DIABETES HEALTH 3-CLASSIFICATOR (3-DHC)
A NEURAL NETWORK FOR DIABETES RISK PREDICTION

AGENDA

Introduction

Architecture and Components of 3-DHC Neural Network

Input data

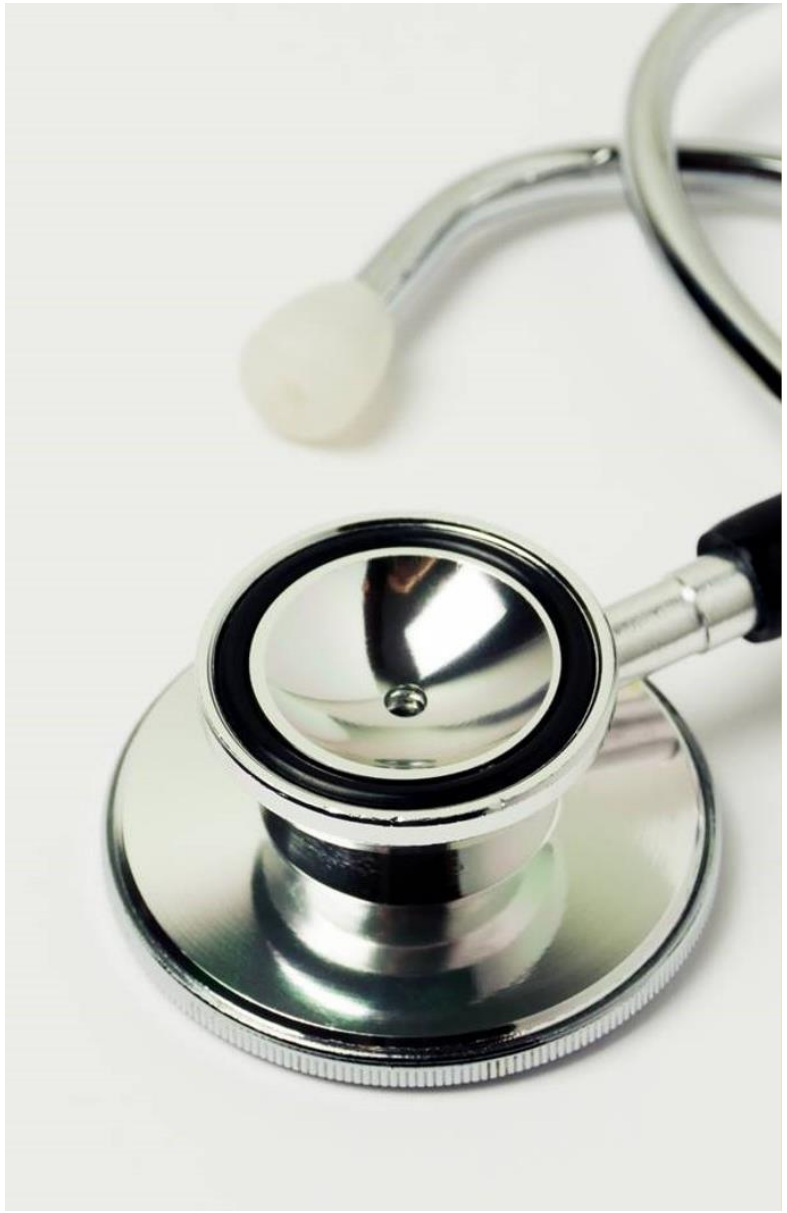
Training Process

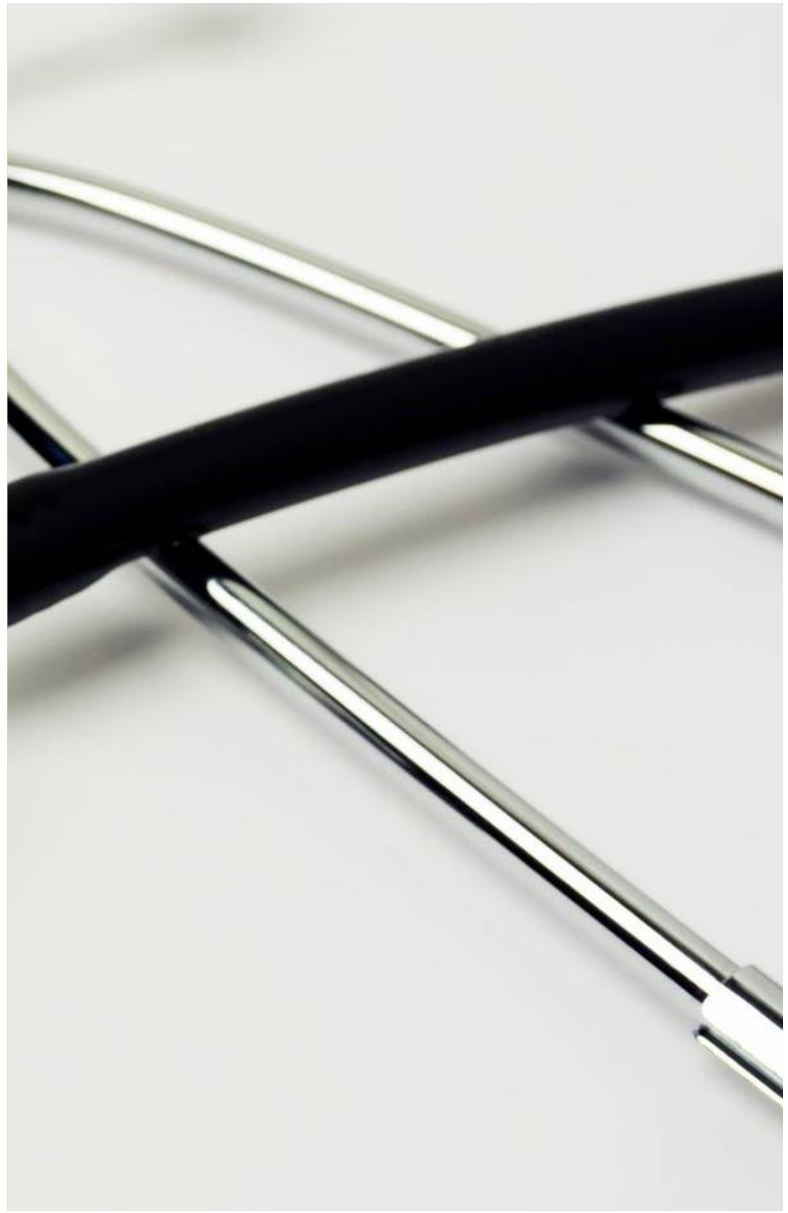
Results and their analysis

Testing phase

Prediction on new patients

Conclusions





INTRODUCTION

1. Objective
    a. Develop a neural network to classify individuals as **diabetic**, **prediabetic** or **non-diabetic** based on health indicators.
2. Dataset
    a. Source: Nhanes 2015-2016
    b. Size: 6328 records
    c. Features: 14 features. Some of these features:
        - HbA1c (Glycated Hemoglobin), Fasting Glucose (mg/dL), Fasting Glucose (mmol/L), Body Mass Index (BMI), Waist Circumference, Age, Lipid Profile (Total Cholesterol, HDL Cholesterol, LDL Cholesterol, Triglycerides)
3. Model Goals
    a. Build an accurate and generalizable predictive model
    b. Analyze feature importance
    c. Visualize training behavior and performance metrics
4. Implementation
    a. Language: Python
    b. Framework: JupiterLab, Voilà
    c. Libraries: NumPy, Pandas, Machine Learning tools
5. Task Type:
    a) 3- Classification: Diabetic, Prediabetic and Non-Diabetic
6. Relevance
    a. Early diabetes detection can significantly improve clinical outcomes.

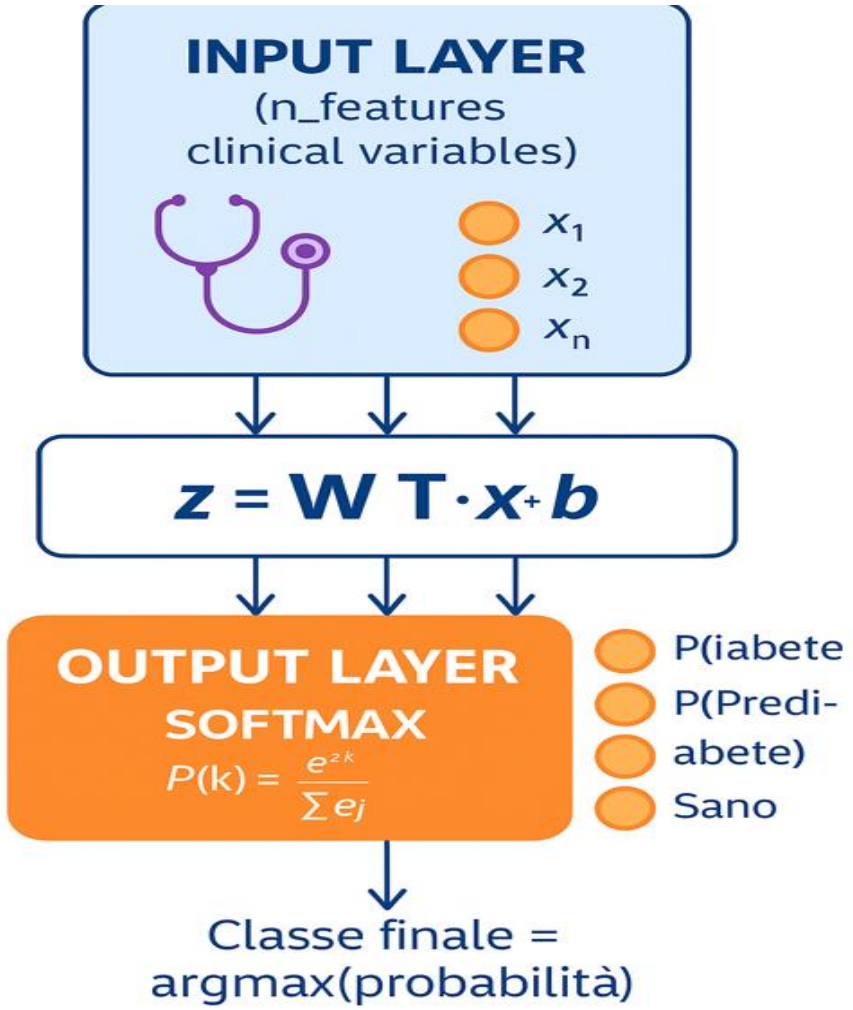
ARCHITECTURE AND COMPONENTS OF 3-DHC NEURAL NETWORK

ARCHITECTURE AND COMPONENTS OF 3-DHC NEURAL NETWORK

- The neural network designed for the 3-Class Diabetes Health Classifier (3-DHC) project is a simple but effective feedforward network for a 3-Class classification task (diabetic, prediabetic, no diabetic).
- It is a single-layer linear neural network, with fully connected input to 3 output neurons (one per class), Softmax activation, and training via cross-entropy minimization with LBFGS optimizer.

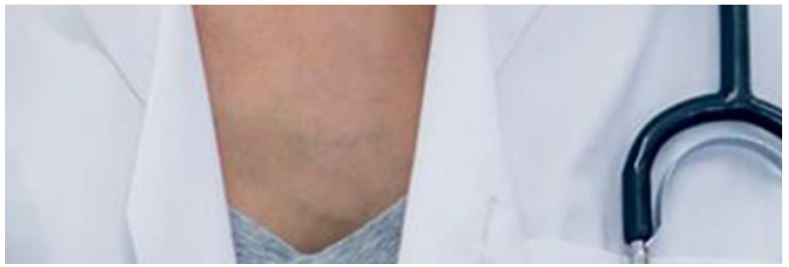
Layer	Number of Neurones	Activaction function
Input layer	14	None
Output layer	3	Softmax

ARCHITECTURE AND COMPONENTS OF 3-DHC NEURAL NETWORK



1. Input layer
    a. Number of neurons = number of selected features (
    b. Each neuron receives a scaled input value
2. Output Layer
    a. Number of neurons = 3 (Healthy, Prediabetes, Diabetes)
    b. Each neuron calculates:
        1. $z_k = w_k^T x + b_k$
    c. The outputs are transformed by Softmax into probabilities:
        1. $P(y = k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$
3. Final decision
    a. The model selects the class with maximum probability:
        1. $\hat{y} = \arg\max_k P(y = k)$
4. Loss function (Cross-Entropy)
    a. Formula
        1. Loss $= -\sum_i y_i \log(p_i)$
        2. $y_i = 1$for the correct class, 0 for the others (one-hot encoding)
        3. p_i= probabilities predicted by the softmax

INPUT DATA



INPUT DATA

- The dataset used to train and validate the multi-classes Diabetes Health Classifier (3-DHC) neural network is publicly available on the National Center for Health Statistics, NHANES 2015-2016
- The dataset contains:
  - Over 6328 samples (instances)
  - 14 features for each individual
  - 3-label as target
- Features include information on medical indicators: HbA1c (Glycated Hemoglobin), Fasting Glucose (mg/dL), Fasting Glucose (mmol/L), Body Mass Index (BMI), Waist Circumference, Age, Lipid Profile (Total Cholesterol, HDL Cholesterol, LDL Cholesterol, Triglycerides)

	SEQN	RIDAGEYR	RIAGENDR	BMXWT	BMXHT	BMXBMI	BMXWAIST	LBXGH	LBXGLU	LBDGLUSI	LBXTC	LBDHDD	LBXTR	LBDLDL	PHAFSTHR	Classe
0	83732	62	1	94,8	184.0	27,8	101.0	7	NaN	NaN	173.0	46.0	NaN	NaN	3	Diabete
1	83733	53	1	90,4	171.0	30,8	108.0	5,5	101.0	5,6	265.0	63.0	147.0	173.0	12	Sano
2	83734	78	1	83,4	170.0	28,8	116.0	5,8	84.0	4,7	229.0	30.0	269.0	145.0	10	Prediabete
3	83735	56	2	109,8	161.0	42,4	110.0	5,6	NaN	NaN	174.0	61.0	NaN	NaN	2	Sano
4	83736	42	2	55,2	165.0	20,3	80.0	5,6	84.0	4,7	204.0	53.0	47.0	142.0	10	Sano

INPUT DATA

- Preprocessing. To prepare the data for neural network training, some main preprocessing operations are performed:
  - Converting a column from string to float. This is typical for Italian datasets where the decimal separator is a comma.
  - For each feature, it calculates the column median and replaces any NaNs. The median is robust to odds (more so than the mean). This choice is deliberate and makes a lot of sense, especially in clinical datasets. Logistic Regression is sensitive to scale and distribution; using the median avoids excessive data skewing.
  - Label mapping (text class -> numeric code): Diabetes -> 0, Prediabetes -> 1, Healthy -> 2
  - Normalization
    - When training a neural network (or a machine learning algorithm in general), the input data must be scaled and normalized. Main reason: different feature scales can slow down, prevent or damage learning. *Min-Max* normalization scales each feature in the range [0, 1]:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

    - Summarizing: stable learning, balanced gradients, better generalization, all features treated equally

INPUT DATA

- Split of dataset:
  - Splits the dataset into a training set (70%) and a validation set (30%), based on a specified percentage. Training set is used for training phase, validation set is used for validation phase. Validation is used to verify that the training carried out with the training set does not generate overfitting, that is, the network is too attached to the data in the training set, or that it only knows how to classify the data in the training set. random_state=RANDOM_STATE (fixed value 42) makes the split playable.

TRAINING PROCESS

TRAINING PROCESS

- The training of the Diabetes Health Classifier (3-DHC) neural network occurs through two main phases:
  - Forward Propagation. The model makes a prediction based on the current parameters.
  - Backward Propagation. The model compares the prediction with the actual value, calculates the error, and updates the weights to improve the prediction.
- Training cycle according following parameters:
  - multi_class = multinomial' : with multi_class='multinomial' it uses a single model that simultaneously estimates the probabilities of all classes via *softmax*. With 'multinomial', the model "sees" all classes at once and learns decision boundaries with a unique logic. It is more consistent in problems such as 3-classes (e.g., Healthy / Prediabetes / Diabetes), because the sum of the probabilities is always 1 and the classes are mutually exclusive.
  - solver = 'lbfgs' : lbfgs is the optimization engine that performs forward and backward propagation, updating the weights to reduce the loss. It is a quasi-Newton method:
    - it uses the gradient of the cost function (derivatives),
    - it also approximates curvature information (Hessian),
    - it converges in few iterations on moderate-sized problems
  - max_iter = 1000 : "You have a thousand attempts to learn the weights." This is a prudent choice and almost protects you from non-convergence warnings.
  - random_state = 42 : every time you rerun the script, with the same data and parameters, you get exactly the same results (same metrics, same coefficients). It's useful for:
    - comparing different experiments in a meaningful way
    - being able to redo the same training and get the same numbers
    - solidifying your story to colleagues: "If you run it again, you'll see the same metrics."

RESULTS AND THEIR ANALYSIS

RESULTS AND THEIR ANALYSIS

- Accuracy Train: On the training set, the model correctly assigns the class (Healthy / Prediabetes / Diabetes) to 85.78% of patients. Of 100 patients in the training set, approximately 86 are correctly classified, 14 incorrectly.
- Accuracy Validation: On the validation set (data never seen during the training phase, randomly selected with a 70/30 split), the model is correct 85.99% of the time. Of 100 patients in the validation set, approximately 86 are correctly classified, 14 incorrectly.
- The key point is the ratio between the two values: very close (difference ≈ 0.0021 → 0.21 percentage points) and this implies there is no evident overfitting
- The multiclass logistic model you trained is stable, shows no signs of overfitting, and captures the pattern present in the dataset well. The quality is good for a prototype, but accuracy alone isn't enough to judge its clinical suitability: you need to analyze it by class (**classification report + confusion matrix**), paying particular attention to its ability to identify prediabetes and diabetes.

Metric	Value
Accuracy train	85,78%
Accuracy validation	85,99%

RESULTS AND THEIR ANALYSIS

- For each class
  - Precision = Of all the patients the model predicts in that class, how many actually belong to that class?
    - *When I say Diabetes, how often am I right?*
  - Recall = Of all the patients who actually belong to that class, how many do I intercept?
    - *Of 100 actual diabetics, how many do I identify as such?*
  - F1-score = Harmonic mean of precision and recall, summarizing the trade-off between the two.
  - Support = How many patients of that class are in the validation set.

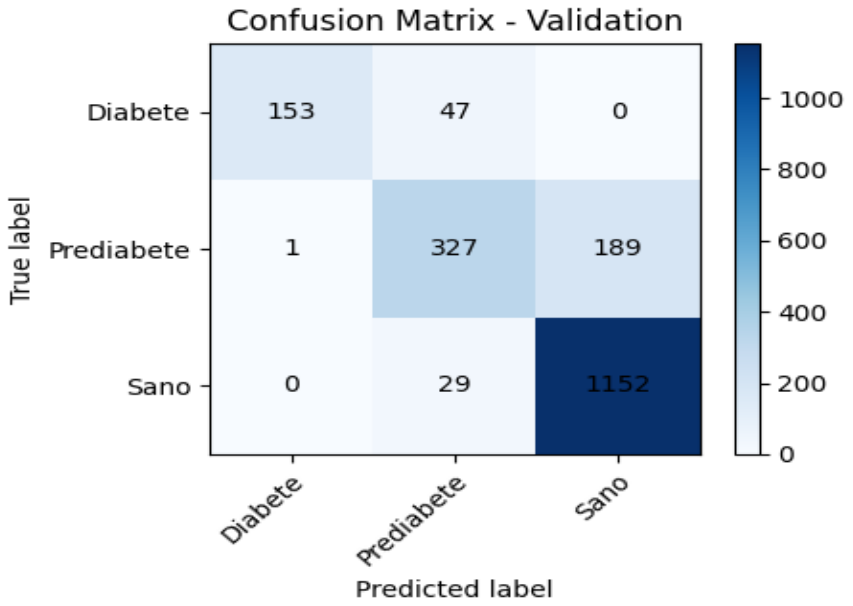
Classe	Precision	Recall	F1-score	Support
Diabete	0.99	0.77	0.86	200
Prediabete	0.81	0.63	0.71	517
Sano	0.86	0.98	0.91	1181

RESULTS AND THEIR ANALYSIS

- "Diabetes" row (200 actual diabetic patients)
  - 153 correctly predicted as having diabetes
  - 47 correctly predicted as prediabetes
  - 0 correctly predicted as healthy
- Percentages:
  - Diabetes → Diabetes: 153 / 200 ≈ 76.5%
  - Diabetes → Prediabetes: 47 / 200 ≈ 23.5%
  - Diabetes → Healthy: 0%

- "Prediabetes" row (517 actual prediabetic patients)
  - 1 predicted as having diabetes
  - 327 correctly predicted as having prediabetes
  - 189 predicted as having healthy
- Percentages:
  - Prediabetes → Diabetes: 1 / 517 ≈ 0.2%
  - Prediabetes → Prediabetes: 327 / 517 ≈ 63.2%
  - Prediabetes → Healthy: 189 / 517 ≈ 36.6%©√

- "Healthy" row (1,181 truly healthy patients
  - 0 predicted as having diabetes
  - 29 predicted as having prediabetes
  - 1,152 correctly predicted as having healthy
- Percentages:
  - Healthy → Diabetes: 0 / 1,181 = 0%
  - Healthy → Prediabetes: 29 / 1,181 ≈ 2.5%
  - Healthy → Healthy: 1,152 / 1,181 ≈ 97.5%



CONCLUSION

CONCLUSION

- Overall Assessment:
  - Good and stable multiclass baseline model (~86% accuracy).
  - Errors only between adjacent metabolic states.
- Strengths:
  - No overfitting (train≈validation).
  - High precision for Diabetes (0.99).
  - Excellent recall for Healthy (0.98).
- Weaknesses:
  - Lower recall for Prediabetes (0.63).
  - Moderate false negatives in risk classes.
- Improvement Areas:
  - Apply class_weight to increase sensitivity on risk classes.
  - Adjust Softmax thresholds.
  - Test non-linear models (RF, XGBoost, MLP).
  - Consider a 2-stage hierarchical model.

THANK YOU

Corrado Vaccaro

Senior Customer Program Manager, PMP®

corrado.vaccaro@gmail.com