Machine Learning for Sustainable Development Goal 3: "Good Health and Well-being"

1. Introduction

Project Objective:

- This project aims to develop an AI-driven solution to identify underserved areas by analyzing socioeconomic and healthcare infrastructure data. By pinpointing areas with limited healthcare resources, we can recommend resource allocation strategies to improve accessibility, ensuring equitable healthcare access and promoting overall community well-being.

- We can solve this problem by implementing logistic regression model to identify and classify underserved areas based on healthcare accessibility data.

- Motivation: "It is health that is real wealth and not pieces of gold and silver." – *Mahatma Gandhi*

- Mahatma Gandhi's words remind us that true wealth lies in health, not material riches. Good health enables us to pursue goals and contribute meaningfully to society. By prioritizing well-being, individuals and communities build resilience and harmony, aligning with Sustainable Development Goal (SDG) 3 to promote personal and societal prosperity.

- So the motivation to implement this project is to Access to healthcare is essential for well-being, and identifying underserved areas is crucial for fair resource distribution.

2. Data Collection

The dataset used in this project was created using ChatGPT to simulate healthcare accessibility data across various regions, including features like population density, distance to healthcare facilities, income levels, and doctor availability.

Dataset Description:

- Information on healthcare infrastructure (e.g., number of hospitals and clinics)

-Demographic and socioeconomic factors (e.g., population density, median income, unemployment rate)indices for healthcare access and quality.

- Size: 500 rows by 13 columns

3. Exploratory Data Analysis (EDA):

This project aims to explore and analyze a dataset through a series of comprehensive statistical and visual techniques. Key tasks include examining the dataset's structure, calculating summary statistics and visualizing the distribution of numerical features. Additionally, pairwise relationships will be assessed using pairplots, while a correlation heatmap will highlight relationships between variables. Finally, boxplots will be used to identify potential outliers, providing insights into the data's underlying patterns and quality.

4. Data Preprocessing

Encode categorical columns: Encoding categorical columns converts text data into numbers, enabling machine learning models to process and learn from categorical information.

- Encoding Categorical Variables: [LabelEncoder] for converts categories (text labels) into numbers.

- Feature Scaling: [StandardScaler] for Adjusts numerical values so they have a mean of 0 and a standard deviation of 1.

5. Machine Learning Model Selection

- Logistic Regression: It selected due to its simplicity and interpretability, fitting well for binary classification tasks like identifying underserved areas. This model serves as a baseline for this active learning framework.
-Why Scikit-Learn: Easy implementation, variety of algorithms, and effective performance metrics.
Evaluation Metric: Accuracy, Precision, Recall, and F1-Score due to the critical nature of accurately identifying contamination.

6. Model Implementation

Data Splitting: Split dataset into 80% training and 20% testing sets using `train_test_split` from Scikit-Learn model.

- Data Augmentation: The uncertain samples were labeled and added to the training set, after which the model was retrained to improve its predictive power.

- Evaluation: Model performance was tracked after each iteration on a separate test set to monitor improvements.

Code Example:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(random_state=42)

model.fit(X_train, y_train)

from sklearn.metrics import accuracy_score, classification_report# Initial Evaluation

y_pred = model.predict(X_test)

print("Initial Accuracy:", accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

Active learning loop to iteratively improve model by querying uncertain samples.

```
def active_learning(model, X_train, y_train, X_pool, y_pool, X_test, y_test, n_iterations=10, batch_size=10):

    # Ensure that X_pool and y_pool are DataFrames or Series

    X_train = pd.DataFrame(X_train)

    y_train = pd.Series(y_train)

    X_pool = pd.DataFrame(X_pool)

    y_pool = pd.Series(y_pool)

    X_test = pd.DataFrame(X_test)

    y_test = pd.Series(y_test)


    for i in range(n_iterations):

        # Predict probabilities on the pool set

        probs = model.predict_proba(X_pool)[:, 1]

        uncertainty = np.abs(probs - 0.5)  # uncertainty for binary classification

        # Select samples with highest uncertainty

        uncertain_indices = uncertainty.argsort()[:batch_size]

        X_selected, y_selected = X_pool.iloc[uncertain_indices], y_pool.iloc[uncertain_indices]

        # Add these to the training set

        X_train = pd.concat([X_train, X_selected], ignore_index=True)

        y_train = pd.concat([y_train, y_selected], ignore_index=True)
        # Remove selected samples from pool
        X_pool = X_pool.drop(X_selected.index).reset_index(drop=True)
        y_pool = y_pool.drop(y_selected.index).reset_index(drop=True)
        model.fit(X_train, y_train)# Retrain model
        # Evaluate on test set
        y_test_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_test_pred)
        print(f"Iteration {i + 1}, Test Accuracy: {accuracy}")
        print(confusion_matrix(y_test, y_test_pred))
 return model
# Split remaining data as pool for active learning
X_pool, y_pool = X_train[100:], y_train[100:]
X_train, y_train = X_train[:100], y_train[:100]
# Re-initialize model for active learning process
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
from sklearn.metrics import accuracy_score, classification_report
y_final_pred = model.predict(X_test)
print("Final Accuracy:", accuracy_score(y_test, y_final_pred))
print(classification_report(y_test, y_final_pred))
```

7. Results and Evaluation

- The active learning approach led to incremental improvements in the model's ability to identify underserved areas. Evaluation metrics, including accuracy and F1 score, indicated that the model became more accurate with each iteration, achieving better results with fewer labeled samples.

```
Final Accuracy: 0.93
              precision    recall  f1-score   support

           0       0.97      0.93      0.95        74
           1       0.83      0.92      0.87        26

    accuracy                           0.93       100
   macro avg       0.90      0.93      0.91       100
weighted avg       0.93      0.93      0.93       100
```

8. Conclusion and Future Work

- This project demonstrates the potential of using active learning to optimize model performance in resource-constrained environments. Future work could involve testing with different sampling strategies, integrating more comprehensive data sources, or applying more complex models to enhance predictive accuracy.

9. References

- Using ChatGPT(make own created Dataset)
- Scikit-Learn Documentation and Google colab application