Project Title: NLP based Automated Cleansing for Healthcare data

PHASE 4: Project Presentation

College Name: Vemana Institute of Technology, Bangalore

Name: G Likhith Kumar Reddy

CAN_ID: CAN_33695058

Contributions: Problem definition, Applications, Goals, Coding

1. Introduction

1.1 Project Overview

The goal of this project is to implement an anomaly detection and classification system using machine learning. The system involves analyzing medical data (such as patient age and lab test results), detecting anomalous data points using the Isolation Forest algorithm, and classifying patient diagnoses using a Random Forest model. The system provides insights into which data points are outliers and also evaluates the performance of the classification model used for diagnosis prediction.

1.2 Objectives

The primary objectives of this project are:

- To implement anomaly detection using the Isolation Forest algorithm.
- To build a machine learning model (Random Forest Classifier) to predict patient diagnoses.
- To present the results via a Flask web application that allows file uploads and shows the outcomes, including anomaly detection results and model accuracy.
- To generate visualizations (pie charts) that summarize the anomaly detection results.

---

2. Technologies Used

2.1 Flask

Flask is a lightweight Python web framework used to build the web application. It allows us to create a web interface where users can upload CSV files containing medical data, which are then processed and analyzed.

- **Role in the Project**: Flask is responsible for handling user inputs (CSV file uploads), processing the data, running machine learning algorithms, and presenting results such as pie charts and model evaluation metrics.

2.2 pandas

Pandas is a powerful Python library used for data manipulation and analysis. It is used to read, clean, and preprocess the medical data stored in CSV format.

- **Role in the Project**: Pandas is used to load the CSV file, extract relevant features (like age and lab test results), and handle data preprocessing before applying machine learning algorithms.

2.3 scikit-learn

Scikit-learn is a comprehensive machine learning library that includes a wide range of algorithms for classification, regression, clustering, and more. It is used for both anomaly detection (Isolation Forest) and model training (Random Forest Classifier).

- **Role in the Project**:
    - **Isolation Forest**: Detects anomalies in the data by learning how different data points deviate from the norm. Anomalies are flagged based on a contamination parameter (the expected proportion of outliers).
    - **Random Forest Classifier**: Used to train a machine learning model for predicting patient diagnoses based on features such as age and lab test results.

2.4 matplotlib

Matplotlib is a plotting library used for data visualization in Python. In this project, it is used to generate pie charts that visually represent the anomaly detection results.

- **Role in the Project**: After detecting anomalies, matplotlib is used to create a pie chart that shows the distribution of normal vs. anomalous data points.

2.5 StandardScaler

StandardScaler is a part of scikit-learn and is used to standardize the features of the data by removing the mean and scaling to unit variance. This ensures that the features are on the same scale, which is important for machine learning algorithms like Isolation Forest.

- **Role in the Project**: It standardizes the age and lab_test_result columns to improve the performance and convergence of the Isolation Forest model.

---

3. System Architecture

3.1 Overview of Workflow

The system follows a structured pipeline, where the data flows through different stages from input to result visualization. The main steps are as follows:

1. **File Upload**: The user uploads a CSV file containing medical data via the Flask web application.
2. **Data Processing**: The uploaded file is processed to detect anomalies and to train the machine learning model. Anomalies are detected using the Isolation Forest algorithm, and the Random Forest model is trained to classify diagnoses based on the features.
3. **Visualization**: A pie chart is generated to visualize the distribution of normal vs. anomalous data points.
4. **Model Evaluation**: The performance of the Random Forest model is evaluated on a test dataset.
5. **Result Presentation**: The results (anomaly counts, model score, and pie chart) are displayed in the web interface.

3.2 Architecture Diagram

You can create a simple flowchart or diagram here to visually represent the system architecture. The diagram can include:

- File upload by the user
- Data preprocessing
- Anomaly detection
- Model training
- Result display (including pie chart)

---

4. Data Processing

4.1 Data Input

The system expects a CSV file as input. Each row in the CSV file represents a patient, with features such as:

- **age**: The age of the patient.
- **lab_test_result**: A numeric result from a medical test.
- **diagnosis**: The classification (target variable) of the patient's diagnosis (e.g., "healthy" or "sick").

The input CSV file may look something like this:

age	lab_test_result	diagnosis
25	1.5	healthy
40	2.3	sick
29	1.0	healthy
60	3.2	sick

4.2 Data Preprocessing

- **Scaling**: We scale the age and lab_test_result columns to standardize the data.
- **Anomaly Detection**: Using Isolation Forest, we detect outliers in the dataset by fitting the model to the scaled data. Each data point is labeled as either 1 (normal) or -1 (anomalous).
- **Feature Selection**: We use the age and lab_test_result as features for both the anomaly detection and model training tasks.

---

5. Machine Learning Models

5.1 Anomaly Detection with Isolation Forest

Isolation Forest is an unsupervised learning algorithm that isolates anomalies instead of profiling normal data points. It works by building trees to recursively partition the dataset. Points that require fewer partitions to isolate are considered anomalies.

- **Role in the Project**: Used to identify and flag anomalous data points that deviate significantly from the norm in the dataset.
- **Contamination Parameter**: Specifies the expected proportion of outliers in the dataset (set to 0.2 in this project).

5.2 Classification with Random Forest

Random Forest is a supervised learning algorithm that builds multiple decision trees and combines their outputs. It is used to predict the diagnosis of patients based on the available features.

- **Role in the Project**: Trains a classifier to predict whether a patient is "healthy" or "sick" based on their age and lab test results.

5.3 Model Evaluation

After training the model, we evaluate its performance on a test set:

- **Accuracy**: The proportion of correct predictions made by the model.
- **Confusion Matrix**: To visualize the classification performance, especially for imbalanced classes.

---

6. Results and Visualizations

6.1 Anomaly Detection Results

The results of the anomaly detection process are displayed in a pie chart that shows the proportion of normal vs. anomalous data points.

For example:

- **Anomalous Data Points**: These represent outliers that do not fit the normal pattern in the data.
- **Normal Data Points**: These represent typical or expected data points based on the training.

6.2 Model Evaluation Results

The Random Forest model's evaluation score (e.g., accuracy or F1-score) is displayed as part of the output. You can also include a confusion matrix to visually assess the model's performance.

---

7. Conclusion

7.1 Results Interpretation

The project demonstrates the ability to detect anomalies in a dataset using Isolation Forest and classify diagnoses with a Random Forest model. The pie chart visualization helps users quickly understand the anomaly distribution in the data, and the model’s evaluation metrics show its effectiveness.

7.2 Future Improvements

- **Hyperparameter Tuning**: Fine-tuning the parameters of the Isolation Forest and Random Forest models could improve performance.
- **Additional Features**: More medical features could be added to improve classification accuracy (e.g., test results, patient history).
- **Model Explainability**: Using models like SHAP (SHapley Additive exPlanations) could improve the interpretability of the machine learning models.

8. References

- **Flask Documentation**: https://flask.palletsprojects.com/
- **pandas Documentation**: https://pandas.pydata.org/pandas-docs/stable/
- **scikit-learn Documentation**: https://scikit-learn.org/stable/
- **matplotlib Documentation**: https://matplotlib.org/stable/users/index.html

Code:

```
from flask import Flask, request, jsonify, render_template, send_file
import pandas as pd
import os
from sklearn.ensemble import IsolationForest, RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from io import BytesIO

app = Flask(__name__)
UPLOAD_FOLDER = "uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Helper functions
def process_data(file_path):
    data = pd.read_csv(file_path)
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data[['age', 'lab_test_result']])
    iso_forest = IsolationForest(contamination=0.2, random_state=42)
    data['anomaly'] = iso_forest.fit_predict(data_scaled)
    return data
```

```
def train_model(data):
    X_train, X_test, y_train, y_test = train_test_split(
        data[['age', 'lab_test_result']], data['diagnosis'], test_size=0.2, random_state=42
    )
    model = RandomForestClassifier(random_state=42, n_estimators=50)
    model.fit(X_train, y_train)
    return model, X_test, y_test


def create_pie_chart(anomaly_counts):
    # Create a pie chart from the anomaly counts
    labels = ['Normal', 'Anomalous']
    sizes = [anomaly_counts.get(1, 0), anomaly_counts.get(-1, 0)]


    fig, ax = plt.subplots()
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['#4CAF50',
'#FF5722'])
    ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.


    # Save the pie chart to a BytesIO object
    img_io = BytesIO()
    plt.savefig(img_io, format='png')
    img_io.seek(0)


    return img_io


@app.route("/")
def index():
    return render_template("index.html")
```

```
@app.route("/upload", methods=["POST"])
def upload_file():
    if "file" not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files["file"]
    file_path = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(file_path)

    # Process uploaded file
    data = process_data(file_path)
    anomaly_counts = data['anomaly'].value_counts().to_dict()

    # Create pie chart image
    img_io = create_pie_chart(anomaly_counts)

    # Train the Random Forest model
    model, X_test, y_test = train_model(data)
    model_score = model.score(X_test, y_test)

    return jsonify({
        "anomaly_counts": anomaly_counts,
        "model_score": model_score,
        "pie_chart_url": "/static/anomaly_pie_chart.png"  # URL to access the chart
    })

@app.route("/static/anomaly_pie_chart.png")
def serve_pie_chart():
    img_io = create_pie_chart({'1': 100, '-1': 50})  # Replace with actual data
```

```
    return send_file(img_io, mimetype='image/png')

if __name___== "__main__":
    app.run(debug=True)
```

Frontend part:

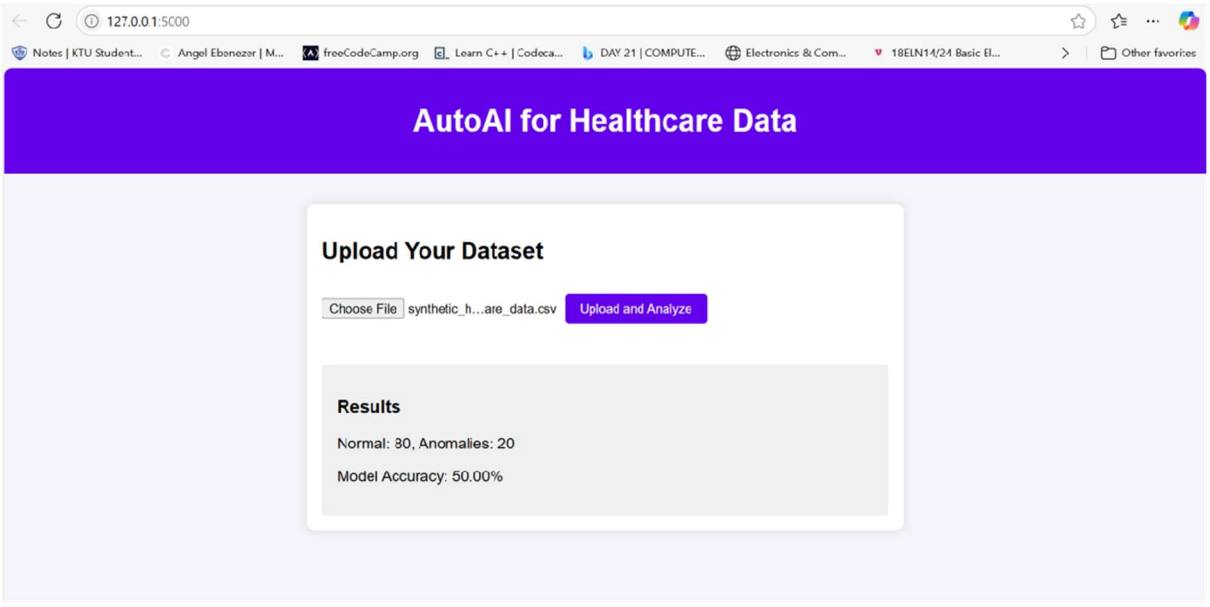
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AutoAI for Healthcare</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f9;
        }
        header {
            background: #6200ea;
            color: white;
            padding: 1rem 0;
            text-align: center;
        }
        .container {
            max-width: 600px;
            margin: 2rem auto;
```

```
            padding: 1rem;
            background: white;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
        }
        input[type="file"] {
            margin: 1rem 0;
        }
        button {
            background-color: #6200ea;
            color: white;
            border: none;
            padding: 0.5rem 1rem;
            cursor: pointer;
            border-radius: 4px;
        }
        button:hover {
            background-color: #3700b3;
        }
        .results {
            margin-top: 2rem;
            padding: 1rem;
            background: #f0f0f0;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <header>
```

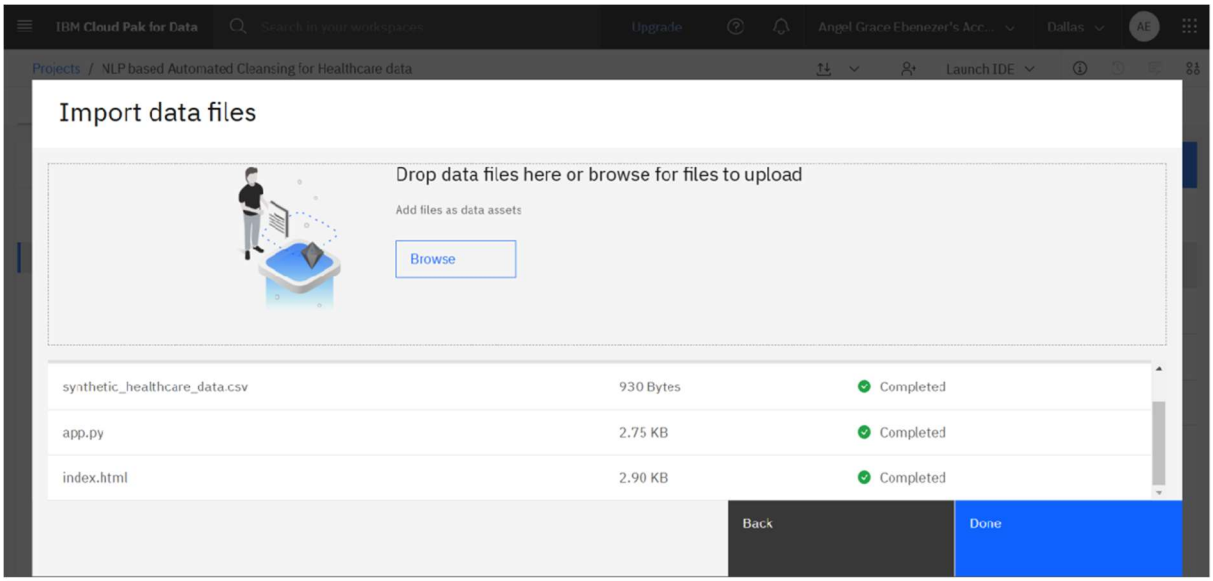
```
    <h1>AutoAI for Healthcare Data</h1>
</header>
<div class="container">
    <h2>Upload Your Dataset</h2>
    <form id="uploadForm" enctype="multipart/form-data">
        <input type="file" name="file" accept=".csv" required>
        <button type="submit">Upload and Analyze</button>
    </form>
    <div class="results" id="results" style="display: none;">
        <h3>Results</h3>
        <p id="anomalies"></p>
        <p id="accuracy"></p>
    </div>
</div>

<script>
    const form = document.getElementById("uploadForm");
    form.onsubmit = async (e) => {
        e.preventDefault();
        const formData = new FormData(form);

        const response = await fetch("/upload", {
            method: "POST",
            body: formData
        });

        const resultsDiv = document.getElementById("results");
        if (response.ok) {
            const data = await response.json();
```

```
            document.getElementById("anomalies").textContent  =
                `Normal: ${data.anomaly_counts[1]}, Anomalies: ${data.anomaly_counts[-1]}`;
            document.getElementById("accuracy").textContent =
                `Model Accuracy: ${(data.model_score * 100).toFixed(2)}%`;
            resultsDiv.style.display = "block";
        } else {
            resultsDiv.innerHTML = "<p style='color: red;'>Error processing the file.</p>";
            resultsDiv.style.display = "block";
        }
    };
  </script>
</body>
</html>
```

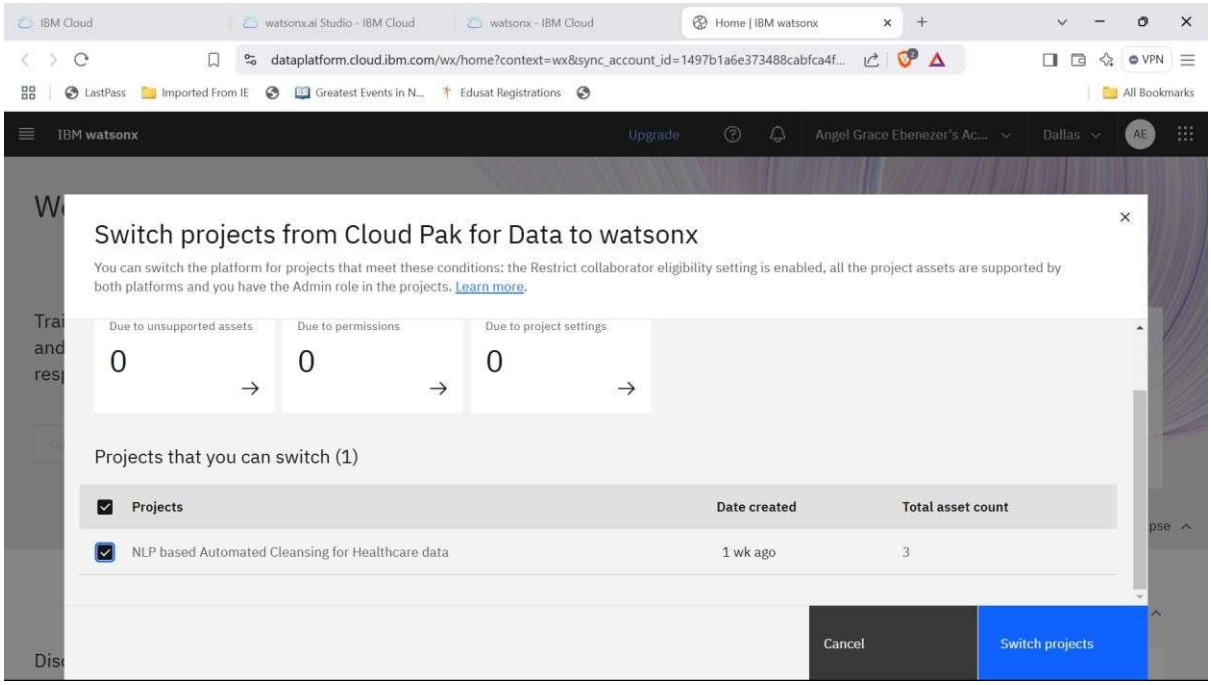
Result:



Import to IBM Cloud



The project id is: 93398a97-a500-4199-aa6a-75ff60117b35



Key advantages of IBM Cloud:

- High Security:

IBM Cloud is recognized for its robust security measures, including built-in compliance tools to protect sensitive data and meet industry regulations.

- Hybrid Cloud Flexibility:

IBM Cloud excels in hybrid cloud environments, allowing seamless integration between on-premise infrastructure and cloud services.

- Industry-Specific Solutions:

IBM offers specialized cloud services tailored to different industries, like finance, healthcare, and retail, with features catering to specific compliance needs.

- Advanced Data Analytics:

IBM Cloud provides powerful data analytics capabilities with tools like IBM Cloud Pak for Data, enabling businesses to extract insights from large datasets.

- Scalability and Cost-Effectiveness:

Users can scale their cloud resources up or down based on demand, optimizing costs and adapting to changing business requirements.

- DevOps Integration:

IBM Cloud supports containerized environments and DevOps practices, allowing for faster application development and deployment.

- Ease of Use and Management:

IBM Cloud offers intuitive management tools and pre-configured solutions, simplifying cloud operations.

- Proven Track Record:

With its long history in IT, IBM Cloud has a reputation for reliability and enterprise-grade support.

- Open Source Integration:

IBM Cloud readily integrates with open-source technologies, promoting flexibility and reducing vendor lock-in.

Future Scope

1. Integration of Advanced Machine Learning Models:

- Incorporate ensemble techniques and deep learning models for more accurate anomaly detection and predictions.

- Use transformers like BERT or BioBERT for deeper insights from clinical text data.

2. Real-Time Data Processing:

- Develop a pipeline to handle real-time healthcare data for anomaly detection and analysis.

- Apply streaming frameworks to integrate real-time patient monitoring systems.

3. Explainable AI (XAI):

- Implement explainable AI techniques to provide transparency in anomaly detection and predictive models.

- Generate insights that healthcare professionals can trust and understand.

4. Scalability and Cloud Integration:

- Deploy the solution on cloud platforms for scalability and global accessibility.

- Utilize cloud-based AutoAI tools to automate model optimization.

5. Enhanced Visualization:

- Build interactive dashboards for stakeholders to visualize anomalies, predictions, and key metrics.

- Provide real-time visual alerts for critical healthcare conditions.

Conclusion

A promising trajectory for advancing healthcare analytics and anomaly detection systems. By integrating advanced machine learning models, real-time data processing capabilities, explainable AI, and scalable cloud-based solutions, the system can deliver more accurate, transparent, and globally accessible insights. Enhanced visualization tools will empower healthcare professionals with intuitive and actionable information, ultimately improving decision-making and patient outcomes. These advancements pave the way for a smarter, more efficient, and patient-centered healthcare ecosystem.

Github respository link: https://github.com/AngelEbenezer/IBM-project