Massey University

158736 Advanced Machine Learning

Assignment 1: PIMA Indians Diabetes

Luis Vieira [23012096]

1. What is the size of the dataset?

This Diabetes dataset has 768 rows and 9 columns.

2. How many features and classes are there in the dataset?

The dataset has 8 features: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction and Age. Outcome is the target column, and has two classes: (1 and 0, patient has diabetes or not respectively).

3. What can you comment about the feature data types?

All features have numerical values, with BMI and DiabetesPedigreeFunction being floats (float64), and all other are integer (int64), with Outcome having a binary numerical value.

4. How do you think the missing values in the dataset have been handled?

The apparent lack of labelled missing values can be misleading, as it seems to have missing values imputed with zeros (eg.: due to privacy, input error). It is unrealistic to find cases such as zero values for Glucose (5 or 0.65%), BloodPressure (35 or 4.56%), SkinThickness (227 or 29.56%), Insulin (374 or 48.7%), and BMI (11 or 1.43%).

5. Discuss two different ways to handle missing values in a dataset (do some reading. This is a general question, not specific to this dataset)

Two common approaches for handling missing values are deletion and imputation. Imputation techniques, such as median, kNN or multiple imputation methods, are better suited for data that is Missing Completely at Random (MCAR) or Missing at Random (MAR). Mean or median imputation is simple to apply but can sometimes introduce bias. In contrast, multiple imputation is a more robust approach, especially for MAR data, as it considers the variability across different imputations.

On the other hand, deletion (row or column wise) is more straightforward and is frequently used when only a small part of the data is MCAR, to avoid biased analysis and predictions. Deleting data can result in a considerable loss of relevant information if there is a large number of missing in proportion to the total dataset.

Deciding between imputation or deleting data should take into account the nature and quantity of missing data, with the goal of minimising bias and maintain the integrity of the dataset.

In my implementation, median imputation outperformed kNN even when using a large k of 25. I also tested a median imputation based on each class but with the exception of PCA which had identical performance all other models were worse than normal median imputation.

Sources: 161.324 Data Mining Study Guide (Chapter 3),

Vahdati, A. et al. (2024) Enhancing data integrity in Electronic Health Records: Review of methods for handling missing data. *medRxiv* https://doi.org/10.1101/2024.05.13.24307268

Ibrahim, E. et al. (2021) Handling missing and outliers values by enhanced algorithms for an accurate diabetic classification system. *Multimed Tools Appl 80*, 20125–20147. https://doi.org/10.1007/s11042-021-10727-0

6. Looking at the nature of this dataset, briefly discuss what kind of safeguards (at least two) should be followed to guarantee that the data collection and maintenance is ethical?

When working with Health datasets of this nature (PIMA Indians Diabetes), it is crucial to follow strict ethical standards to protect the patients' rights and privacy.

One fundamental safeguard is having informed consent, as patients must be clearly informed about the purpose of the data collection, how their data will be used, and risks involved. They must voluntarily agree to participate and be given the option to withdraw at any time. This is especially important in health related studies, due to the collection of personal and sensitive information such as their health and medical condition (eg.: diabetes).

Another important measure is data anonymisation, which means removing or hiding any identifying information from the dataset to ensure that individuals cannot be linked to their data. Anonymisation is especially important in health studies to prevent breaches of confidentiality and to protect participants from any potential misuse of their data. Regular reviews should be conducted to ensure that these ethical standards are consistently kept in check.

7. What percentage of data would you use for testing, and why?

A test set of 20% to 30% is commonly used, depending on the size of the dataset. In some cases, if we are using a validation set a split of 70/15/15 (train, validation, test) is often used. Given the relatively small dataset sample of 768 cases, not using a validation set, and the presence of missing data, I decided to use 20% since 10-15% showed signs of overfitting. This provides a good balance between having enough data to train, whilst keeping 20% to evaluate the models' performance.
I also used stratified splits and ensured the test set was composed only with complete cases representative of the whole dataset and classes, and prior to the imputation step to avoid data leakage.

8. Do you think data normalization is needed for this dataset? Briefly explain.

Yes, normalisation is necessary because features like Glucose and BMI have different scales, which can lead to biased model performance. Normalisation ensures that all features contribute equally to the model's learning process. I tested 3 different methods StandardScaler, MinMaxScaler, and RobustScaler from scikit-learn. Before normalising I removed one extreme outlier (SkinThickness of 99), which improved considerably the StandardScaler and MinMaxScaler performance, as these are more sensitive to outliers. MinMaxScaler showed the best overall performance for pretty much all models.

9. What is the multi-layer perceptron and how is it related to the perceptron? Briefly describe.

The multi-layer perceptron (MLP) represents an evolution beyond the basic perceptron model. The simple perceptron is limited to solving linear separable problems, whereas the MLP introduced multiple layers, including hidden layers. These allow it to model complex non-linear relationships, using non-linear activation functions, such as sigmoid or ReLU, allowing it to approximate any continuous function. The MLP is trained using a back-propagation algorithm that adjusts the weights by propagating errors backward. This iterative process allows the model to learn from its mistakes and improve its performance over time.

This makes MLP far more powerful than earlier perceptrons, making them suitable for a wide range of complex machine learning applications.

Source: Haykin, S. (2009), Neural Networks and Learning Machines, Pearson 3rd ed.. Chapter 4, pp. 122-124

10. What is the logistic regression classifier and how is it similar to and different from the perceptron? Briefly describe.

The logistic regression is a linear classifier method used for binary classification. Like the perceptron, it aims to separate the data between different classes using a linear decision boundary.

However, whilst the perceptron outputs a binary result 0 or 1, the logistic regression outputs a probability between 0 and 1, commonly using a sigmoid function to transform its output. This allows logistic regression to provide more robust and interpretable prediction and is often more robust to outliers and noise in the data.

Both models use similar training approaches, adjusting weights to minimise errors, but logistic regression uses a maximum likelihood estimation instead of the perceptron's error correction.

11. Describe the different parameters of the Multi-layer perceptron classifier.

The MLP classifier has several important parameters that impact its performance: **Hidden Layer Sizes** determine the architecture by controlling the number of neurons in each layer; **Activation** functions like ‘relu’ or ‘tanh’ introduce non-linearity into the model; **Solver** is the algorithm used for weight optimisation, such as ‘adam’ or ‘lbfgs’; **Alpha** is the regularization parameter that helps to avoid overfitting; and **Learning Rate** decides the step size for the weight updates, which affects convergence.

Source: Scikit-learn documentation.

12. In your code, run the Multi-layer perceptron with different values to the solver parameter. What are your observations? Do all the solvers work equally good on this dataset? If not, why?

I initially tested ‘adam’ against ‘sgd’, with ‘adam’ performing significantly better, likely due to its adaptive learning rate making it more effective for or ‘sgd’ needed further tunning and due to the small dataset size and non-linearity. Thus, following the Scikit-learn documentation suggestion, I then tested ‘lbfgs’ for this small dataset. While ‘lbfgs’ showed similar performance to ‘adam’, ‘adam’ still outperformed ‘lbfgs’, particularly when using the ‘tanh’ activation function. This suggests that ‘adam’ is generally more robust across different scenarios.

Source: Scikit-learn documentation

13. Vary the alpha parameter of the Multi-layer perceptron. What value range gives the optimal results for the classifier?

Following the solver tests, using ‘adam’ as solver, ‘hidden_layer_sizes’: (100, 50), ‘activation’: ‘tanh’ and a prior best alpha of 1, I performed hyperparameter tuning for a range around this value of ‘alpha’: np.linspace(0.01, 5, 20) using a 10-fold GridSearchCV (30 iterations) and obtained an **‘alpha’: 1.3231578947368423**.

The MLP model accuracy rose about 1% to 81%. I was then able to further improve the model accuracy to 82%, by performing a GridSearchCV (10-fold) using SelectFromModel feature selection (max features = 5).

14. Briefly describe the parameters of the SVM and logistic regression classifiers.

The SVM and Logistic Regression classifiers each have important parameters that influence their performance. In the case of SVM, **C** manages the balance between maximising the margin and minimising classification errors, where higher values allow the model better fit the training data. The **kernel** parameter decides how data is transformed, with options like 'linear' or 'rbf' allowing the SVM to capture non-linear relationships.

In Logistic Regression, **C** also controls regularisation strength, where smaller values increase regularisation. The **penalty** parameter specifies whether to use L1/ L2 regularisation. The **solver** parameter determines the optimisation algorithm used, affecting how efficiently the model converges.

15. Briefly discuss the relationship between the number of features and the computational complexity of SVM and logistic regression classifiers.

The computational complexity for both SVM and logistic regression increases with the number of features. Logistic regression, being a linear model, is typically less affected with the number of features, but as the feature count increases, the optimisation process can still become more computationally expensive, especially with solvers like ‘lbfgs’.

Whereas SVM, especially in the case of non-linear kernels like RBF, complexity can increase more quickly than logistic regression with the number of features, as it can struggle with decision boundaries in higher-dimensional space.

Both algorithms may require more time and memory for training, potentially leading to the "curse of dimensionality". Thus, models can benefit from feature and dimensionality reduction techniques to reduce this complexity in the preprocessing steps.

16. Is there any observed difference in the training times of the SVM and logistic regression classifiers? briefly discuss the reasons for your observations.

Yes, there is often a difference in training times but this dataset is relatively small I did not notice any considerable differences. As I mention before, the training times of SVM and logistic regression can differ considerably. Logistic regression often trains faster, especially for large datasets, due to its relatively simple optimisation problem. SVM, especially with non-linear kernels, can be computationally more intensive. In practice, the actual performance will vary based on the specific dataset, implementation and parameters used, such as the complexity of the optimisation algorithms used by these models and decision boundaries they are trying to learn.

17. What is the difference of macro and micro averages of precision, recall, F1 values?

Macro and micro averages are methods for aggregating precision, recall, and f1-scores across multiple classes. Macro averaging computes the metric independently for each class and then takes the average, treating all classes equally, which is particularly useful in imbalanced datasets. Micro averaging, on the other hand, aggregates the contributions of all classes to compute the metric, giving more weight to classes with more samples. Micro averaging is better suited when overall performance across all classes is of interest, while macro averaging provides insight into model performance on minority classes.

18. When you observe the results returned by sklearn, you would notice that precision, recall and F1 results are reported per class, and accuracy results are only for the whole dataset. Why is this difference? Briefly explain.

Precision, recall, and f1-score are reported per class because they offer insights into how well the model performs for each individual class, which is crucial in multi-class or imbalanced datasets, as the case of sensitive medical data. These metrics help to understand how well each class is being predicted, beyond just overall accuracy. While accuracy, is a global metric that summarises the model's overall correctness across all classes. It gives a general sense of performance, but it can be misleading in imbalanced datasets, hence the importance of class-specific metrics like precision, recall, and f1-score.

19. A colleague suggested that you need to identify whether one classifier is significantly better than the other. With the results you obtained above, do you think you can conduct a significance test? Briefly explain.

Yes, conducting a significance test is essential to determine if the differences seen between classifiers are truly significant or simply random chance. In machine learning, statistical tests compare performance metrics, like accuracy or f1-score, across multiple folds or data splits, offering a more robust evaluation than simply looking at raw metrics.

When comparing classifiers like Logistic Regression, SVM, and MLP, the initial metrics provide an overview of their effectiveness. However, to rigorously establish whether one model significantly outperforms the others, statistical tests are crucial. McNemar’s test, for instance, is often used for comparing two classifiers on a single test set by focusing on their misclassification rates, ensuring that any observed differences are indeed meaningful.

20. Briefly describe two significant tests (other than student t-test).

Two significant tests for comparing classifiers are the 5x2 Cross-Validation combined *F*-test and McNemar's test. The 5x2 CV combined *F*-test compares two classifiers by running five iterations of 2-fold cross-validation, generating a distribution of different performances that are tested for significance. McNemar's test is a non-parametric test used on paired nominal data, often employed to compare the performance of two classifiers by focusing on the misclassifications that are different between them. These tests help deciding whether the observed differences in classifier performance are statistically significant.

Source: mlxtend 5x2cv combined *F* test

https://ecs.wgtn.ac.nz/foswiki/pub/Groups/ECRG/StatsGuide/Significance%20Testing%20for%20Classification.pdf

Dietterich, T. (1998) Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, *Neural Computation*, 10 (7) https://doi.org/10.1162/089976698300017197