Predicting Disease Outcomes in Diabetes Patients using Machine Learning

Team Members:

Athish Sivakumaran - 71762133006

Rithis Naha – 71762133037

Predicting Disease Outcomes in Diabetes Patients using Machine Learning

Abstract:

The rising prevalence of diabetes poses significant challenges in managing patient care and predicting disease progression. This project focuses on developing a machine learning model to predict disease outcomes in diabetes patients, integrating Exploratory Data Analysis (EDA), Logistic Regression, and various advanced algorithms including classification through Artificial Neural Networks (ANN), clustering via K-Means, and utilization of neural networks such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

The initial phase involves thorough exploratory data analysis to understand the patterns within the dataset, emphasizing factors such as patient demographics, lifestyle choices, and historical medical data. Visualizations and statistical analyses are employed to uncover correlations and dependencies that influence disease outcomes.

Following the EDA, Logistic Regression is implemented to model the likelihood of specific disease outcomes based on relevant patient features. This serves as a foundational model, providing interpretability and insights into the impact of individual factors on disease progression.

To capture complex relationships and patterns inherent in the data, the project explores the use of Artificial Neural Networks (ANN) for classification. These networks leverage deep learning to discern intricate dependencies, contributing to improved accuracy in predicting disease outcomes.

For patient stratification and identifying groups with similar health characteristics, K-Means clustering is applied. This clustering algorithm aids in segmenting patients with comparable health profiles, enabling targeted interventions and personalized treatment plans.

Additionally, Convolutional Neural Networks (CNN) are employed for their ability to process image-like medical data, such as X-rays or scans, providing insights into the impact of visual information on disease prediction. Recurrent Neural Networks (RNN) are utilized to model temporal dependencies in patients' health records, capturing the evolving nature of their health over time.

The amalgamation of these algorithms aims to create a robust predictive model for disease outcomes in diabetes patients. Such a model can assist healthcare providers in early intervention, personalized treatment planning, and overall improvement in patient care, ultimately contributing to better management of diabetes and its associated complications.

Dataset:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, LSTM
```

```
[ ]
```

```
[33] data = pd.read_csv('diabetes.csv')
     data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Data Training:

```
[39] X = data.drop('Outcome', axis=1)
     y = data['Outcome']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Models:

1. **Logistic Regression :**

```
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)
logreg_predictions = logreg_model.predict(X_test)
```

Accuracy:

```
[38] log=np.array(logreg_predictions)
     error=np.array(abs(log-np.array(y_test)))

     round(1-(X_test.shape[1]/error.sum()),1)

     0.8

[46] X_train1=np.array(X_train)
     X_train1

     array([[  2.   ,  84.   ,   0.   , ...,   0.   ,   0.304,  21.   ],
            [  9.   , 112.   ,  82.   , ...,  28.2  ,   1.282,  50.   ],
            [  1.   , 139.   ,  46.   , ...,  28.7  ,   0.654,  22.   ],
            ...,
            [ 10.   , 101.   ,  86.   , ...,  45.6  ,   1.136,  38.   ],
            [  0.   , 141.   ,   0.   , ...,  42.4  ,   0.205,  29.   ],
            [  0.   , 125.   ,  96.   , ...,  22.5  ,   0.262,  21.   ]])
```

2. ANN:

```
[66] import tensorflow as tf
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.metrics import accuracy_score, classification_report

     X = data.drop('Outcome', axis=1)
     y = data['Outcome']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)

     model = tf.keras.Sequential([
         tf.keras.layers.Dense(8, activation='relu', input_shape=(X_train.shape[1],)),
         tf.keras.layers.Dense(1, activation='sigmoid')
     ])

     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

     model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=2)

     y_pred = model.predict(X_test)
     y_pred = (y_pred > 0.5).astype(int)
     accuracy = accuracy_score(y_test, y_pred)
     print(f'Accuracy: {accuracy:.2f}')
     print('\nClassification Report:')
     print(classification_report(y_test, y_pred))
```

```
5/5 [==============================] - 0s 3ms/step
Accuracy: 0.74

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.78      0.79        99
           1       0.63      0.67      0.65        55

    accuracy                           0.74       154
   macro avg       0.72      0.73      0.72       154
weighted avg       0.75      0.74      0.74       154
```

3. Random Forest:

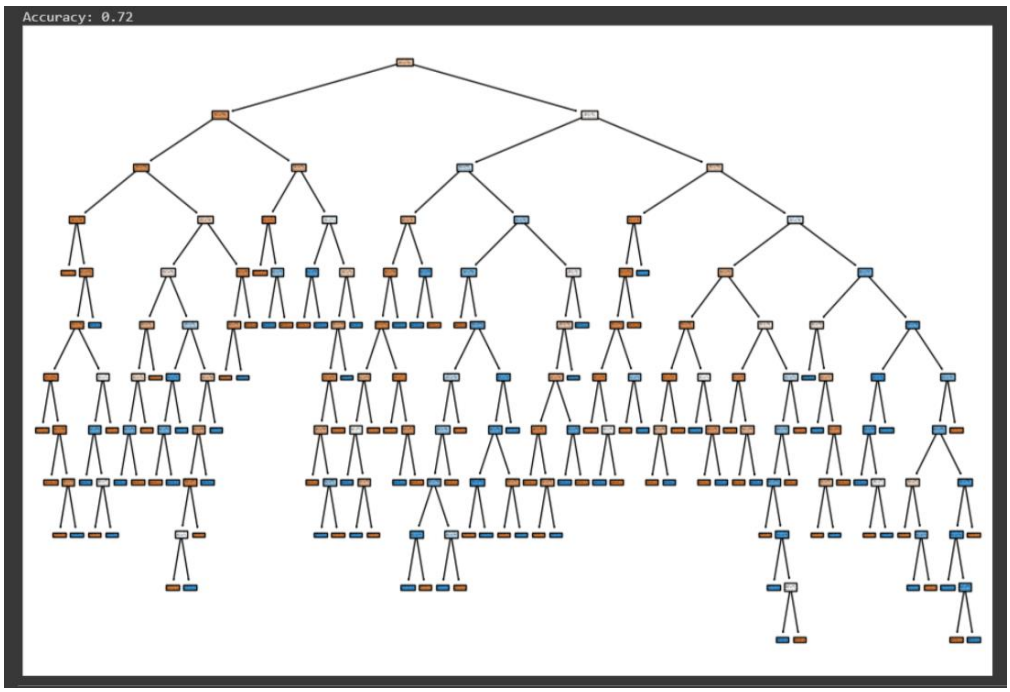
```
X = data.drop('Outcome', axis=1)
y = data['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

rf_classifier.fit(X_train, y_train)

accuracy = rf_classifier.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")

plt.figure(figsize=(12, 8))
tree.plot_tree(rf_classifier.estimators_[0], filled=True)
plt.show()
```



Comparison:

```
plt.bar(["LOGISTIC REGRESSION","ANN","RANDOM FOREST"],[0.8,0.74,0.72])
plt.title("CLASSIFICATION ACCURACY COMPARISON")
plt.show();
```