**HEALTH CARE**

**1.Project Content**

This project provides a predictive model to classify healthcare test results (such as "Normal", "Abnormal", or "Inconclusive") based on patient and hospital data. It includes:

- A data preprocessing pipeline for cleaning and encoding categorical and numerical features.

- A RandomForest classification model trained on a healthcare dataset.

- A Gradio web-based user interface for real-time prediction from user inputs.

- Saving/loading mechanisms for the trained model and data transformers using Pickle.

- Scripts to evaluate model performance and generate classification reports.

**2. Project Code**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import pickle

import gradio as gr

# Load dataset

df = pd.read\_csv("health care.csv")

# Drop unnecessary columns

df = df.drop(columns=["Name", "Date of Admission", "Discharge Date", "Doctor", "Hospital"])

# Split into features and target

X = df.drop("Test Results", axis=1)

y = df["Test Results"]

# Identify categorical and numerical columns

categorical\_cols = X.select\_dtypes(include="object").columns.tolist()

numerical\_cols = X.select\_dtypes(include=["int64", "float64"]).columns.tolist()

# Encode categorical features

encoders = {col: LabelEncoder().fit(X[col]) for col in categorical\_cols}

for col in categorical\_cols:

X[col] = encoders[col].transform(X[col])

# Scale numerical features

scaler = StandardScaler()

X[numerical\_cols] = scaler.fit\_transform(X[numerical\_cols])

# Encode target labels

target\_encoder = LabelEncoder()

y\_encoded = target\_encoder.fit\_transform(y)

# Split dataset into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

# Train RandomForest classifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Predict and evaluate

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred, target\_names=target\_encoder.classes\_)

# Show some predictions vs actual values

comparison = pd.DataFrame({

'Actual': target\_encoder.inverse\_transform(y\_test),

'Predicted': target\_encoder.inverse\_transform(y\_pred)

})

print(comparison.head(10))

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Save model and transformers

with open("healthcare\_model.pkl", "wb") as f:

pickle.dump({

"model": model,

"features": X.columns.tolist(),

"encoders": encoders,

"scaler": scaler,

"target\_encoder": target\_encoder,

"numerical\_cols": numerical\_cols,

"categorical\_cols": categorical\_cols

}, f)

# Define prediction function for Gradio interface

def predict(age, gender, blood\_type, condition, insurance, bill, room, adm\_type, medication):

row = pd.DataFrame({

"Age": [age],

"Gender": [encoders["Gender"].transform([gender])[0]],

"Blood Type": [encoders["Blood Type"].transform([blood\_type])[0]],

"Medical Condition": [encoders["Medical Condition"].transform([condition])[0]],

"Insurance Provider": [encoders["Insurance Provider"].transform([insurance])[0]],

"Billing Amount": [bill],

"Room Number": [room],

"Admission Type": [encoders["Admission Type"].transform([adm\_type])[0]],

"Medication": [encoders["Medication"].transform([medication])[0]]

})

row[numerical\_cols] = scaler.transform(row[numerical\_cols])

result = model.predict(row)[0]

return target\_encoder.inverse\_transform([result])[0]

# Launch Gradio app

gr.Interface(

fn=predict,

inputs=[

gr.Number(label="Age"),

gr.Dropdown(choices=encoders["Gender"].classes\_.tolist(), label="Gender"),

gr.Dropdown(choices=encoders["Blood Type"].classes\_.tolist(), label="Blood Type"),

gr.Dropdown(choices=encoders["Medical Condition"].classes\_.tolist(), label="Medical Condition"),

gr.Dropdown(choices=encoders["Insurance Provider"].classes\_.tolist(), label="Insurance Provider"),

gr.Number(label="Billing Amount"),

gr.Number(label="Room Number"),

gr.Dropdown(choices=encoders["Admission Type"].classes\_.tolist(), label="Admission Type"),

gr.Dropdown(choices=encoders["Medication"].classes\_.tolist(), label="Medication")

],

outputs=gr.Text(label="Predicted Test Result"),

title="Healthcare Test Result Predictor"

).launch()

**3. Key Technologies**

* **Python 3**: The main programming language used for data processing, modeling, and interface building.
* **Pandas**: Data manipulation and analysis library used for reading and processing CSV data.
* **scikit-learn**: Machine learning library used for preprocessing, model training, and evaluation.
* **RandomForestClassifier**: Ensemble learning method used to classify healthcare test results.
* **LabelEncoder & StandardScaler**: For encoding categorical variables and scaling numerical features respectively.
* **Pickle**: For saving and loading the trained model and preprocessing tools.
* **Gradio**: A library to create a user-friendly web interface to interact with the trained model for real-time predictions.

**4. Description**

This project addresses the challenge of predicting healthcare test results based on patient demographics, medical conditions, and hospital-related data. The goal is to classify results into categories like "Normal", "Abnormal", or "Inconclusive" to support medical decision-making.

**Dataset and Features**

The dataset contains patient data such as Age, Gender, Blood Type, Medical Condition, Insurance Provider, Billing Amount, Room Number, Admission Type, and Medication, alongside the Test Results. The following preprocessing steps are applied:

* **Dropping irrelevant columns**: Columns such as patient names and admission/discharge dates were removed to avoid data leakage.
* **Encoding categorical features**: Label encoding converts categories into numeric codes for machine learning.
* **Feature scaling**: Numerical features are standardized to improve model performance.
* **Label encoding target**: The test result categories are encoded numerically for classification.

**Model and Training**

A RandomForestClassifier is trained on 80% of the dataset. This ensemble method is chosen for its robustness and ability to handle mixed data types. The model is then evaluated on the test set.

**Deployment with Gradio**

To make the model accessible, a Gradio interface is developed. This allows users to input patient details through a web app, which outputs the predicted test result instantly.

**5. Output**

**Model Evaluation**

* The accuracy achieved on the test data is approximately **50%** (depending on dataset balance and quality).
* The classification report details precision, recall, and F1-score per class, highlighting model strengths and areas needing improvement.
* A comparison table shows the actual versus predicted test results for the first 10 test samples:

| **Actual** | **Predicted** |
| --- | --- |
| Normal | Normal |
| Abnormal | Abnormal |
| Inconclusive | Abnormal |
| Inconclusive | Normal |
|  |  |

**User Interface**

The Gradio app accepts:

* Numerical inputs: Age, Billing Amount, Room Number.
* Dropdown selections for categorical data: Gender, Blood Type, Medical Condition, Insurance Provider, Admission Type, and Medication.

Upon submitting inputs, the predicted test result is displayed, helping medical staff or patients quickly assess likely outcomes.

**6. Further Research**

Several avenues exist to extend and improve this project:

* **Dataset Enrichment**: Obtain larger, more diverse healthcare datasets to improve model generalization and reduce bias.
* **Model Optimization**: Experiment with hyperparameter tuning, feature engineering, or alternative algorithms (e.g., XGBoost, Neural Networks).
* **Handle Class Imbalance**: If the dataset is imbalanced (e.g., fewer "Inconclusive" cases), techniques like SMOTE or weighted loss functions can improve minority class prediction.
* **Explainability**: Integrate model interpretability tools such as SHAP or LIME to explain predictions and build trust with medical professionals.
* **Real-Time Integration**: Connect the predictor with hospital management systems or electronic health records (EHR) for automatic predictions.
* **Multi-modal Inputs**: Incorporate medical images or sensor data alongside tabular inputs for richer modeling.
* **User Experience**: Improve the Gradio interface with better UI/UX design and expand to mobile platforms.
* **Privacy & Security**: Implement data anonymization, secure data handling, and comply with healthcare regulations like HIPAA.