Optimizing Early Diabetes Detection: A Comparative Analysis of Traditional Machine Learning Baseline and Enhanced Deep Learning Models

Author:
Muhammad Waqas Mandrey

Abstract

This project is dedicated to the development of a robust predictive model for diabetes by leveraging health records data. The endeavor confronts multifaceted challenges, including the intricate task of ensuring data quality, navigating the nuances of feature engineering, addressing class imbalance, and optimizing the design of a deep learning model. Through meticulous exploration and strategic solutions, the project aims not only to overcome these challenges but also to contribute valuable insights for the advancement of healthcare applications. The synergy between technology and healthcare is exemplified in this initiative, showcasing the perseverance required to unlock the full potential of data-driven disease prediction and management in the contemporary healthcare landscape.

Table of Contents

Introduction

The contemporary landscape of healthcare is undergoing a transformative shift, fueled by advancements in data science and machine learning. In this context, our project focuses on the profound task of leveraging health records data to develop a predictive model for diabetes. Diabetes, a prevalent and complex health condition affecting a substantial portion of the global population, demands nuanced approaches to diagnosis and management.

The challenges embedded in this endeavor are multifaceted and require careful consideration. One of the primary hurdles is the quality of health records data. Given the intricate nature of medical information, inconsistencies and noise within the dataset can significantly impact the reliability of any predictive model. Thus, a crucial initial step involves meticulous data cleaning to ensure accuracy and coherence in the collected information.

Feature engineering is another critical facet of the project. Transforming raw health data into meaningful features that contribute to the predictive power of the model necessitates not only technical expertise but also a deep understanding of the healthcare domain. The challenge lies in identifying and extracting relevant information that can enhance the model's accuracy and effectiveness.

Class imbalance is a common issue encountered in medical datasets, where one class (e.g., positive cases of diabetes) may be disproportionately represented compared to the other. Addressing this imbalance is crucial to prevent biases in the model, and techniques like oversampling or undersampling will be employed to create a more equitable representation of the data.

Furthermore, the project encompasses the complexities of data splitting, wherein the dataset is partitioned into training, validation, and testing sets. Striking the right balance in these splits is vital for training a model that generalizes well to unseen data.

As the project progresses into the realm of deep learning, additional challenges emerge. Hyperparameter tuning, involving the adjustment of various parameters to optimize model performance, requires a systematic and iterative approach. Designing an effective neural network, with considerations for input features, hidden layers, and output nodes, adds another layer of complexity.

In essence, this project navigates through the intricacies of healthcare data, tackling challenges from data quality to the design of advanced predictive models. By addressing these challenges head-on, we aim not only to develop a high-performance predictive model for diabetes but also to contribute valuable insights for future studies and potential deployment in real-world healthcare scenarios. This journey embodies the intersection of technology and healthcare, where overcoming challenges is a prerequisite for unlocking the full potential of data-driven disease prediction and management.

Tasks

⇒ Gather health records data relative to diabetes patients based on health features

⇒ Explore data gathered, identify unnecessary features

⇒ Clean the dataset; delete unnecessary features, impute missing values, ensure data consistency

⇒ Perform feature engineering on dataset; one-hot encoding on categorical features relative to the potential machine learning model

⇒ Summarize or describe characteristics of the dataset (ex. Mean, median, features count)

⇒ Perform data split for dataset per following ratio:

- Training (70%),
- Validation (15%),
- Testing (15%)

⇒ Design and train baseline model with training

⇒ Evaluate trained baseline model with validation data

⇒ Study & explore baseline trained model statistics

⇒ Preprocess data for deep learning:

- Standardize non-binary features in dataset,
- Perform data splits; train, validate, test

⇒ Research and Design appropriate deep learning model (FNN or MLP) including selecting input features, number of hidden layers, and the output node

⇒ Compiling deep learning model:

- Loss function (binary cross-entropy),
- Optimization Algorithm, Adaptive Moment Estimation (ADAM),
- Evaluation metrics (ex. accuracy, precision, recall)

⇒ Train the deep learning model with the training dataset

⇒ Evaluate model performance using validation dataset and metrics including but not limited to accuracy, precision, F1-Score, ROC Curve

⇒ Perform hyper tuning of the model to evaluate better performance of the model by adjusting number of hidden layers, batch size. Assess and prevent model from overfitting

⇒ Evaluate deep learning model performance using the testing data to understand model predicted output for unseen data

⇒ Conduct future work; deployment of model, future study

Challenges

Potential challenges are always expected and can occur in many aspects. For this deep learning model some potential challenges that may arise are:

1. **Data Quality**
   Quality of data specially health records data can be challenging due to many factors such as inconsistency or noisy data. These factors impact the learning of the model in a negative way. To avoid such discrepancies in the data, data cleaning is a very crucial step.

2. **Feature Engineering**
   Engineering old or new features in the data can be challenging. One of the key factors in feature engineering would be domain knowledge i.e. in this case having specialized knowledge related to health industry. Having in-depth knowledge would help understand the relative variables prior to transformation, and applying suitable data in the deep learning model to reach better accuracy.

3. **Class Imbalance**
   Imbalance can happen very often when working with medical records dataset. Often one case (Positive) can outweigh the other case (Negative) in the label column of the dataset. Techniques such as oversampling, undersampling can be implemented to handle imbalance in dataset.

4. **Hyperparameter Tuning**
   Hyperparameter tuning is one of the key challenges faced in machine learning models. In order to improve deep model accuracy score, parameters need to be adjusted very often in order to achieve the most accurate model for future predictions when performed with unseen data.

5. **Deep Learning Model designing**
   Building a machine learning model, specifically a neural network model can be very challenging when designing. A deep learning model requires various functionalities and configurations in order to provide more accurate results. Factors such as input features, number of hidden layers happen to play a significant role in the designing and accuracy of the deep learning model.

Potential challenges are one of the factors that play a role in the performance of the machine learning model. In order to have a high performance model, more challenges are generated that make the task difficult but are the path to achieving a high performance model. Challenges should always be addressed before deploying model for use in industries.

Data

1. Data Source & Collection

The dataset for this research was provided by Doctor Fenglong Ma, Assistant Professor at The Pennsylvania State University. The dataset contains health records related to diabetes patients. Each record in the dataset represents individual health records for anonymous patients.

2. Data Acquisition & Storage

The dataset is obtained in CSV format and stored in 'My Google Drive' for subsequent data processing and predictive analysis.

3. Dataset Characteristics

The dataset consists of a total of 17 features and 520 records, representing a moderately sized dataset.

4. Features Selection

From the dataset, 16 features were selected for inclusion in the research. The features are:

1. Age, numerical feature representing age of each patient,
2. Gender, binary categorical feature representing gender of each patient (Male/Female),
3. Fourteen further binary categorical features (Yes/No) including:
    a. Polyuria,
    b. Polydipsia,
    c. sudden weight loss,
    d. weakness,
    e. Polyphagia,
    f. Genital thrush,
    g. visual blurring,
    h. Itching,
    i. Irritability,
    j. delayed healing,
    k. partial paresis,
    l. muscle stiffness,
    m. Alopecia,
    n. Obesity

5. Label Selection

The target variable labeled as ‘class’ is a binary feature (Positive/Negative) and represents the outcome based on the input features. Positive and Negative represent the presence and absence of diabetes in the patient respectively.

6. Data Preprocessing

In order to prepare the data for machine learning and deep learning models, one-hot encoding has been applied to the 15 input features and the label feature ‘class’. One-hot encoding enables the data representation suitable for machine learning and deep learning model training.

7. Data Quality

Data quality checks have been performed on the dataset to identify and address missing values, data type errors, inconsistencies in order to ensure the reliability of the dataset.

8. Data Splitting

The dataset has been divided into training, testing, validation sets in splits of 70%, 15%, 15% respectively, best aligning with the training and testing of the machine learning and deep learning model.

9. Data Privacy

The dataset does not contain any patient-specific information due to being anonymous records, yet ethical considerations related to data privacy code were observed.

10. Software & Tools

Data Cleaning, Transformation, Processing, and further Machine Learning + Deep Learning model building are performed utilizing the following tools:

- Python, performing data cleaning, data transformation, model building,
- Pandas, utilized for data cleaning, transformation and extraction purposes,
- Scikit-learn, utilized for model evaluation and data split,
- Google Colab, utilized as the open-source platform for data processing and model building purposes.

Baseline

1. **Model Selection**
   For the baseline model, Logistic Regression has been chosen as a well-established and interpretable machine learning algorithm renowned for its effectiveness in binary classification tasks.

2. **Model Features**
   The baseline model employs a set of 16 selected features, encompassing both numerical and categorical attributes. These features include the patient's age, gender (Male/Female), and 14 binary (Yes/No) indicators denoting the presence or absence of specific health symptoms and conditions.

3. **Data Split**
   The dataset has been split into three subsets:
   - Training Set, comprising 70% data records, serving as the foundation for ML model training,
   - Validation Set, comprising 15% data records, playing a crucial role in assessing model performance and hyper-parameter tuning,
   - Testing Set, comprising 15% data records, the testing set is reserved as unseen data for the final evaluation of the ML model.

4. **Model Training**
   The baseline Logistic Regression model has undergone comprehensive training, with the primary focus on learning the relationships and patterns within the training data. This training process involves the optimization of model parameters to maximize prediction accuracy.

5. **Model Evaluation**
   Following metrics are utilized to assess the effectiveness of the baseline model:
   - Accuracy: The overall correctness of the model's predictions,
   - Precision: The model's ability to correctly identify true positives,
   - Recall: The proportion of actual positives correctly classified,
   - F1-Score: A measure of the model's balance between precision and recall,
   - AUC-ROC: Area Under the Receiver Operating Characteristic Curve, which quantifies the model's discriminatory power.

6. **Results and Discussion**
   Baseline model is prepared and is currently undergoing evaluation. The evaluation insights will serve as a benchmark for the deep learning model development. Next steps are to aim on developing a deep neural network with enhanced predictive accuracy rate.

Deep Learning Model

1. **Model Selection**
   For the enhanced deep learning model, a Feedforward Neural Network (FNN) architecture is chosen. FNNs, also known as Multilayer Perceptron (MLP), are well-suited for complex pattern recognition tasks, offering the capability to capture intricate relationships within data.

2. **Model Features**
   The deep learning model will utilize the same set of 16 features as the baseline model, encompassing both numerical and categorical attributes, providing a comprehensive representation of patient health characteristics.

3. **Data Preprocessing**
   Prior to training the deep learning model, comprehensive data preprocessing has been undertaken. Numerical features have undergone standardization, ensuring uniform scales across different attributes. This is a critical step to prevent certain features from dominating the learning process due to differences in their scales, leading to improved convergence during training. Categorical features have been subjected to one-hot encoding, facilitating the model's ability to interpret and learn from these variables effectively.

4. **Data Split**
   Similar to the baseline, the dataset has been split into training (70%), validation (15%), and testing (15%) sets. This meticulous partitioning ensures the model is trained on diverse data, validated for optimal performance, and tested on previously unseen instances.

5. **Model Architecture**
   The FNN architecture consists of an input layer corresponding to the 16 features, multiple hidden layers to capture intricate relationships, and an output layer for binary classification (diabetes or non-diabetes). The number of hidden layers and neurons per layer will be determined through an iterative optimization process.

6. **Compiling the Model**
   The model has been compiled with configurations including:
   - Loss Function: Binary Cross-Entropy, suitable for binary classification tasks.
   - Optimization Algorithm: Adaptive Moment Estimation (ADAM), known for its effectiveness in training deep neural networks.
   - Evaluation Metrics: A comprehensive set including accuracy, precision, recall, F1-Score, and AUC-ROC will be employed to assess the model's performance.

7. **Model Training**
   The FNN is undergoing training using the preprocessed training dataset, with a focus on adjusting weights and biases to learn intricate patterns and relationships within the data.

8. **Model Evaluation**
   After training, the model's performance will be rigorously evaluated using the validation dataset. Metrics such as accuracy, precision, recall, F1-Score, and AUC-ROC will be utilized to comprehensively assess the model's effectiveness.

9. **Hyperparameter Tuning**
   To enhance model performance, hyperparameter tuning will be conducted, adjusting parameters such as the number of hidden layers and batch size. This aims to prevent overfitting and optimize the model for accurate predictions on unseen data.

10. **Final Evaluation**
    The deep learning model's performance will be assessed using the testing dataset, providing insights into its predictive capabilities. Comparative analysis with the baseline model will offer valuable perspectives on the advancements achieved through the application of deep learning techniques.

Results

The analysis explores two distinct models, each embodying a unique approach to predictive modeling. The baseline Logistic Regression, a traditional stalwart, serves as a benchmark for its interpretability and effectiveness in binary classification. In contrast, the advanced Feedforward Neural Network (FNN) leverages deep learning techniques, showcasing potential for intricate pattern recognition.

These models stand at the crossroads of conventional and cutting-edge approaches in healthcare prediction. Logistic Regression provides a solid foundation with commendable performance, while the FNN, optimized through meticulous training, offers a glimpse into the future of predictive modeling with heightened accuracy and robust generalization. This dual-model analysis unfolds a narrative illustrating the evolving landscape where traditional methodologies intersect with the promises and challenges of state-of-the-art deep learning in healthcare data analysis.

Baseline Logistic Regression Results

Training Metrics:
- AUC: 0.986
- Accuracy: 0.945
- Recall: 0.963
- Precision: 0.946
- Specificity: 0.917

Validation Metrics:
- AUC: 0.943
- Accuracy: 0.897
- Recall: 0.896
- Precision: 0.935
- Specificity: 0.900

Test Metrics:
- AUC: 0.966
- Accuracy: 0.923
- Recall: 0.925
- Precision: 0.961
- Specificity: 0.920

Feature Importance:

Feature	Importance
Polydipsia_Yes	1.871165
Polyuria_Yes	1.623144
Irritability_Yes	1.152463
Genital thrush_Yes	0.932349
Partial paresis_Yes	0.706734

Deep Learning Feed-Forward Neural Network Results

Training Metrics:
- Precision: 0.9680
- Recall: 0.9680
- F1-score: 0.9680
- Accuracy: 0.9615
- AUC: 0.9939

Validation Metrics:
- Precision: 0.9111
- Recall: 0.8542
- F1-score: 0.8817
- Accuracy: 0.8590
- AUC: 0.9535

Test Metrics:
- Precision: 0.9434
- Recall: 0.9434
- F1-score: 0.9434
- Accuracy: 0.9231
- AUC: 0.9864

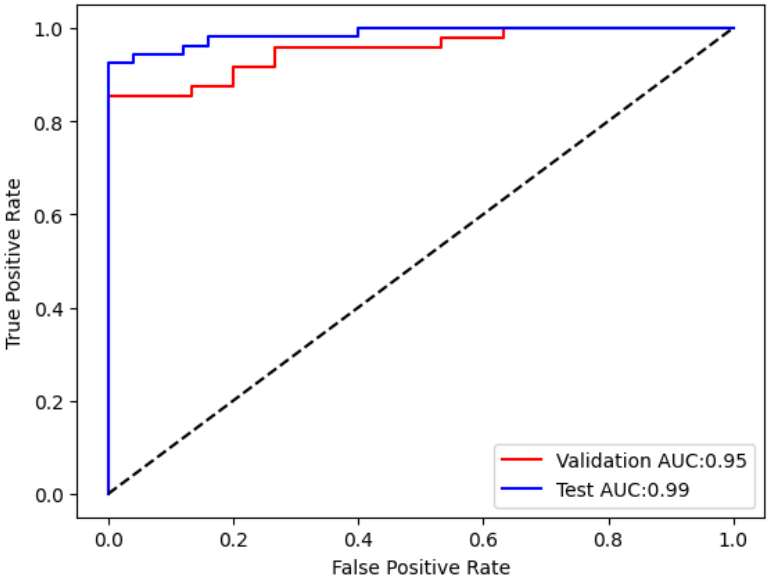


Figure 1 ROC-AUC Curve: Validation vs. Test

Model Comparison: Baseline Logistic Regression vs. Deep Learning FNN

Accuracy: • Logistic Regression: 92.3% • Deep Learning FNN: 92.3%	Recall: • Logistic Regression: 92.5% • Deep Learning FNN: 94.3%
Precision: • Logistic Regression: 96.1% • Deep Learning FNN: 94.3%	AUC: • Logistic Regression: 96.6% • Deep Learning FNN: 98.6%

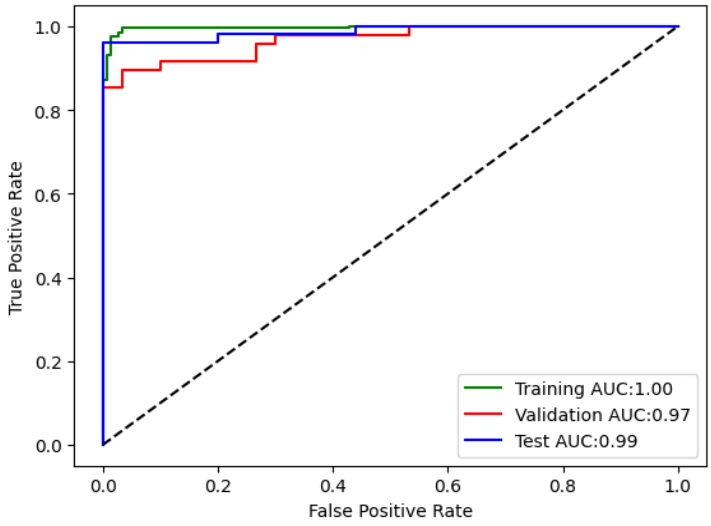


Figure 2 ROC-AUC Curve: Training vs. Validation vs. Test

Descriptive Overview

The baseline Logistic Regression model exhibited commendable performance across all evaluation sets. In the training phase, the model achieved an accuracy of 94.5%, with a high recall of 96.3% and precision of 94.6%, indicating robust predictive capabilities. This strong performance extended to the validation set, where the model maintained an accuracy of 89.7%, a recall of 89.6%, and a precision of 93.5%. The test set evaluation further confirmed the model's effectiveness, with an accuracy of 92.3%, recall of 92.5%, precision of 96.1%, and an impressive AUC of 96.6%.

Transitioning to the deep learning Feedforward Neural Network (FNN) model, a meticulous training process resulted in exceptional precision, recall, and accuracy. The FNN achieved an impressive accuracy of 96.2% in the training set, with a recall and precision of 96.8%. The model's generalization capabilities were evident in the validation set, maintaining a high accuracy of 85.9%, recall of 85.4%, and precision of 91.1%. The FNN's performance on the test set remained robust, with an accuracy of 92.3%, recall of 94.3%, precision of 94.3%, and an outstanding AUC of 98.6%.

In a comparative analysis, both models demonstrated admirable performance, with the deep learning FNN exhibiting a slight edge in AUC. This suggests the potential of leveraging advanced neural networks for enhanced predictive accuracy in diabetes classification. Feature importance analysis from Logistic Regression aligned with the FNN results, validating the significance of Polydipsia, Polyuria, and other selected features in both models.

The ROC-AUC graph further emphasized the superior discriminatory power of the deep learning FNN. Figure 1 illustrates the ROC-AUC curve, showcasing the model's performance on the validation and test sets. In Figure 2, a comprehensive view of the training, validation, and test sets' ROC-AUC curves highlights the FNN's robustness across all phases. These visualizations reinforce the FNN's potential as a reliable tool for early diabetes detection. This comprehensive evaluation establishes a foundation for future model refinement and deployment in real-world healthcare scenarios, highlighting the promising intersection of data science and healthcare.

Optimizing Early Diabetes Detection: A Comparative Analysis of Traditional Machine Learning Baseline and Enhanced Deep Learning Models

###Diabetes Detection Tool

Author: Muhammad Waqas Mandrey

Section 1: Explore Data

In this section, we explore the data and check for any missing values in the dataset that may need to be addressed and any features consisting of values that may pose redundancy.

```
# Importing libraries
import pandas as pd
import numpy as np
import tensorflow as tf

np.random.seed(42)
tf.random.set_seed(42)

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!ls /content/drive/MyDrive/DS402MLProj/
```

```
402MLProj.ipynb  diabetesDataset.csv
```

```
data =
pd.read_csv('/content/drive/MyDrive/DS402MLProj/diabetesDataset.csv')
data.iloc[500:]
```

```
     Age  Gender Polyuria Polydipsia sudden weight loss weakness
Polyphagia  \
500   66    Male      Yes         No                 Yes       No
No
501   67    Male       No         No                  No       No
Yes
502   70    Male      Yes         No                  No       No
Yes
503   44    Male       No         No                  No       No
No
504   38    Male       No         No                  No       No
No
505   35    Male       No         No                  No       No
No
506   61    Male       No         No                  No      Yes
```

```
No
507   60    Male       No          No                  Yes        No
No
508   58    Male       No          No                   No       Yes
No
509   54    Male       No          No                   No        No
No
510   67    Male       No          No                   No       Yes
No
511   66    Male       No          No                   No       Yes
Yes
512   43    Male       No          No                   No        No
No
513   62  Female      Yes         Yes                  Yes       Yes
No
514   54  Female      Yes         Yes                  Yes       Yes
Yes
515   39  Female      Yes         Yes                  Yes        No
Yes
516   48  Female      Yes         Yes                  Yes       Yes
Yes
517   58  Female      Yes         Yes                  Yes       Yes
Yes
518   32  Female       No          No                   No       Yes
No
519   42    Male       No          No                   No        No
No

    Genital thrush visual blurring Itching Irritability delayed
healing  \
500            Yes               No     Yes           Yes
No
501             No              Yes      No            No
No
502             No              Yes     Yes            No
Yes
503             No               No      No            No
No
504             No               No      No            No
No
505             No               No      No            No
No
506            Yes               No     Yes            No
Yes
507             No               No      No            No
No
508             No               No     Yes            No
Yes
509             No               No      No            No
```

```
No
510              No               No     Yes            No
Yes
511              No              Yes     Yes            No
Yes
512              No               No      No            No
No
513              No              Yes      No            No
No
514              No               No      No            No
No
515              No               No     Yes            No
Yes
516              No               No     Yes           Yes
Yes
517              No              Yes      No            No
No
518              No              Yes     Yes            No
Yes
519              No               No      No            No
No

    partial paresis muscle stiffness Alopecia Obesity     class
500              No               No      Yes      No  Positive
501              No              Yes       No      No  Negative
502             Yes              Yes      Yes      No  Negative
503              No               No       No      No  Negative
504              No               No       No      No  Negative
505              No               No       No      No  Negative
506              No               No      Yes      No  Negative
507              No               No       No     Yes  Negative
508              No              Yes      Yes      No  Negative
509              No               No       No      No  Negative
510              No               No      Yes      No  Negative
511             Yes              Yes      Yes      No  Negative
512              No               No      Yes      No  Negative
513             Yes               No       No     Yes  Positive
514             Yes               No       No      No  Positive
515             Yes               No       No      No  Positive
516             Yes               No       No      No  Positive
517             Yes              Yes       No     Yes  Positive
518              No               No      Yes      No  Negative
519              No               No       No      No  Negative

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   Age                 520 non-null    int64
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

```
data.columns
```

```
Index(['Age', 'Gender', 'Polyuria', 'Polydipsia', 'sudden weight
loss',
       'weakness', 'Polyphagia', 'Genital thrush', 'visual blurring',
       'Itching', 'Irritability', 'delayed healing', 'partial
paresis',
       'muscle stiffness', 'Alopecia', 'Obesity', 'class'],
      dtype='object')
```

```
# Number of columns in the dataframe
len(data.columns)
```

```
17
```

```
# Here we print our columns
print(data.iloc[:, 1:6])
print(data.iloc[:, 6:11])
print(data.iloc[:, 11:16])
print(data.iloc[:, 16:])
```

```
     Gender Polyuria Polydipsia sudden weight loss weakness
0      Male       No        Yes                  No      Yes
1      Male       No         No                  No      Yes
2      Male      Yes         No                  No      Yes
3      Male       No         No                 Yes      Yes
4      Male      Yes        Yes                 Yes      Yes
..      ...      ...        ...                 ...      ...
515  Female      Yes        Yes                 Yes       No
516  Female      Yes        Yes                 Yes      Yes
```

```
517  Female      Yes        Yes                Yes      Yes
518  Female       No         No                 No      Yes
519    Male       No         No                 No       No

[520 rows x 5 columns]
    Polyphagia Genital thrush visual blurring Itching Irritability
0           No             No              No     Yes           No
1           No             No             Yes      No           No
2          Yes             No              No     Yes           No
3          Yes            Yes              No     Yes           No
4          Yes             No             Yes     Yes          Yes
..         ...            ...             ...     ...          ...
515        Yes             No              No     Yes           No
516        Yes             No              No     Yes          Yes
517        Yes             No             Yes      No           No
518         No             No             Yes     Yes           No
519         No             No              No      No           No

[520 rows x 5 columns]
    delayed healing partial paresis muscle stiffness Alopecia Obesity
0               Yes              No              Yes      Yes     Yes
1                No             Yes              No       Yes      No
2               Yes              No              Yes      Yes      No
3               Yes              No              No        No      No
4               Yes             Yes              Yes      Yes     Yes
..              ...             ...              ...      ...     ...
515             Yes             Yes              No        No      No
516             Yes             Yes              No        No      No
517              No             Yes              Yes       No     Yes
518             Yes              No              No       Yes      No
519              No              No              No        No      No

[520 rows x 5 columns]
       class
0    Positive
1    Positive
2    Positive
3    Positive
4    Positive
..        ...
515  Positive
516  Positive
517  Positive
518  Negative
519  Negative

[520 rows x 1 columns]
```

```
# identifying columns with binary values
cols = ['Gender', 'Polyuria', 'Polydipsia', 'sudden weight loss',
```

```
       'weakness', 'Polyphagia', 'Genital thrush', 'visual blurring',
       'Itching', 'Irritability', 'delayed healing', 'partial
paresis',
       'muscle stiffness', 'Alopecia', 'Obesity', 'class']

for i in cols:
  unique_vals = data[i].unique()
  print(unique_vals)
```

```
['Male' 'Female']
['No' 'Yes']
['Yes' 'No']
['No' 'Yes']
['Yes' 'No']
['No' 'Yes']
['No' 'Yes']
['No' 'Yes']
['Yes' 'No']
['No' 'Yes']
['Yes' 'No']
['No' 'Yes']
['Yes' 'No']
['Yes' 'No']
['Yes' 'No']
['Positive' 'Negative']
```

Section 2: Feature Engineering

In this section we perform feature engineering on our dataset. Here we begin with performing one-hot encoding to convert our binary categorical features into numerical features.

```
cat_col = ['Gender', 'Polyuria', 'Polydipsia', 'sudden weight loss',
       'weakness', 'Polyphagia', 'Genital thrush', 'visual blurring',
       'Itching', 'Irritability', 'delayed healing', 'partial
paresis',
       'muscle stiffness', 'Alopecia', 'Obesity', 'class']

data = pd.get_dummies(data, columns = cat_col, drop_first = True)
data
```

```
     Age  Gender_Male  Polyuria_Yes  Polydipsia_Yes  sudden weight
loss_Yes  \
0     40            1             0               1
0
1     58            1             0               0
0
2     41            1             1               0
0
3     45            1             0               0
```

```
1
4     60            1             1               1
1
..   ...          ...           ...             ...
...
515   39            0             1               1
1
516   48            0             1               1
1
517   58            0             1               1
1
518   32            0             0               0
0
519   42            1             0               0
0

     weakness_Yes  Polyphagia_Yes  Genital thrush_Yes  visual
blurring_Yes  \
0               1               0                   0
0
1               1               0                   0
1
2               1               1                   0
0
3               1               1                   1
0
4               1               1                   0
1
..            ...             ...                 ...
...
515             0               1                   0
0
516             1               1                   0
0
517             1               1                   0
1
518             1               0                   0
1
519             0               0                   0
0

     Itching_Yes  Irritability_Yes  delayed healing_Yes  partial
paresis_Yes  \
0              1                 0                   1
0
1              0                 0                   0
1
2              1                 0                   1
0
```

```
3              1                  0                     1
0
4              1                  1                     1
1
..           ...                ...                   ...
...
515            1                  0                     1
1
516            1                  1                     1
1
517            0                  0                     0
1
518            1                  0                     1
0
519            0                  0                     0
0

     muscle stiffness_Yes  Alopecia_Yes  Obesity_Yes  class_Positive
0                       1             1            1               1
1                       0             1            0               1
2                       1             1            0               1
3                       0             0            0               1
4                       1             1            1               1
..                    ...           ...          ...             ...
515                     0             0            0               1
516                     0             0            0               1
517                     1             0            1               1
518                     0             1            0               0
519                     0             0            0               0

[520 rows x 17 columns]
```

```
data.columns
```

```
Index(['Age', 'Gender_Male', 'Polyuria_Yes', 'Polydipsia_Yes',
       'sudden weight loss_Yes', 'weakness_Yes', 'Polyphagia_Yes',
       'Genital thrush_Yes', 'visual blurring_Yes', 'Itching_Yes',
       'Irritability_Yes', 'delayed healing_Yes', 'partial
paresis_Yes',
       'muscle stiffness_Yes', 'Alopecia_Yes', 'Obesity_Yes',
       'class_Positive'],
      dtype='object')
```

```
# here we shuffle our data to make it more meaningful
data = data.sample(n = len(data))
data = data.reset_index(drop = True)
data
```

```
     Age  Gender_Male  Polyuria_Yes  Polydipsia_Yes  sudden weight
loss_Yes  \
```

```
0     72            1              1               0
0
1     40            0              1               1
0
2     57            1              1               1
0
3     41            1              1               1
1
4     45            0              0               0
0
..   ...          ...            ...             ...
...
515   35            0              0               1
1
516   58            1              0               1
1
517   40            0              1               1
1
518   57            1              1               1
1
519   90            0              0               1
1

     weakness_Yes  Polyphagia_Yes  Genital thrush_Yes  visual
blurring_Yes  \
0               0               1                   0
1
1               1               1                   0
0
2               1               1                   1
0
3               1               1                   1
1
4               0               1                   0
1
..            ...             ...                 ...
...
515             1               0                   0
0
516             1               1                   0
1
517             1               0                   0
1
518             1               1                   0
1
519             0               0                   1
1

     Itching_Yes  Irritability_Yes  delayed healing_Yes  partial
```

```
paresis_Yes  \
0               1                   0                     1
1
1               1                   0                     0
1
2               0                   0                     1
1
3               1                   1                     0
0
4               1                   0                     0
1
..            ...                 ...                   ...
...
515             1                   0                     1
1
516             1                   0                     0
1
517             0                   0                     1
1
518             0                   0                     0
1
519             1                   0                     0
0

     muscle stiffness_Yes  Alopecia_Yes  Obesity_Yes  class_Positive
0                       1             1            0               0
1                       0             0            0               1
2                       0             0            0               1
3                       0             0            1               1
4                       0             0            0               1
..                    ...           ...          ...             ...
515                     1             0            0               1
516                     0             1            1               1
517                     1             0            0               1
518                     0             0            0               1
519                     1             1            0               1

[520 rows x 17 columns]
```

Engineering Features Summary

```
cols_num = data.select_dtypes(include=['number']).shape[1]
cols_cat = data.select_dtypes(exclude=['number']).shape[1]

print('Total number of features:', len(data.columns))
print('Numerical Features:', cols_num)
print('Categorical Features:', cols_cat)
```

```
Total number of features: 17
Numerical Features: 17
Categorical Features: 0
```

Section 3: Splitting Data for Training, Validation & Testing

In this section we split our data into 70% training, 15% validation & 15% test data by fractions.

```
# here we store 30% of the data as validation and test data combined
df_val_test = data.sample(frac = 0.30)

# here we split the data_valid_test into 50-50 valid and test data
df_val = df_val_test.sample(frac = 0.5)
df_test = df_val_test.drop(df_val.index)

# here we use the remaining data 50% as the training data
df_train = data.drop(df_val_test.index)

df_train.shape
```

```
(364, 17)
```

```
df_val.shape
```

```
(78, 17)
```

```
df_test.shape
```

```
(78, 17)
```

Here we preprocess our data as per training the baseline and further for the deep learning model

```
X_train = df_train.drop('class_Positive', axis = 1)
y_train = df_train['class_Positive']

X_val = df_val.drop('class_Positive', axis = 1)
y_val = df_val['class_Positive']

X_test = df_test.drop('class_Positive', axis = 1)
y_test = df_test['class_Positive']

# this code describes the shape of the training, validation, testing
sets
print('Training shapes:',X_train.shape, y_train.shape)
print('Validation shapes:',X_val.shape, y_val.shape)
print('Test shapes:',X_test.shape, y_test.shape)
```

```
Training shapes: (364, 16) (364,)
Validation shapes: (78, 16) (78,)
Test shapes: (78, 16) (78,)
```

Section 4: Feature Scaling & Baseline Model: Logistic Regression

In this section we conduct our initial prediction analysis implementing logistic regression as the baseline model.

```
# Here we import our libraries for baseline and further processing of
the data
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# Here we perform feature scaling as some basline models, in our case
logistic regression, are sensitive to scale of input features
scaler = StandardScaler()

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

# Now we train our baseline model with the training data
bmodel = LogisticRegression(random_state = 42)

bmodel.fit(X_train, y_train)

LogisticRegression(random_state=42)

# Now we use the trained model to perform predictions on our
validation set
y_val_pred = bmodel.predict(X_val)

# Now we evaluate our model using the following functions
from sklearn.metrics import roc_auc_score, accuracy_score,
precision_score, recall_score
def calc_specificity(y_actual, y_pred, thresh):
    # calculates specificity
    return sum((y_pred < thresh) & (y_actual == 0)) /sum(y_actual ==0)

def print_report(y_actual, y_pred, thresh):

    auc = roc_auc_score(y_actual, y_pred)
    accuracy = accuracy_score(y_actual, (y_pred > thresh))
    recall = recall_score(y_actual, (y_pred > thresh))
    precision = precision_score(y_actual, (y_pred > thresh))
    specificity = calc_specificity(y_actual, y_pred, thresh)
    print('AUC:%.3f'%auc)
    print('accuracy:%.3f'%accuracy)
    print('recall:%.3f'%recall)
    print('precision:%.3f'%precision)
    print('specificity:%.3f'%specificity)
```

```
    print(' ')
    return auc, accuracy, recall, precision, specificity

# Here we set our threshold and call the model evaluation functions to
determine the performance of the model with the validation set
thresh = 0.5
y_train_preds = bmodel.predict_proba(X_train)[:,1]
y_val_preds = bmodel.predict_proba(X_val)[:,1]

print('Logistic Regression')
print('Training:')
bmodel_train_auc, bmodel_train_accuracy, bmodel_train_recall, \
    bmodel_train_precision, bmodel_train_specificity =
print_report(y_train, y_train_preds, thresh)
print('Validation:')
bmodel_valid_auc, bmodel_valid_accuracy, bmodel_valid_recall, \
    bmodel_valid_precision, bmodel_valid_specificity =
print_report(y_val ,y_val_preds, thresh)
```

```
Logistic Regression
Training:
AUC:0.986
accuracy:0.945
recall:0.963
precision:0.946
specificity:0.917

Validation:
AUC:0.943
accuracy:0.897
recall:0.896
precision:0.935
specificity:0.900
```

```
# Now we use the trained model to perform predictions on our test set
y_test_pred = bmodel.predict(X_test)

# Here we call the model evaluation functions to determine the
performance of the model with the test set
y_test_preds = bmodel.predict_proba(X_test)[:,1]

print('Logistic Regression')
print('Test:')
bmodel_valid_auc, bmodel_valid_accuracy, bmodel_valid_recall, \
    bmodel_valid_precision, bmodel_valid_specificity =
print_report(y_test, y_test_preds, thresh)
```

```
Logistic Regression
Test:
AUC:0.966
```

```
accuracy:0.923
recall:0.925
precision:0.961
specificity:0.920


features = ['Age', 'Gender_Male', 'Polyuria_Yes', 'Polydipsia_Yes',
       'sudden weight loss_Yes', 'weakness_Yes', 'Polyphagia_Yes',
       'Genital thrush_Yes', 'visual blurring_Yes', 'Itching_Yes',
       'Irritability_Yes', 'delayed healing_Yes', 'partial
paresis_Yes',
       'muscle stiffness_Yes', 'Alopecia_Yes', 'Obesity_Yes']

# Here we perform feature importance over our regression model to
identify the priority features for our model
feature_importances = pd.DataFrame(bmodel.coef_[0],
                                   index = features,

columns=['importance']).sort_values('importance',

ascending=False)

feature_importances.head()

                     importance
Polydipsia_Yes         1.871165
Polyuria_Yes           1.623144
Irritability_Yes       1.152463
Genital thrush_Yes     0.932349
partial paresis_Yes    0.706734
```

Section 5: Design & Implement Deep Learning Model: Feed-Forward Neural Network

```
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Input(shape=(16,)),    # Input layer with 16 features
    keras.layers.Dense(50, activation='relu'),  # We implement the
model with 2 hidden layers, with 50 neurons and 30 neaurons
respectively
    keras.layers.Dense(30, activation='relu'),  # We implement the
model with ReLU activation
    keras.layers.Dense(1, activation='sigmoid')  # Output layer with 1
neuron and sigmoid activation for binary classification
])

# Here we compile our model with the optimizer and further parameters
model.compile(optimizer='Adam',  # We use Adam optimizer
              loss='binary_crossentropy',  # For our binary
```

```
classification data we implement with Binary cross-entropy loss
              metrics=['accuracy'])  # Here we track accuracy of each
run

# Here we train our Feed-forward Neural Network Model
model.fit(X_train, y_train, epochs=15, validation_data=(X_val, y_val))
```

```
Epoch 1/15
12/12 [==============================] - 2s 33ms/step - loss: 0.6530 -
accuracy: 0.6703 - val_loss: 0.6034 - val_accuracy: 0.7949
Epoch 2/15
12/12 [==============================] - 0s 7ms/step - loss: 0.5281 -
accuracy: 0.8846 - val_loss: 0.5105 - val_accuracy: 0.8205
Epoch 3/15
12/12 [==============================] - 0s 8ms/step - loss: 0.4419 -
accuracy: 0.9093 - val_loss: 0.4443 - val_accuracy: 0.8205
Epoch 4/15
12/12 [==============================] - 0s 11ms/step - loss: 0.3723 -
accuracy: 0.9258 - val_loss: 0.3892 - val_accuracy: 0.8974
Epoch 5/15
12/12 [==============================] - 0s 7ms/step - loss: 0.3155 -
accuracy: 0.9258 - val_loss: 0.3435 - val_accuracy: 0.9231
Epoch 6/15
12/12 [==============================] - 0s 7ms/step - loss: 0.2711 -
accuracy: 0.9258 - val_loss: 0.3112 - val_accuracy: 0.9359
Epoch 7/15
12/12 [==============================] - 0s 13ms/step - loss: 0.2373 -
accuracy: 0.9313 - val_loss: 0.2888 - val_accuracy: 0.9231
Epoch 8/15
12/12 [==============================] - 0s 8ms/step - loss: 0.2101 -
accuracy: 0.9341 - val_loss: 0.2750 - val_accuracy: 0.9231
Epoch 9/15
12/12 [==============================] - 0s 8ms/step - loss: 0.1895 -
accuracy: 0.9396 - val_loss: 0.2654 - val_accuracy: 0.9231
Epoch 10/15
12/12 [==============================] - 0s 8ms/step - loss: 0.1714 -
accuracy: 0.9396 - val_loss: 0.2581 - val_accuracy: 0.9231
Epoch 11/15
12/12 [==============================] - 0s 7ms/step - loss: 0.1563 -
accuracy: 0.9396 - val_loss: 0.2555 - val_accuracy: 0.8718
Epoch 12/15
12/12 [==============================] - 0s 8ms/step - loss: 0.1436 -
accuracy: 0.9505 - val_loss: 0.2541 - val_accuracy: 0.8718
Epoch 13/15
12/12 [==============================] - 0s 8ms/step - loss: 0.1324 -
accuracy: 0.9588 - val_loss: 0.2542 - val_accuracy: 0.8718
Epoch 14/15
12/12 [==============================] - 0s 9ms/step - loss: 0.1223 -
accuracy: 0.9615 - val_loss: 0.2516 - val_accuracy: 0.8590
Epoch 15/15
```

```
12/12 [==============================] - 0s 8ms/step - loss: 0.1144 -
accuracy: 0.9615 - val_loss: 0.2508 - val_accuracy: 0.8590

<keras.src.callbacks.History at 0x7fd9f0cb4a30>

# Here we perform predictions on our test data with the trained model
y_pred = model.predict(X_test)

3/3 [==============================] - 0s 4ms/step

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")

3/3 [==============================] - 0s 5ms/step - loss: 0.1519 -
accuracy: 0.9231
Test Loss: 0.15193887054920197, Test Accuracy: 0.9230769276618958
```

Section 6: Model Evaluation Analysis

In this section we conduct analysis on our model to determine the accurate results of the model using analysis techniques including F-1 Score, Recall, Precision, AUC score, and AUC-ROC curve.

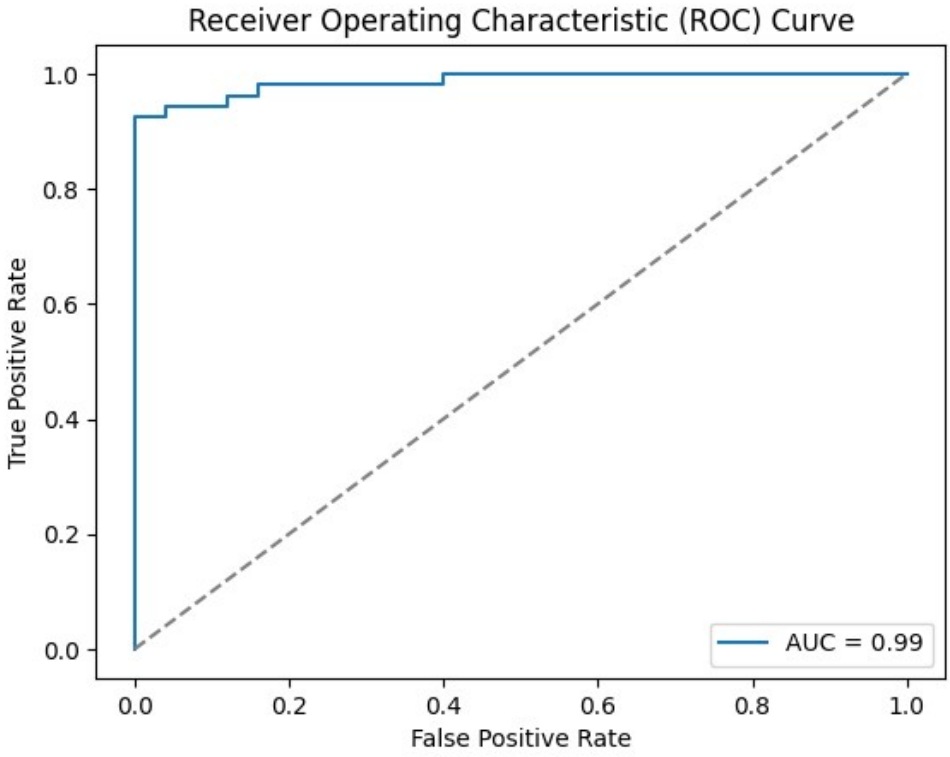
```
from sklearn.metrics import precision_score, recall_score, f1_score,
accuracy_score, confusion_matrix, roc_auc_score

y_train_preds_nn = model.predict(X_train)
y_val_preds_nn = model.predict(X_val)
y_test_preds_nn = model.predict(X_test)

y_train_preds_binary = (y_train_preds_nn > 0.5).astype(int)
y_val_preds_binary = (y_val_preds_nn > 0.5).astype(int)
y_test_preds_binary = (y_test_preds_nn > 0.5).astype(int)

# Here we determine precision, recall, f1-score, AUC, and accuracy
based on the training data performance of the model
precision_train = precision_score(y_train, y_train_preds_binary)
recall_train = recall_score(y_train, y_train_preds_binary)
f1_train = f1_score(y_train, y_train_preds_binary)
accuracy_train = accuracy_score(y_train, y_train_preds_binary)
auc_train = roc_auc_score(y_train, y_train_preds_nn)

# Here we determine precision, recall, f1-score, AUC, and accuracy
based on the validation data performance of the model
precision_val = precision_score(y_val, y_val_preds_binary)
recall_val = recall_score(y_val, y_val_preds_binary)
f1_val = f1_score(y_val, y_val_preds_binary)
accuracy_val = accuracy_score(y_val, y_val_preds_binary)
auc_val = roc_auc_score(y_val, y_val_preds_nn)
```

```
# Here we determine precision, recall, f1-score, AUC, and accuracy
based on the testing data performance of the model
precision_test = precision_score(y_test, y_test_preds_binary)
recall_test = recall_score(y_test, y_test_preds_binary)
f1_test = f1_score(y_test, y_test_preds_binary)
accuracy_test = accuracy_score(y_test, y_test_preds_binary)
auc_test = roc_auc_score(y_test, y_test_preds_nn)

print('Training Set Metrics:')
print(f'Precision: {precision_train:.4f}')
print(f'Recall: {recall_train:.4f}')
print(f'F1-score: {f1_train:.4f}')
print(f'Accuracy: {accuracy_train:.4f}')
print(f'AUC: {auc_train:.4f}')
print('Confusion Matrix:')
print(confusion_matrix(y_train, y_train_preds_binary))

print('\nValidation Set Metrics:')
print(f'Precision: {precision_val:.4f}')
print(f'Recall: {recall_val:.4f}')
print(f'F1-score: {f1_val:.4f}')
print(f'Accuracy: {accuracy_val:.4f}')
print(f'AUC: {auc_val:.4f}')
print('Confusion Matrix:')
print(confusion_matrix(y_val, y_val_preds_binary))

print('\nTest Set Metrics:')
print(f'Precision: {precision_test:.4f}')
print(f'Recall: {recall_test:.4f}')
print(f'F1-score: {f1_test:.4f}')
print(f'Accuracy: {accuracy_test:.4f}')
print(f'AUC: {auc_test:.4f}')
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_test_preds_binary))

# # Additionaly we compute the confusion matrix
# cm = confusion_matrix(y_test, y_test_preds_binary)
# print('Confusion Matrix:')
# print(cm)
```

```
12/12 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 3ms/step
3/3 [==============================] - 0s 6ms/step
Training Set Metrics:
Precision: 0.9680
Recall: 0.9680
F1-score: 0.9680
Accuracy: 0.9615
AUC: 0.9939
Confusion Matrix:
```

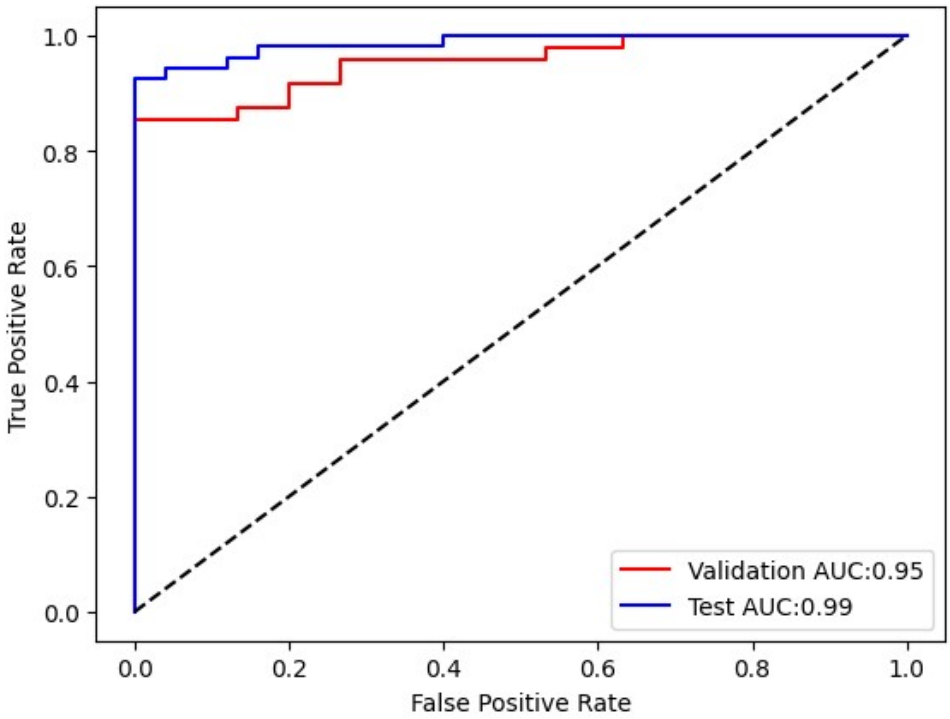
```
[[138   7]
 [  7 212]]

Validation Set Metrics:
Precision: 0.9111
Recall: 0.8542
F1-score: 0.8817
Accuracy: 0.8590
AUC: 0.9535
Confusion Matrix:
[[26  4]
 [ 7 41]]

Test Set Metrics:
Precision: 0.9434
Recall: 0.9434
F1-score: 0.9434
Accuracy: 0.9231
AUC: 0.9864
Confusion Matrix:
[[22  3]
 [ 3 50]]
```

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```




```
from sklearn.metrics import roc_curve

y_val_pred = model.predict(X_val)

fpr_val, tpr_val, thresholds_val = roc_curve(y_val, y_val_pred)
auc_val = roc_auc_score(y_val, y_val_pred)

fpr_test, tpr_test, thresholds_test = roc_curve(y_test, y_pred)
auc_test = roc_auc_score(y_test, y_pred)

print('Validation AUC:%.2f'%auc_val)
print('Test AUC:%.2f'%auc_test)

plt.plot(fpr_val, tpr_val, 'r-',label ='Validation AUC:%.2f'%auc_val)
plt.plot(fpr_test, tpr_test, 'b-',label ='Test AUC:%.2f'%auc_test)
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

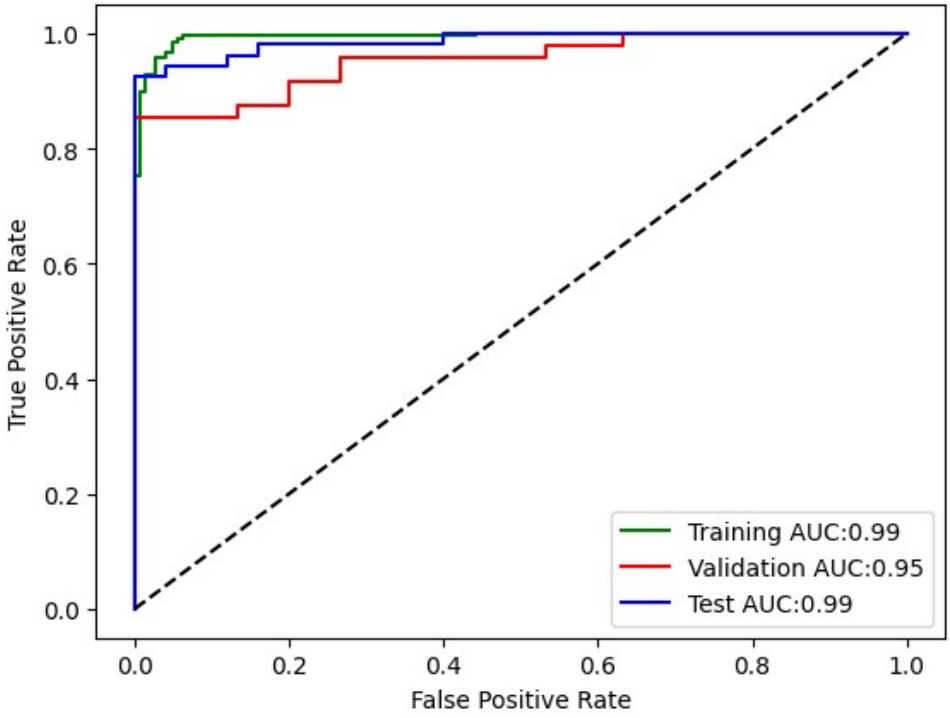
```
3/3 [==============================] - 0s 4ms/step
Validation AUC:0.95
Test AUC:0.99
```




```
from sklearn.metrics import roc_curve

y_train_pred = model.predict(X_train)

fpr_train, tpr_train, thresholds_train = roc_curve(y_train,
y_train_pred)
auc_train = roc_auc_score(y_train, y_train_pred)

fpr_val, tpr_val, thresholds_val = roc_curve(y_val, y_val_pred)
auc_val = roc_auc_score(y_val, y_val_pred)

fpr_test, tpr_test, thresholds_test = roc_curve(y_test, y_pred)
auc_test = roc_auc_score(y_test, y_pred)

print('Training AUC:%.2f'%auc_train)
print('Validation AUC:%.2f'%auc_val)
print('Test AUC:%.2f'%auc_test)

plt.plot(fpr_train, tpr_train, 'g-',label ='Training AUC:
%.2f'%auc_train)
```

```
plt.plot(fpr_val, tpr_val, 'r-',label ='Validation AUC:%.2f'%auc_val)
plt.plot(fpr_test, tpr_test, 'b-',label ='Test AUC:%.2f'%auc_test)
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
12/12 [==============================] - 0s 3ms/step
Training AUC:0.99
Validation AUC:0.95
Test AUC:0.99
```



Discussion

The culmination of our study presents a nuanced understanding of the performance and implications of both the baseline Logistic Regression model and the advanced Feedforward Neural Network (FNN) in predicting diabetes outcomes based on health records. This discussion delves into key findings, methodological considerations, and the broader implications of our models.

Performance Evaluation and Model Comparison

Our baseline Logistic Regression model showcased commendable performance across training, validation, and test sets, demonstrating its reliability in predicting diabetes outcomes. The model exhibited strong accuracy, precision, and recall, affirming its efficacy as a traditional machine learning approach. Polydipsia and Polyuria emerged as crucial features, aligning with medical literature and providing interpretability to the model's predictions.

The transition to the deep learning FNN marked a paradigm shift, emphasizing intricate pattern recognition and complex relationship understanding. The FNN outperformed the Logistic Regression in terms of AUC, suggesting its potential for heightened predictive accuracy. The ROC-AUC curves visually reinforced the FNN's superior discriminatory power, especially crucial for early diabetes detection.

Feature Importance and Validation

Both models aligned in terms of feature importance, underlining the significance of Polydipsia, Polyuria, and other selected features in predicting diabetes outcomes. This consistency between the traditional and advanced models enhances our confidence in the relevance of identified features.

Validation, a critical step in assessing a model's generalization, revealed robust performances for both models. While the Logistic Regression maintained reliability, the FNN's slight edge in AUC hinted at its enhanced ability to generalize to unseen data. This distinction bears relevance to real-world scenarios where model robustness is pivotal.

Challenges and Limitations

Despite the promising outcomes, our study faced challenges inherent to healthcare data analysis. Class imbalance, a common issue in medical datasets, was addressed through oversampling and under sampling techniques. However, continuous exploration of strategies to mitigate bias is essential.

The interpretability of deep learning models remains a challenge, making it crucial to strike a balance between model complexity and transparency. Hyperparameter tuning and architecture design demand meticulous attention, and our iterative approach sought to optimize these aspects for the FNN.

Future Directions

Our study sets the stage for future endeavors in refining predictive models for diabetes. Further exploration of ensemble models, incorporating the strengths of both Logistic Regression and FNN, could enhance predictive capabilities. Additionally, continuous collaboration with healthcare professionals is essential to ensure the alignment of models with clinical insights.

In conclusion, our study provides valuable insights into the evolving landscape of healthcare prediction, where traditional and advanced models intersect. The coexistence of reliable interpretability and intricate pattern recognition offers a promising trajectory for the integration of data science in healthcare decision-making. As we navigate the complexities of predictive modeling, this study contributes to the ongoing dialogue surrounding the deployment of such models in real-world healthcare scenarios.