Dylan Cozloff
AST06
CS 422 1001

Diabetes Prediction MLP Classification

Installs:
pip install pandas
pip install scikit-learn
pip install matplotlib
pip install seaborn

Dataset: https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

I used a diabetes prediction dataset (100,000 entries) which is a collection of medical and demographic data from patients, with their diabetes status. All features: age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. But, I only manually extracted four continuous features (age, BMI, HbA1c level, and blood glucose level) from this dataset. The binary target was whether the patient was positive or negative for diabetes. I loaded the dataset using the pandas library, in which I use the function read_csv().

Training/Test Data Splits: I used the sklearn.model_selection function train_test_split() to use a random 20% of the dataset for testing, and the remaining for training.

Model Training: I used the sklearn.neural_network function MLPClassifier(), alongside with fit() to fit the training data using a multi-layer peceptron classifier model. I ran 3 experiments with varying parameters to see any significant improvements in certain combinations. I analyzed the best model found (Model 2) in greater comparitive detail, below that analyzes the other 2.

Model 1: 3 hidden layers of 100 nodes each, relu activation function, 0.01 learning rate, and stop after 10 epochs of no improvement because it was taking very long.
- Test accuracy: 0.9543, Train accuracy: 0.9559, Test Log-loss: 0.1406

Model 2: 2 hidden layers of 100 nodes each, tanh activation function, 0.001 learning rate, 30 epochs of no improvement.
- Test accuracy: 0.9691, Train accuracy: 0.9695, Test Log-loss: 0.0945

Model 3: 1 default hidden layer, relu activation function, batch size 64, 30 epochs of no improvement.
- Test accuracy: 0.9653, Train accuracy: 0.9671, Test Log-loss: 0.1038

Model 2 Predictions: I used the sklearn.naive_bayes prediction() function to perform classification on both the training and test vectors. I also used predict_proba() to get the probability estimates for these vectors as well.

Dylan Cozloff
AST06
CS 422 1001

Model 2 Accuracy: I used the sklearn.metrics function accuracy_score() to pass the training & test data with its classification predictions to get a more precise model accuracy as the training and test accuracy were very close: **Training Accuracy** = 0.9695, **Test Accuracy** = 0.9691. As you can see the model generalized very well, accuracy dropped by only 0.04%.

Classification Report:

Test Data Model 2: (compared to NBC)

	precision	recall	f1-score	support
0	0.97 (+0.01)	1.00 (+0.01)	0.98	18292
1	0.98 (+0.13)	0.64 (+0.05)	0.78 (+0.09)	1708
accuracy			0.97 (+0.02)	20000
macro avg	0.98 (+0.07)	0.82 (+0.01)	0.88 (+0.04)	20000
weighted avg	0.97 (+0.03)	0.97 (+0.01)	0.97 (+0.03)	20000

Sensitivity (Test Data): 64% (+5%)
Specificity (Test Data): 100% (+1%)
Log Loss (Test Data): 0.09447016675422958 (-0.05)

MLP has a 13% improvement in precision, 5% in sensitivity. This would be preferred if diabetes misdiagnosis and undetection was an important factor in model usefulness. Combining 2 hidden layers of 100 nodes each with tanh seemed to be a noticable difference.

Training Data Model 2:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	73208
1	1.00	0.64	0.78	6792
accuracy			0.96	80000
macro avg	0.98	0.82	0.88	80000
weighted avg	0.97	0.97	0.97	80000

Sensitivity (Training Data): 59%
Specificity (Training Data): 100%
Log Loss (Training Data): 0.09265261494671903

Dylan Cozloff
AST06
CS 422 1001

Test Data Model 1:

	precision	recall	f1-score	support
0	0.95	1.00	0.97	18292
1	1.00	0.45	0.62	1708
accuracy			0.95	20000
macro avg	0.97	0.73	0.80	20000
weighted avg	0.96	0.95	0.94	20000

Sensitivity (Test Data): 45%
Specificity (Test Data): 100%
Log Loss (Test Data): 0.14064547874146846

Training Data Model 1:

	precision	recall	f1-score	support
0	0.95	1.00	0.98	73208
1	1.00	0.48	0.64	6792
accuracy			0.96	80000
macro avg	0.98	0.74	0.81	80000
weighted avg	0.96	0.96	0.95	80000

Sensitivity (Training Data): 48%
Specificity (Training Data): 100%
Log Loss (Training Data): 0.13630373191471234

DOWN

Dylan Cozloff
AST06
CS 422 1001

Test Data Model 3:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	18292
1	0.96	0.62	0.75	1708
accuracy			0.97	20000
macro avg	0.96	0.81	0.87	20000
weighted avg	0.97	0.97	0.96	20000

Sensitivity (Test Data): 62%
Specificity (Test Data): 100%
Log Loss (Test Data): 0.10380256909790538

Training Data Model 3:

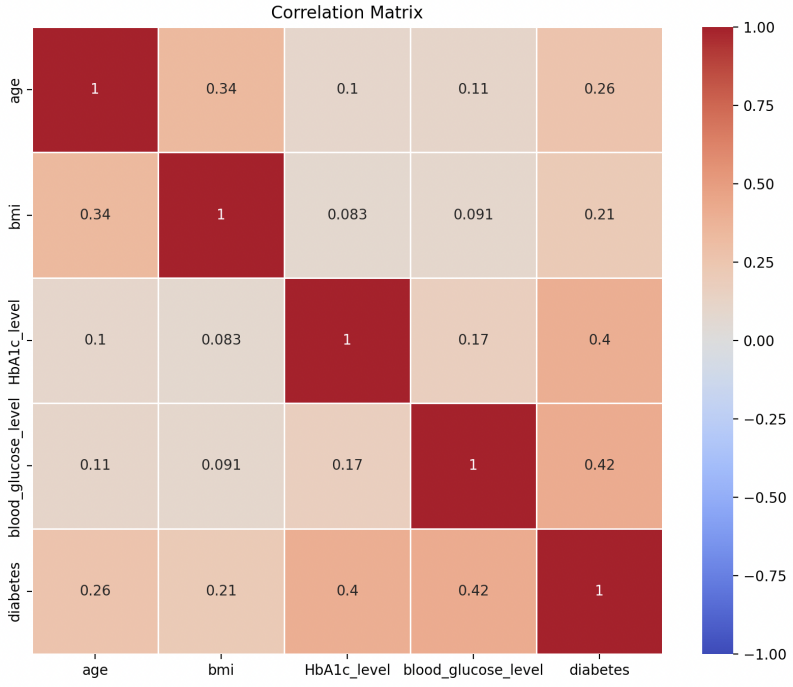
	precision	recall	f1-score	support
0	0.97	1.00	0.98	73208
1	0.97	0.63	0.77	6792
accuracy			0.97	80000
macro avg	0.97	0.82	0.87	80000
weighted avg	0.97	0.97	0.96	80000

Sensitivity (Training Data): 63%
Specificity (Training Data): 100%
Log Loss (Training Data): 0.1023394581221614

DOWN

Dylan Cozloff
AST06
CS 422 1001

Correlation Heat Map:



I used the corr() function to evaluate the dataset, and matplotlib.pyplot / seaborn to visualize the correlation matrix. As you can see there is definitely a correlation between diabetes and HbA1c_levels, blood glucose levels, age, and bmi.