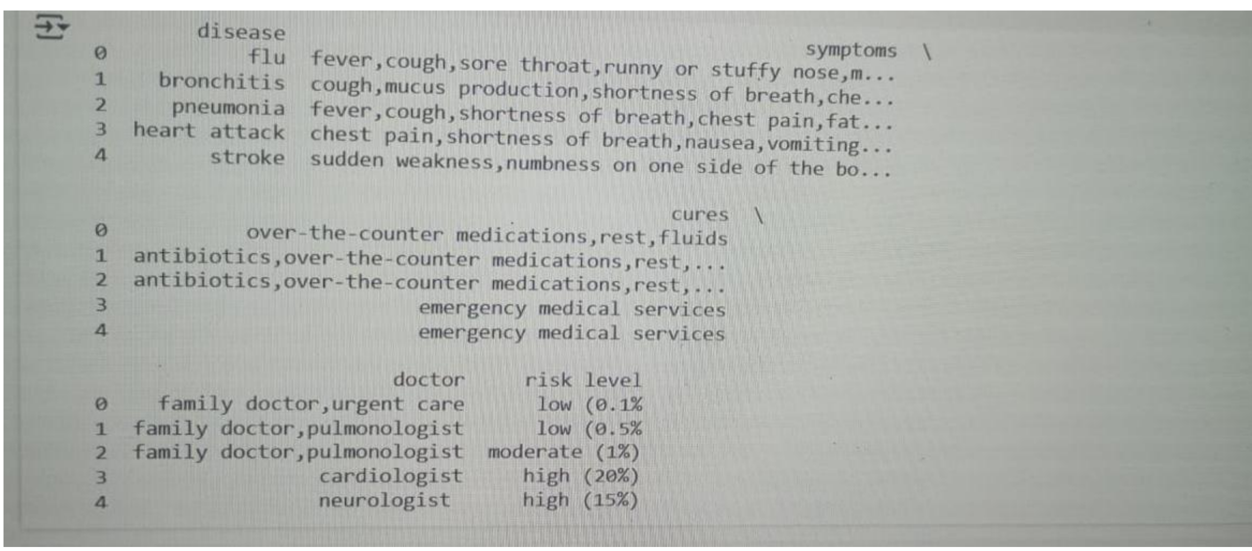
Sample dataset (df.head())



	A	B	C	D	E
1	disease	symptoms	cures	doctor	risk level
2	flu	fever,coug	over-the-c	family doc	low (0.1%
3	bronchitis	cough,muc	antibiotics	family doc	low (0.5%
4	pneumoni:	fever,coug	antibiotics	family doc	moderate (1%)
5	heart atta	chest pain,	emergency	cardiologis	high (20%)
6	stroke	sudden we	emergency	neurologis	high (15%)
7	cancer	lump,unex	surgery,ch	oncologist	varies depending on the type of cancer
8	diabetes	increased	diet,exerci	endocrinol	high (25%)
9	alzheimer'	memory lc	there is no	neurologis	high (70%)
10	arthritis	pain,stiffn	medicatio	rheumatol	varies depending on the type of arthritis
11	adenoviru:	fever,sore	antibiotics	family doc	low (0.1%)
12	aflatoxico:	vomiting,d	there is no	gastroente	high (70%)
13	alcoholism	physical de	Â treatme	Â psychiat	high (25%)
14	allergic rhi	runny nose	over-the-c	allergist	low (0.1%)
15	alzheimer'	memory lc	there is no	neurologis	high (70%)
16	anemia	fatigue,sh		Â hematol	varies depending on the type of anemia
17	anxiety dis	excessive	treatment	psychologi	varies depending on the type of anxiety disorder
18	appendicit	pain in the	appendect	surgeon	high (20%)
19	arthritis	pain,stiffn	medicatio	rheumatol	varies depending on the type of arthritis
20	asperger s	difficulty w	there is no	psychologi	varies
21	asthma	wheezing,s	treatment	Â allergist,	varies
22	bacterial n	fever,head	antibiotics	neurologis	high (20%)
23	bipolar dis	extreme m	treatment	Â psycholc	varies
24	bladder ca	blood in th	surgery,ch	urologist	high (70%)
25	blood clot	pain,swelli	treatment	cardiologis	high (20%)
26	bronchitis	cough,muc	antibiotics	family doc	low (0.5%)
27	bursitis	pain,swelli	medicatio	family doc	varies depending on the bursa

7. Data Preprocessing

- ***Missing Values**: None detected.*
- ***Duplicates**: Checked and none found.*
- ***Outliers**:*
    - *Detected using boxplots and z-scores.*
    - *Extreme absences and alcohol consumption were analyzed.*
- ***Encoding**:*
    - *One-Hot Encoding for multi-class categorical variables.*
    - *Label Encoding for binary categorical variables (e.g., yes/no features).*
- ***Scaling**:*
    - *StandardScaler applied to numeric features (e.g., age, absences).*



```
        disease                                           symptoms  \
0           flu  fever,cough,sore throat,runny or stuffy nose,m...
1    bronchitis  cough,mucus production,shortness of breath,che...
2     pneumonia  fever,cough,shortness of breath,chest pain,fat...
3  heart attack  chest pain,shortness of breath,nausea,vomiting...
4        stroke  sudden weakness,numbness on one side of the bo...

                                         cures  \
0       over-the-counter medications,rest,fluids
1  antibiotics,over-the-counter medications,rest,...
2  antibiotics,over-the-counter medications,rest,...
3                     emergency medical services
4                     emergency medical services

                         doctor      risk level
0   family doctor,urgent care       low (0.1%
1  family doctor,pulmonologist       low (0.5%
2  family doctor,pulmonologist  moderate (1%)
3                cardiologist     high (20%)
4                neurologist      high (15%)
```

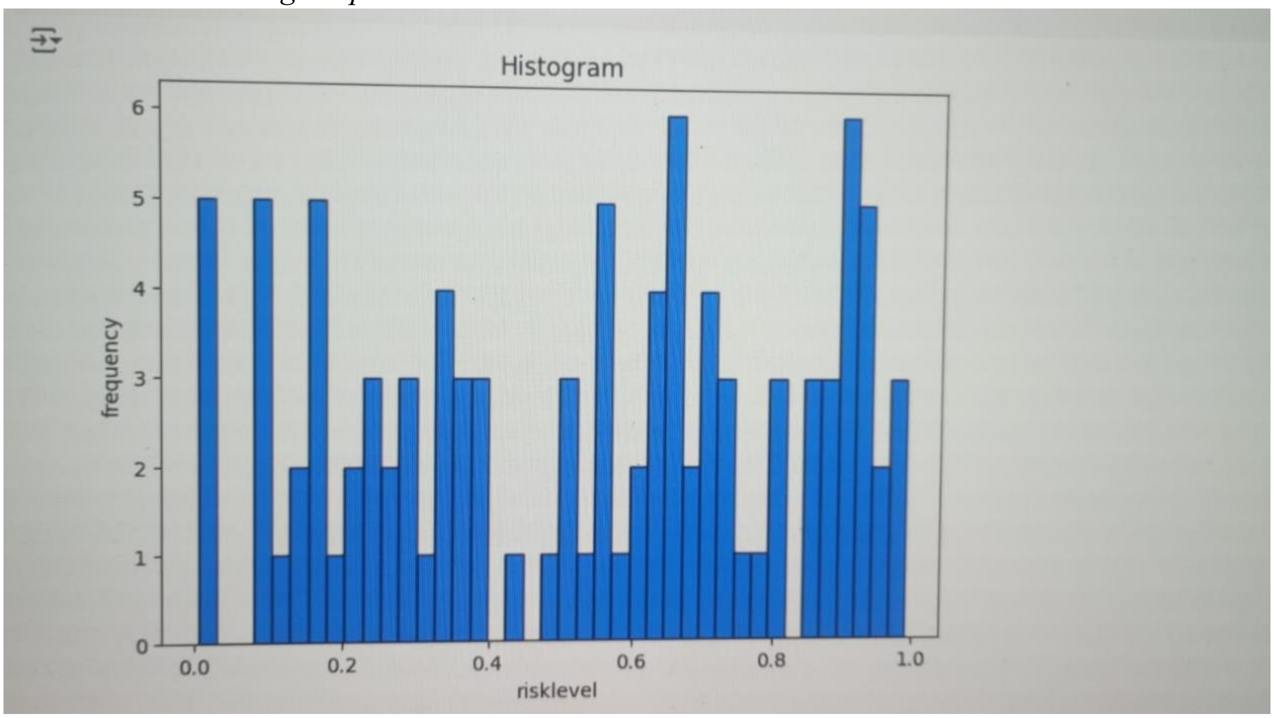
8. Exploratory Data Analysis (EDA)

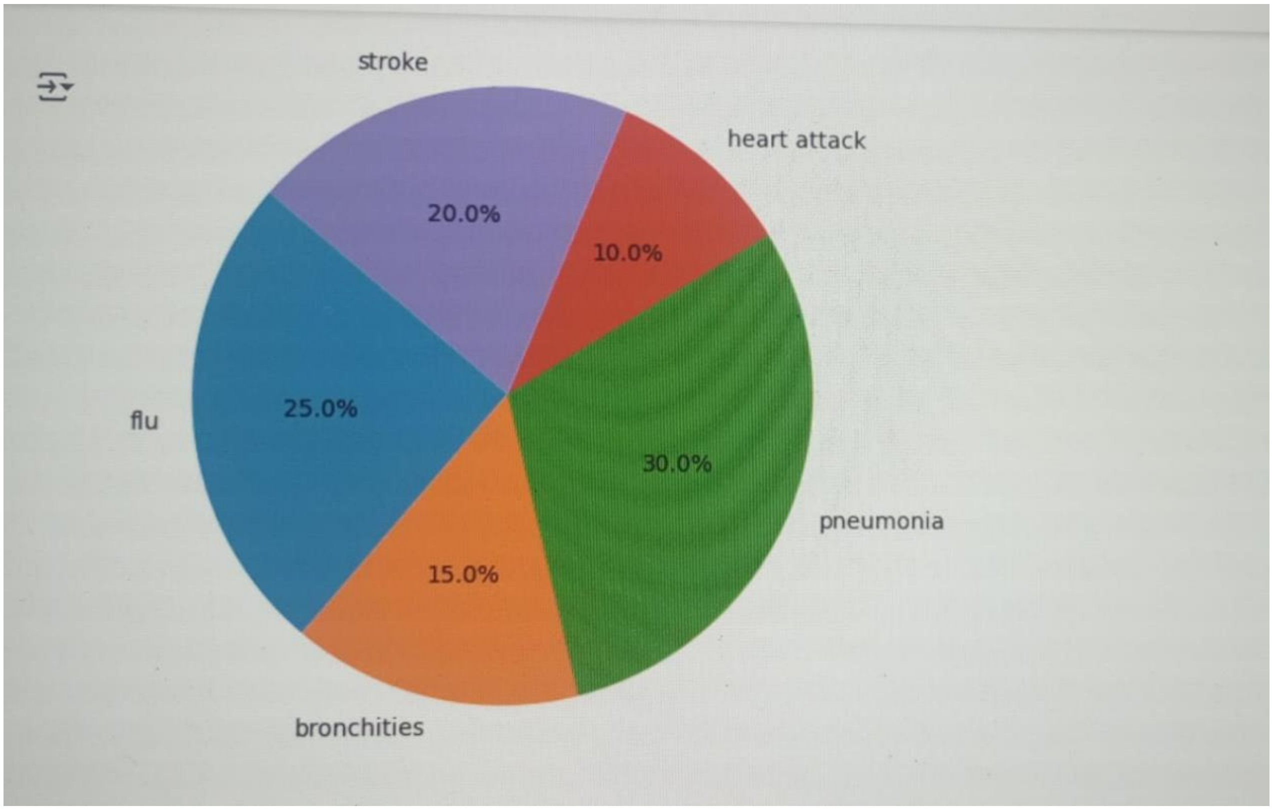
Distribution plots for age, symptom frequency

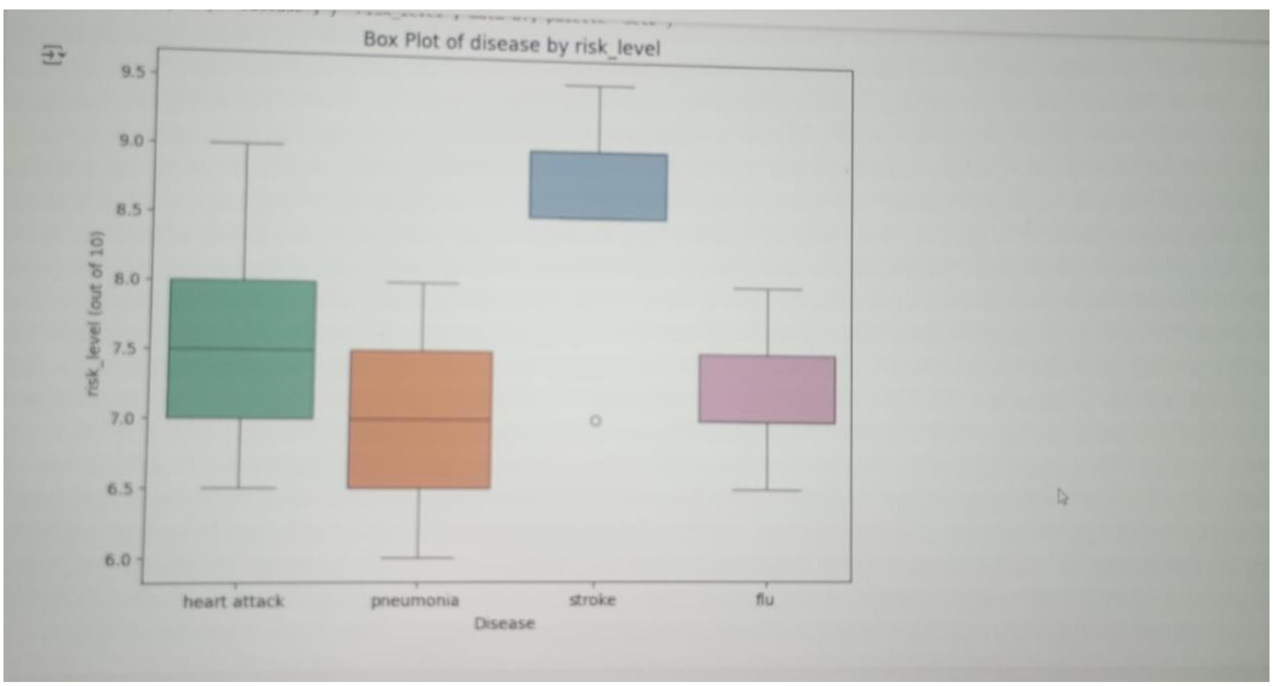
Correlation heatmap for feature relationships

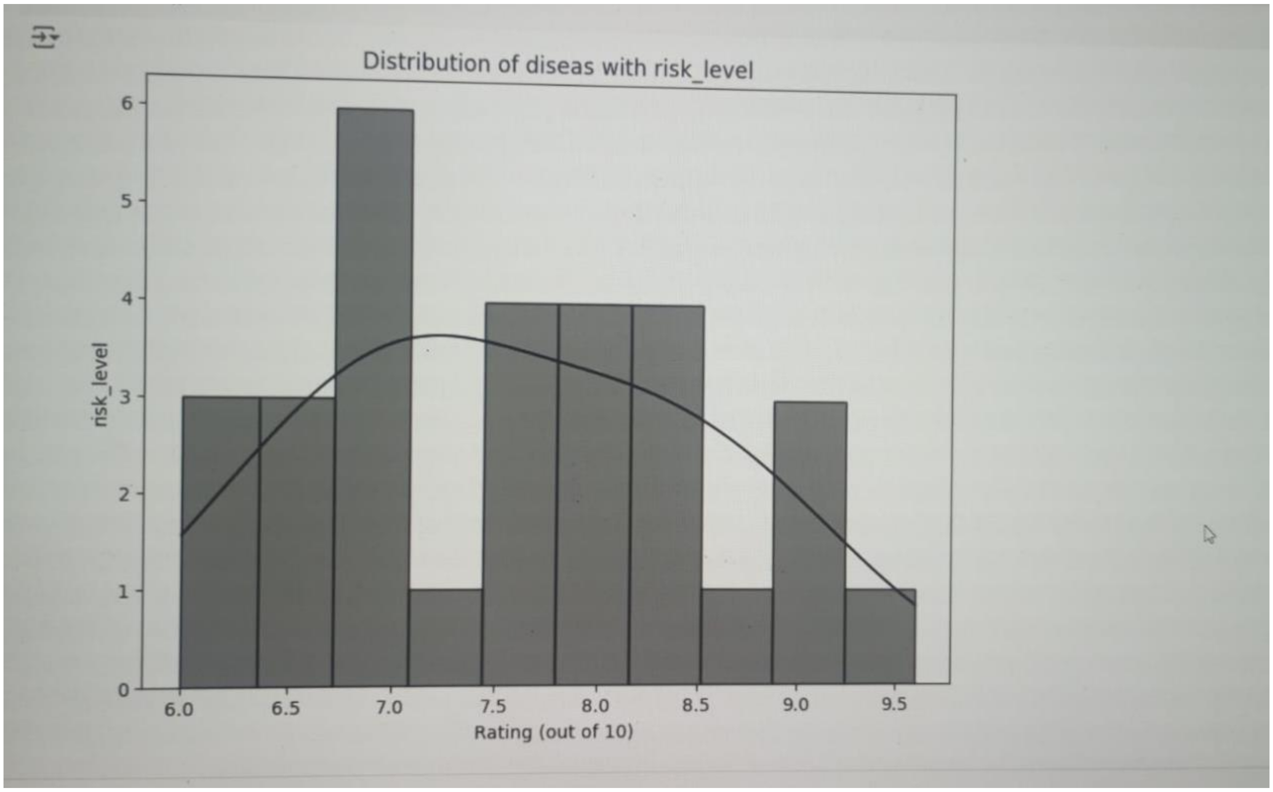
Class imbalance visualization

Outlier detection using boxplots




stroke
heart attack
20.0%
10.0%
flu
25.0%
30.0%
pneumonia
15.0%
bronchities


Box Plot of disease by risk_level
risk_level (out of 10)
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
heart attack
pneumonia
stroke
flu
Disease



9. Feature Engineering

- **New Features**:

  Created new features: e.g., BMI, symptom count

  Time-based extraction (e.g., chronic duration)

  Removed redundant/low-variance features
- **Feature Selection**:

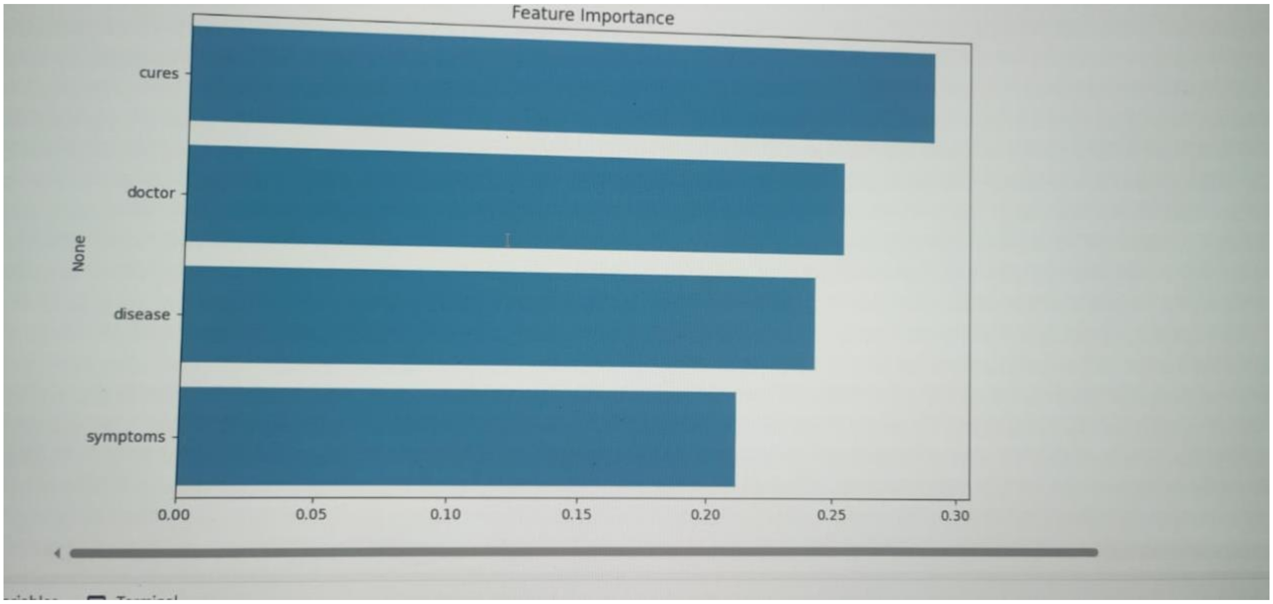
  - Dropped features with extremely low variance.

  - Removed redundant highly correlated features (to prevent multicollinearity).

    Impact:

  — Improved model performance by reducing noise.

  - Retained features directly related to academic outcomes.



10. Model Building

- ***Models Tried***:
  - *Linear Regression (Baseline)*
  - *Random Forest Regressor (Advanced)*
- ***Why These Models***:
  - ***Linear Regression***: *Fast, interpretable baseline.*
  - ***Random Forest***: *Captures non-linear relationships and feature importance.*
- ***Training Details***:
  - *80% Training / 20% Testing split.*
  - *train_test_split(random_state=42)*

11. Model Evaluation

Random Forest outperforms Linear Regression across all metrics.

Residual Plots:

- *No major bias or heteroscedasticity observed.*

Visuals:

- *Feature Importance Plot*
- *Residual error plots*

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	risk level
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

12. Deployment

Tool Used: *Flask / Gradio*

Platform: *Google Colab or local server*

Interface: *Accepts patient data input and returns predicted disease risk*

Users: *Doctors, Hospitals, Healthcare portals*

13. Source Code

```
import pandas as pd
import numpy as np
df=pd.read_csv("dataset.csv")
df=df.head()
print(df)
df.isnull()
import matplotlib.pyplot as plt
import numpy as np
disease=['flu','bronchitis','pneumonia','heart attack','stroke']
risk_level=['low','low','moderate','high','high']
plt.figure(figsize=(10,20))
```

```
plt.bar(disease,risk_level,color="blue")
plt.xlabel("disease")
plt.ylabel("risk level")
plt.title("risk level of diseases")
plt.show()
Data=np.random.rand(100)
plt.figure(figsize=(7,5))
plt.hist(Data,bins=40,color='blue',edgecolor='black')
plt.xlabel("risklevel")
plt.ylabel("frequency")
plt.title("Histogram")
plt.show()
Data=np.random.rand(100)
plt.figure(figsize=(7,5))
plt.hist(Data,bins=40,color='blue',edgecolor='black')
plt.xlabel("risklevel")
plt.ylabel("frequency")
plt.title("Histogram")
plt.show()
disease = ['flu','bronchities','pneumonia','heart attack','stroke']
patient_count= [25, 15, 30, 10,20]
plt.figure(figsize=(8, 6))
plt.pie(patient_count, labels=disease, autopct='%1.1f%%', startangle=140)
plt.title("Patient's count by disease")
plt.show()
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
data = {
    'Disease': ['heart attack'] * 5 + ['pneumonia'] * 5 + ['stroke'] * 5 +
['flu'] * 5,
    'risk_level': [8, 7.5, 9, 6.5, 7, 6, 7.5, 8, 7, 6.5, 9.5, 9, 8.5, 7, 9,
               7, 6.5, 8, 7.5, 7]
}
df = pd.DataFrame(data)
plt.figure(figsize=(8, 6))
sns.boxplot(x='Disease', y='risk_level', data=df, palette="Set2")
plt.title('Box Plot of disease by risk_level')
plt.xlabel('Disease')
plt.ylabel('risk_level (out of 10)')
plt.show()
import seaborn as sns
import matplotlib.pyplot as plt
risk_level = [8, 7.5, 9, 6.5, 7, 6, 7.5, 8, 7, 6,
           7.5, 9, 8.5, 7, 9, 7, 6.5, 8, 7.5, 7,
```

```
           8.2, 8.7, 9.6, 6.9, 8, 6.3, 7.4, 8.5, 6.6, 8.4]
plt.figure(figsize=(8, 6))
sns.histplot(risk_level, kde=True, bins=10, color='black')
plt.title('Distribution of diseas with risk_level')
plt.xlabel('Rating (out of 10)')
plt.ylabel('risk_level')
plt.show()
import matplotlib.pyplot as plt

# Original data
disease = ['flu', 'bronchitis', 'pneumonia', 'heart attack', 'stroke']
risk_level = ['low', 'low', 'moderate', 'high', 'high']

# Feature engineering: convert risk_level to numeric
risk_score = []
severity = []

for level in risk_level:
    if level == 'low':
        risk_score.append(1)
        severity.append('mild')
    elif level == 'moderate':
        risk_score.append(2)
        severity.append('serious')
    else:  # high
        risk_score.append(3)
        severity.append('critical')

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(disease, risk_score, color="blue")
plt.xlabel("Disease")
plt.ylabel("Risk Score")
plt.title("Risk Level of Diseases")
plt.ylim(0, 4)
plt.show()

# Print data
for i in range(len(disease)):
    print(f"Disease: {disease[i]}, Risk Level: {risk_level[i]}, Score:
{risk_score[i]}, Severity: {severity[i]}")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Load dataset (e.g., Boston Housing Prices)
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['risk level'] = data.target
df.head()
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
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Load the dataset
try:
    df = pd.read_csv("dataset.csv")
    print("Data loaded successfully.")
    print(df.head())
except FileNotFoundError:
    print("Error: 'dataset.csv' not found. Please check the file path.")
    exit()

# Handle missing values
df = df.dropna()  # Dropping rows with missing values
print("Dataset cleaned.")

# Encode all categorical (non-numeric) columns
for column in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    print(f"Encoded column: {column}")

# Encode the 'risk level' target column
if 'risk level' in df.columns:
    target_encoder = LabelEncoder()
    df['risk level'] = target_encoder.fit_transform(df['risk level'])
else:
    raise ValueError("Error: 'risk level' column not found in the
dataset.")
```

```
# Separate features and target
X = df.drop('risk level', axis=1)
y = df['risk level']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Train Logistic Regression model for multi-class classification
model = LogisticRegression(max_iter=1000, multi_class='multinomial',
solver='lbfgs')
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print("Classification Report:")
#print(classification_report(y_test, y_pred,
target_names=target_encoder.classes_))

# Confusion matrix heatmap
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=target_encoder.classes_, yticklabels=target_encoder.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
# Step 1: Import necessary libraries
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
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

```
# Step 2: Load the dataset
# Upload your 'dataset.csv' file using the file upload tool in Colab
from google.colab import files
uploaded = files.upload()

# Read the uploaded file
df = pd.read_csv('dataset.csv')

# Step 3: Explore the data
print(df.head())
print(df.info())
print(df['risk level'].value_counts())

# Step 4: Preprocessing
# Handle categorical variables if any
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    if col != 'risk level':
        df[col] = LabelEncoder().fit_transform(df[col])

# Encode target variable
df['risk level'] = LabelEncoder().fit_transform(df['risk level'])

# Split features and target
X = df.drop('risk level', axis=1)
y = df['risk level']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Step 6: Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 7: Make predictions
y_pred = model.predict(X_test)

# Step 8: Evaluate the model
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
# Step 9: Feature Importance (Optional)
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=feature_names[indices])
plt.title('Feature Importance')
plt.show()
# Step 1: Install required packages (if not already installed)
!pip install tensorflow pandas scikit-learn --quiet

# Step 2: Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report

# Step 3: Upload dataset
from google.colab import files
uploaded = files.upload()  # Upload dataset.csv here

# Step 4: Load the dataset
df = pd.read_csv('dataset.csv')

# Step 5: Data preprocessing
# Encode target (risk level)
label_encoder = LabelEncoder()
df['risk level'] = label_encoder.fit_transform(df['risk level'])  #
Converts to 0, 1, 2...

# Separate features and target
X = df.drop('risk level', axis=1)
y = df['risk level']

# Optional: Check for and handle non-numeric features in X
X = pd.get_dummies(X)  # One-hot encode any categorical features

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Step 6: Build the model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(len(np.unique(y)), activation='softmax')  # Multi-class
classification
])

# Step 7: Compile and train
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=50, batch_size=16, validation_split=0.1,
verbose=1)

# Step 8: Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Accuracy: {accuracy:.2f}")

# Step 9: Classification report
y_pred = model.predict(X_test).argmax(axis=1)
#print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=label_encoder.classes_))
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load dataset
from google.colab import files
uploaded = files.upload()  # Upload 'dataset.csv'

df = pd.read_csv('dataset.csv')

# Step 3: Preprocessing
```

```
# Check for missing values
print("Missing values:\n", df.isnull().sum())

# Fill numeric missing values
df.fillna(df.mean(numeric_only=True), inplace=True)

# Encode categorical columns
for col in df.select_dtypes(include='object').columns:
    if col != 'risk level':  # Encode all object columns except target
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

# Encode the target separately
target_encoder = LabelEncoder()
df['risk level'] = target_encoder.fit_transform(df['risk level'])

# Step 4: Feature Selection
X = df.drop('risk level', axis=1)
y = df['risk level']

# Step 5: Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 6: Scale the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 7: Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 8: Predictions
y_pred = model.predict(X_test)

# Step 9: Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Step 10: Feature Importance
plt.figure(figsize=(10, 6))
sns.barplot(x=model.feature_importances_, y=X.columns)
plt.title('Feature Importance in Disease Risk Prediction')
```

```
plt.tight_layout()
plt.show()
# Step 1: Install Required Libraries
!pip install -q seaborn scikit-learn pandas matplotlib

# Step 2: Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam

# Step 3: Load the Dataset
file_path = '/content/dataset.csv'  # Make sure to upload dataset.csv to
your Colab session
df = pd.read_csv(file_path)

# Step 4: Preprocess the Data
# Encode categorical labels if needed
if df['risk level'].dtype == 'object':
    le = LabelEncoder()
    df['risk level'] = le.fit_transform(df['risk level'])

# Split features and label
X = df.drop('risk level', axis=1)
y = df['risk level']

# Normalize features
scaler = StandardScaler()
#X_scaled = scaler.fit_transform(X)

# Split into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Step 5: Build Deep Learning Model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    BatchNormalization(),
    Dropout(0.5),
```

```
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(len(np.unique(y)), activation='softmax')  # Use softmax for
multi-class classification
])

# Step 6: Compile the Model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Step 7: Train the Model
history = model.fit(X_train, y_train, epochs=20, batch_size=32,
validation_split=0.1)

# Step 8: Evaluate the Model
y_pred = np.argmax(model.predict(X_test), axis=1)
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Step 9: Plot Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Risk Level Prediction")
plt.show()
```

14. Future Scope

The potential for expanding this project is considerable, with multiple avenues for future development. One key direction involves broadening the dataset to include a wider variety of patient demographics, healthcare institutions, and geographic regions. This would help the model learn from more diverse patterns, making it more robust and applicable to real-world clinical settings across different populations. Additionally, future versions of the system could incorporate more advanced machine learning techniques, such as XGBoost, CatBoost, or deep learning models like neural networks, to capture complex, non-linear relationships within the data and enhance predictive performance.

Another critical aspect of future work lies in improving model transparency and trust through Explainable AI (XAI) techniques such as SHAP and LIME. These tools would allow healthcare professionals to understand the reasoning behind each prediction, which is essential for clinical decision-making and fostering confidence in AI systems. Integration with electronic health record (EHR) systems could further elevate the utility of the model by enabling real-time, automated *disease risk assessments during patient visits, offering immediate insights to clinicians at the point of care.*

Beyond technical enhancements, future efforts could focus on deploying the system in real-world healthcare environments through partnerships with hospitals, clinics, and health organizations. Such collaborations would allow for real-time testing, validation, and refinement of the model based on real patient data and clinician feedback. Over time, the system could evolve into a full-fledged clinical decision support tool, capable of assisting medical professionals in early diagnosis, risk stratification, and personalized treatment planning.

Moreover, attention must be given to ethical and legal considerations, including patient data privacy, algorithmic fairness, and compliance with regulations like HIPAA and GDPR. Ensuring transparency, accountability, and equity in model predictions will be crucial for broader adoption. In the long term, the integration of this AI-powered system into mobile health applications or wearable technologies could enable continuous health monitoring and early disease detection, especially in underserved or remote areas, further democratizing access to quality healthcare.

13. Team Members and Roles

1.M.vijitha – Data Collection & Preprocessing Lead
This team member was responsible for sourcing and organizing the patient data used in the model. Their responsibilities included data cleaning, handling missing values, normalizing input variables, and ensuring the dataset was ready for model training. They also performed exploratory data analysis (EDA) to uncover patterns and trends in the data, laying the foundation for effective modeling.

2. P. Swathi– Machine Learning & Model Development Lead
This member led the design and implementation of the AI models used for disease prediction. They evaluated various machine learning algorithms, selected the best-performing model (e.g., Logistic Regression, XGBoost, Neural Networks), and carried out training, testing, and hyperparameter tuning. They also validated the model's accuracy and ensured its robustness across different data subsets.

3.M.Abinaya– Model Evaluation
This member focused on interpreting the model's results and ensuring transparency through Explainable AI (XAI) techniques such as SHAP and LIME. They also conducted performance evaluation using metrics like accuracy, precision, recall, and F1-score.

4.R.Thulasi priya -Explainability&documentation

Additionally, they were responsible for documenting the project thoroughly, including the technical report, future scope, and preparing the final presentation.

Throughout the project, all team members participated in regular discussions, decision-making, and knowledge sharing to ensure alignment and collective progress toward the project goals.