Name : Mohamed Nawas Raza Mohamed

Student Number: 24148501

Link to Code : https://colab.research.google.com/drive/1Yj90_FOYbaWCwvToBrzwAzkpYIlgM781?usp=sharing

Diabetes Prediction Using Deep Learning

Diabetes is a global health concern, and early detection plays a crucial role in preventing complications. This study applies deep learning with Optuna hyperparameter tuning to predict diabetes from medical records. We implemented data preprocessing, feature selection, and model evaluation, selecting the most precise model.

Data Processing & Feature Selection

Key preprocessing steps:

- Handling Missing Values: Zero values in A2–A6 were replaced with median values to maintain consistency.
- Feature Scaling: StandardScaler was applied to normalize numerical data.
- Feature Selection: A correlation matrix helped remove redundant features.
- Class Imbalance Handling: SMOTE was applied to balance the dataset.

```
# Feature Correlation Heatmap
sns.heatmap(train_df.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Matrix")
plt.show()
```

Model Comparison & Model Selection & Optuna Hyperparameter Optimization & Justification

We trained multiple models and optimized a DNN model using Optuna. The best model was chosen based on precision, reducing false positives.

	training_accuracy	validation_accuracy	train_val_acc_diff	precision	recall	f1_score	confusion_matrix
DNN	0.724719	0.708955	0.015764	0.769231	0.217391	0.338983	[[85, 3], [36, 10]]
decision_tree	1.000000	0.753731	0.246269	0.606557	0.804348	0.691589	[[64, 24], [9, 37]]
xgboost	1.000000	0.738806	0.261194	0.607843	0.673913	0.639175	[[68, 20], [15, 31]]
lightgbm	1.000000	0.768657	0.231343	0.641509	0.739130	0.686869	[[69, 19], [12, 34]]
naive_bayes	0.769663	0.686567	0.083096	0.535714	0.652174	0.588235	[[62, 26], [16, 30]]
svm	0.829588	0.708955	0.120633	0.563636	0.673913	0.613861	[[64, 24], [15, 31]]

Why DNN?

DNN had the highest precision (76.9%), ensuring fewer false positives. It also performed better than Decision Trees and XGBoost, which tend to overfit.

Best DNN Model Found by Optuna

Optuna optimized learning rate, dropout, L2 regularization, neurons, and activation functions. The best-performing configuration:

Hyperparameter	Best Value (From Optuna)
Learning Rate	**0.0048**
LeakyReLU Alpha	**0.0031**
L2 Regularization	**0.0044**
Neurons (Layer 1)	**128**
Neurons (Layer 2)	**96**
Dropout (Layer 1)	**0.0986**
Dropout (Layer 2)	**0.0767**

This optimized DNN model achieved the highest precision (76.9%), making it the best choice for minimizing false positives.

Best Model Implementation & Training

The DNN consists of two hidden layers with LeakyReLU activation, BatchNormalization, and Dropout to improve generalization.

```
# Define and Compile Optimized DNN Model
model = Sequential([
    Dense(128, activation=LeakyReLU(0.0031), kernel_regularizer=regularizers.l2(0.0044), input_shape=(8,)),
    BatchNormalization(), Dropout(0.0986),

    Dense(96, activation=LeakyReLU(0.0031), kernel_regularizer=regularizers.l2(0.0044)),
    BatchNormalization(), Dropout(0.0767),

    Dense(1, activation='sigmoid')
])
model.compile(optimizer=Adam(learning_rate=0.0048), loss='binary_crossentropy', metrics=['accuracy', Precision()])
```

Model Evaluation & Results

The trained model was evaluated using precision, recall, and F1-score.

```
# Training vs Validation Loss Curve
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss Curve')
plt.legend()
plt.show()
```

The final test results confirm DNN as the best model.

```
# Model Performance on Test Data
y_pred = (model.predict(X_test_scaled) > 0.5).astype(int)
print(f"Test Precision: {precision_score(y_test, y_pred):.4f}")
```

Conclusion

This study demonstrates that deep learning outperforms traditional models in diabetes prediction. The DNN model optimized using Optuna achieved the highest precision (76.9%), making it ideal for minimizing false positives. Future improvements include further hyperparameter tuning and real-world deployment.