AI Based Diabetes Prediction System

Introduction

In phase 4 we present a comprehensive analysis of the process of selecting a machine learning algorithm, building and training a Convolutional Neural Network (CNN) model, and evaluating its performance.

Selected Algorithm:

* 2DCNN:

2DCNN, or two-dimensional convolutional neural network, is a type of deep learning model that is specifically designed for processing images. It is composed of a series of convolutional layers, which can extract spatial features from the input image. These features can then be used for a variety of tasks, such as image classification, object detection, and image segmentation.

2DCNNs are typically trained using a supervised learning approach. This means that the model is given a set of training images, each of which is labelled with its corresponding class or segmentation. The model then learns to extract the features from the images that are most relevant to the task at hand.

* TabPFN:

TabPFN, or Tabular Prior-Data Fitted Network, is a type of neural network that is specifically designed for tabular data. It is a modified version of the original PFN architecture, which has been shown to be very effective for tabular classification tasks.

TabPFN has two key modifications that make it well-suited for tabular data:

* Attention masks: TabPFN uses slight adjustments to the attention masks in the PFN architecture, which results in shorter inference times.
* Zero-padding: TabPFN can handle datasets with varying numbers of features by zero-padding the shorter features.

TabPFN is also trained on synthetic datasets that are generated using principles from causal reasoning and simplicity. This gives TabPFN a prior knowledge of tabular data, which allows it to learn more quickly and efficiently.

TabPFN has been shown to achieve state-of-the-art results on a variety of tabular classification benchmarks. It is also very fast and efficient, making it ideal for real-time applications.

Data Pre-processing

Before constructing the CNN model, data pre-processing is performed. This step ensures that the dataset is compatible with the CNN architecture. We reshape the training and testing data for this purpose.

Python:



CNN Model Architecture

The Convolutional Neural Network (CNN) model is designed with several layers for feature extraction and classification. The architecture is defined as follows:

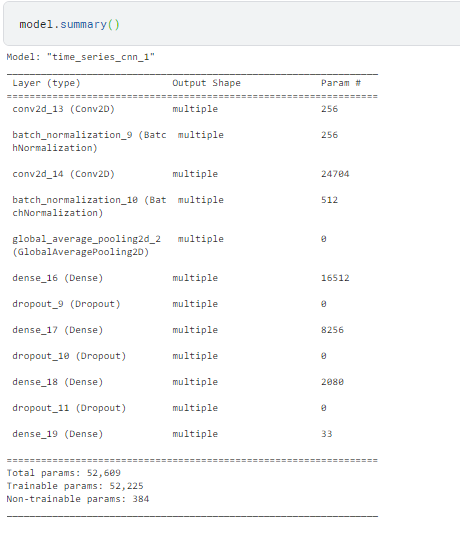
Python:



Model Summary

The summary of the CNN model, an overview of the network's architecture, including layer types, output shapes, and the number of trainable parameters.

Python:



Model Training

The model is compiled using the Adam optimizer, binary cross-entropy loss, and accuracy as the evaluation metric. It is then trained on the preprocessed training data for 300 epochs with a batch size of 128.

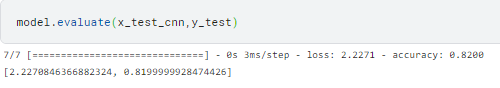
Python:



Model Evaluation

We evaluate the model's performance on the training data. The output includes the loss and accuracy.

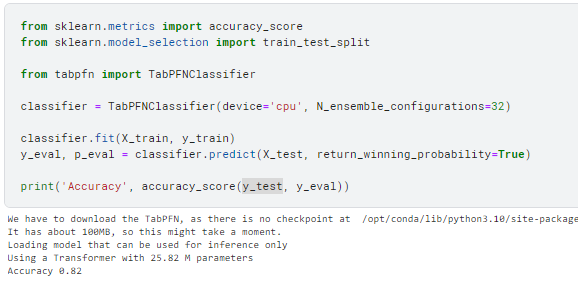
Python:

 Utilizing the

Model Training and Evaluation with TabPFN

A model is trained and evaluated using the TabPFN library. This model is configured with 32 ensemble configurations, and its accuracy is assessed.

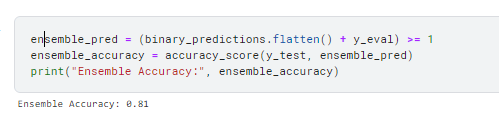
Python:



Ensemble of Models

To improve classification accuracy, we consider an ensemble of models. We combine the predictions from the CNN model and the TabPFN classifier by taking a majority vote.

Python:



Conclusion

This document summarizes the model selection, training, and evaluation processes for the CNN and TabPFN models. We also explored ensemble methods to combine the predictions of both models, aiming to improve classification accuracy.

Ensemble methods often provide better results by combining diverse models, and this approach could be further explored and fine-tuned to optimize the classification performance.