CSC461 Intro. To Machine Learning

Instructor: Dr Wassim Katerji

Spring 2024

Georgio Ghnatos 202106606

Joy Andraos 202100325

Muhammad El Wazir 202204951

Assignment 1: Liver Disease Prediction

Loading & Exploring

Step 1: Data Loading and Initial Exploration

Data Acquisition
The dataset, identified by the ID 1480, was downloaded from OpenML using the custom download_openml_dataset function. This dataset comprises various biochemical parameters and demographic information of patients, which are potential indicators of liver disease.

Initial Assessment
Upon retrieval, the dataset was examined for structure, completeness, and preliminary insights. The data columns were renamed for clarity and interpretability as follows:

Age
Gender
Total Bilirubin
Direct Bilirubin
Alkaline Phosphatase
Alanine Aminotransferase
Aspartate Aminotransferase
Total Proteins
Albumin
Albumin and Globulin Ratio
Class
The first few records of the dataset were printed out to confirm successful loading and correct labeling of columns.

Data Cleaning and Transformation
Handling Missing Values
The data was iteratively processed to handle missing values. For numerical attributes, missing values were substituted with the mean of the respective column. Categorical attributes were treated similarly, with the mode replacing missing entries.

Data Type Conversion
The categorical variable 'Gender' was encoded numerically to facilitate its use in algorithmic processing.

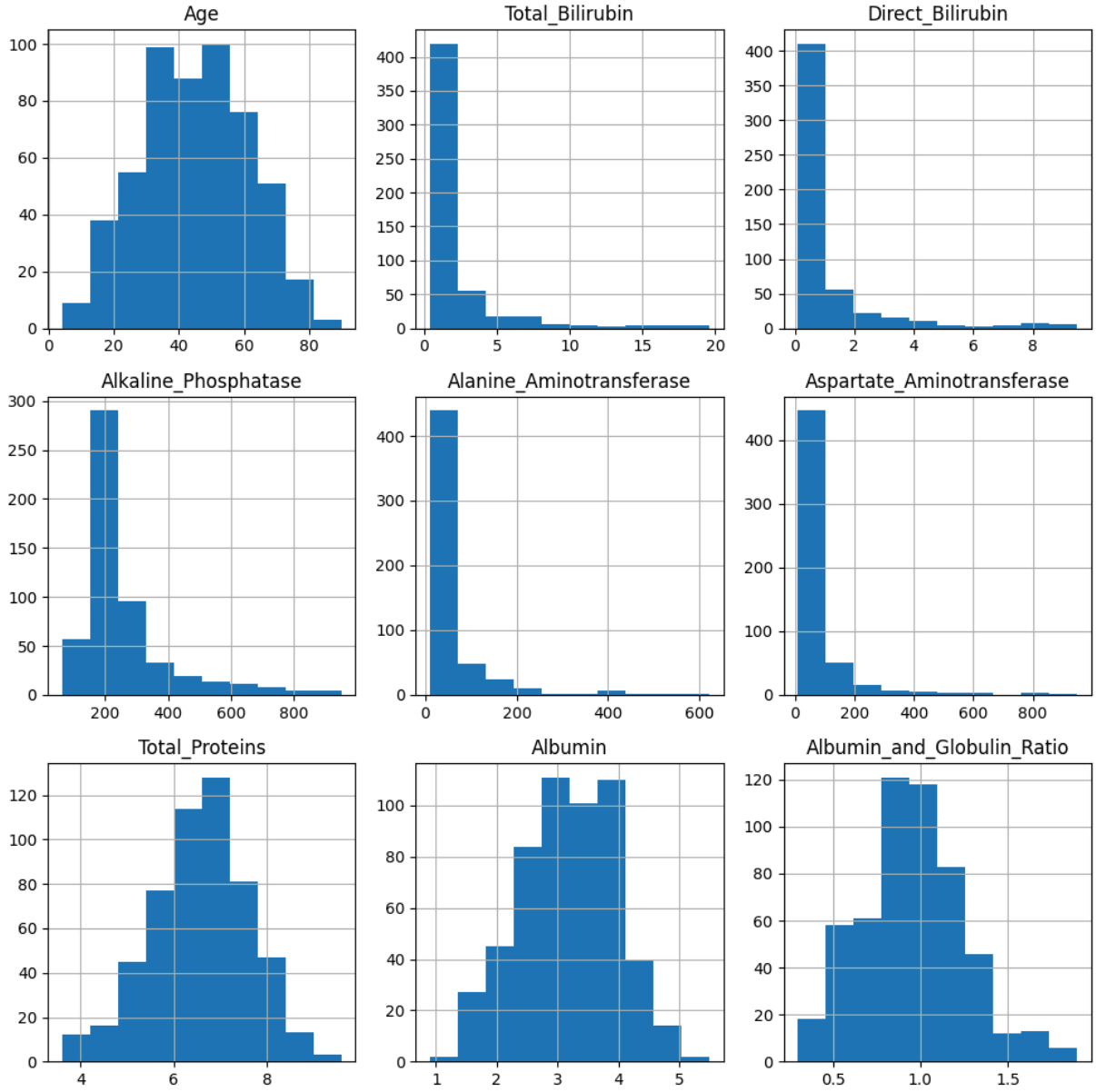
Outlier Detection and Removal

Outliers can skew the results of data analysis and model prediction. Therefore, a z-score based filtering method was employed to identify and remove outliers from the dataset and overall 47 outliers were removed.

Step 2: Exploratory Data Analysis (EDA)

Data Distribution Examination

Histograms were generated for each feature to visualize their distribution across the dataset. This step is crucial in identifying patterns, anomalies, skewness, and potential outliers in individual features.



Age: The distribution is somewhat right-skewed, suggesting a younger population, with a significant concentration of individuals around 30 to 60 years old. This could suggest that middle-aged individuals are more commonly represented in this dataset, which might be reflective of the population most affected or most frequently tested for liver disease.

Total Bilirubin, Direct Bilirubin, Alkaline Phosphatase, Alanine Aminotransferase, Aspartate Aminotransferase: These distributions are highly right-skewed, indicating that most of the values are low with a few individuals having very high levels. Such a pattern often indicates that a small subset of patients has significantly abnormal liver function tests, which could be associated with severe liver disease or other hepatic insults.

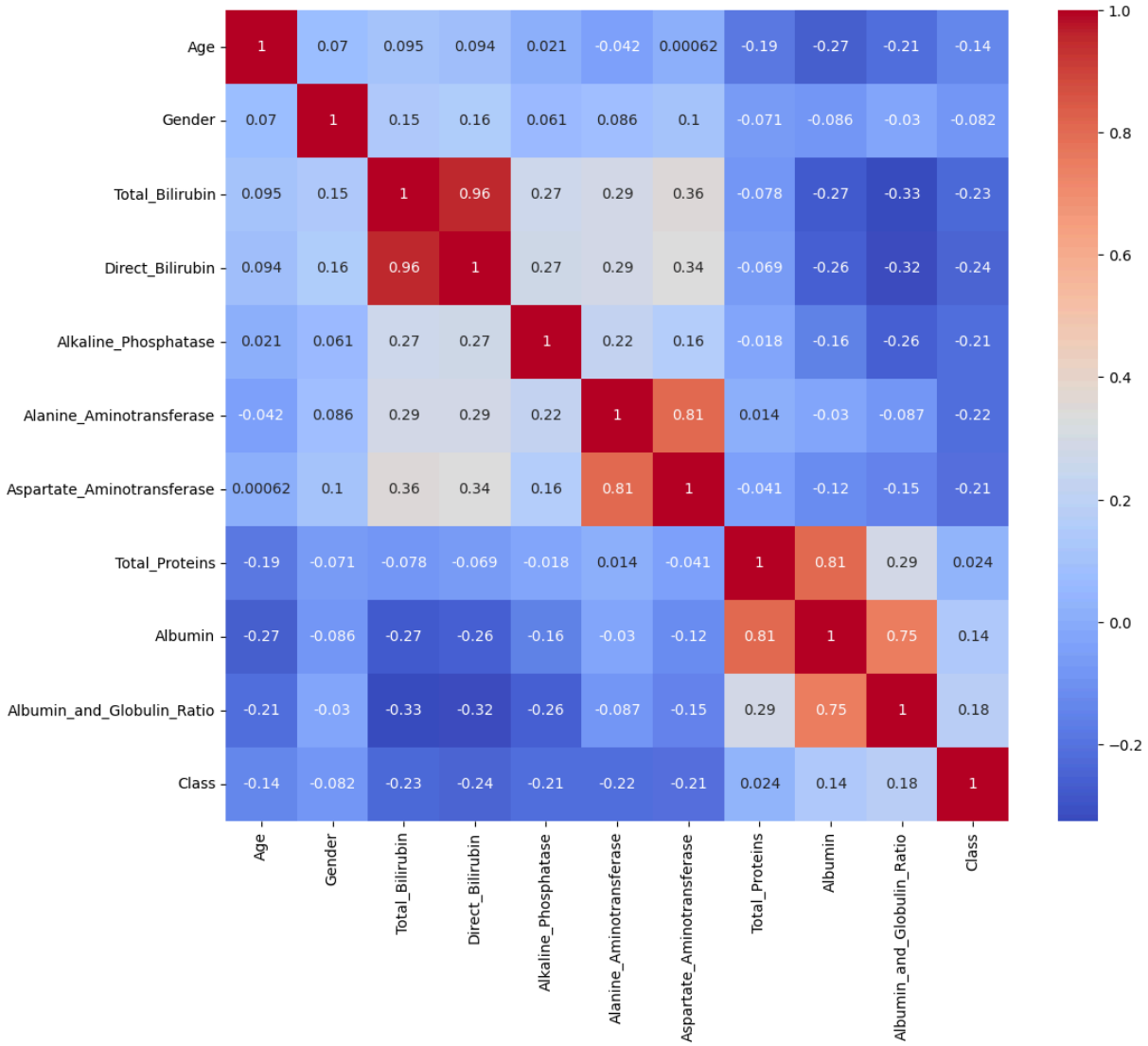
Albumin and Globulin Ratio: This histogram is also right-skewed but shows a less extreme skew than the liver enzymes. The Albumin and Globulin Ratio is a marker often used to assess liver function, with lower values potentially indicative of liver disease. The skewness indicates that most patients have ratios within a normal range, but there's a long tail on the lower end that could represent individuals with liver pathology.

Albumin: The distribution of Albumin seems relatively normal with a slight right skew. Albumin levels are an important measure of liver function, as Albumin is produced by the liver. Most individuals have Albumin levels within the normal range, but there are a few with high levels.

Correlation Analysis

A heatmap of the correlation matrix was created, allowing for the visualization of pairwise relationships between features. This step is instrumental in identifying multicollinearity and potential predictors for the outcome variable 'Class', which indicates the presence of liver disease.

The heatmap indicated several notable correlations:
A strong positive correlation between Aspartate Aminotransferase and Alanine Aminotransferase, suggesting a potential link between these enzymes and liver function.
Total Bilirubin showed a strong positive correlation with Direct Bilirubin, which is expected given their biological relationship.
Albumin showed a significant positive correlation with Total Proteins, another expected relationship due to albumin being a major component of total serum proteins.



Feature Selection

Using the SelectKBest method with the chi2 scoring function, features were ranked based on their statistical significance with respect to the outcome variable. The analysis revealed that Aspartate Aminotransferase and Alanine Aminotransferase were among the top features, which aligns with medical literature that suggests their importance in diagnosing liver diseases.

Model Building

Step 1: load data, define X and y, split sets

The first task is to load the dataset from the given path using pd.read_csv() from pandas, separating the feature columns (X) and the output column (y) from the dataset, and splitting our dataset into training and testing sets using train_test_split() from scikit-learn. Inside the function, we specify the test_size parameter which determines the proportion of the dataset to include in the test split (we chose 20% implying that 80% is used for training, since it is common practice that was proven to be effective) and the random_state parameter which sets the random seed for reproducibility (we chose 23 but this choice is arbitrary and doesn't really have a justification; the point is having a seed for reproducibility which means that the same random numbers will be generated every time we run the code, with the same split).

Step 2: metrics for evaluation

The metrics we used to evaluate our models are the following:

- Accuracy: Ratio of correctly predicted observations to total observations, may not be useful for imbalanced datasets. **(Accuracy=True Positive + True Negative) / Total Population)**

- Precision: measures the proportion of true positive predictions out of all positive predictions made by the model. **(Precision=True Positives/True Positives + False Positives)**

- Recall: measures the proportion of true positive predictions out of all actual positive instances in the dataset. **(Recall=True Positives/True Positives + False Negatives)**

- F1-Score: Weighted average of Precision and Recall, useful for imbalanced datasets. Precision measures correct positive predictions, while Recall measures correct positive instances as mentioned earlier. **(F1-score = 2 x Precision x Recall) / (Precision + Recall)**

Step 3: Logistic Regression

Starting with logistic regression, the most basic algorithm for classification tasks, we got the following results:

```
Logistic Regression results

Logistic Regression model accuracy (in %):  68.51851851851852
Precision:  0.7682926829268293
Recall:  0.8076923076923077
f1 score:  0.7875
```

The accuracy is about 68.52%, indicating that the model is making correct predictions for about 68.52% of the cases. The precision shows that the model guessed correctly 76.83% of the correct labels, while recall shows that the model recognized 80.77% of all the dataset. Overall, this leads to a 0.7875 F1-score meaning there's a good balance between precision and recall. However, there's room for improvement in terms of all metrics, which can be achieved using optimization algorithms like gradient descent.

Step 4: KNN

Moving on to KNN, we start by creating a KNN classifier object using scikit-learn, and specify 70 nearest neighbors to consider when making predictions. This is a significant parameter in KNN. After several trials, the choice 70 appeared to be the most efficient since it is large enough to prevent both overfitting and underfitting. Here are the results:

```
KNN results

KNN model accuracy (in %):  73.14814814814815
Precision:  0.7425742574257426
Recall:  0.9615384615384616
f1 score:  0.8379888268156426
```

KNN shows a higher accuracy compared to Logistic Regression, with a value of 73.15%. While it has a high recall score of 96.15%, its precision score is way lower with a percentage of 74.26%, implying a possibility of classifying non-diseased cases as diseased.

Step 5: Naïve Bayes

Next, we chose to work on Gaussian Naïve Bayes since our data follows a normal distribution. Based on Bayes' theorem, it makes the 'naive' assumption that features are independent. We initialized the classifier with default parameters and got the following results:

```
Naive Bayes results

Naive Bayes model accuracy (in %):  50.92592592592593
Precision:  1.0
Recall:  0.32051282051282054
f1 score:  0.48543689320388356
```

Naive Bayes has the lowest accuracy among the models so far with a score of 50.92%. One remarkable thing is its perfect precision score of 100%, meaning no false positives. However, its recall is quite low with a score of 32%, meaning it fails to identify many positive instances. This shows that the model only predicts positive cases when it is 100% confident which could lead to its failure of detecting cases where intervention is needed.

Step 6: SVM

Finally, for the SVM, we chose to scale our features to the range [0,1] using Min-Max scaling which calculates the minimum and maximum of each feature and scales them accordingly. Next, we create an instance of an SVM classifier. Since SVM requires tuning several hyperparameters, we set up a grid of these variables to find the best values using GridSearchCV and grid_search.best_params. The hyperparameters are the regularization parameter C, the kernel function type, the kernel coefficient gamma and the class_weight to handle class imbalance. Here are the results:

```
SVM results

Best Parameters: {'C': 7, 'class_weight': None, 'gamma': 'scale', 'kernel': 'rbf'}
SVM model accuracy (in %): 73.14814814814815
              precision    recall  f1-score   support

           1       0.77      0.90      0.83        78
           2       0.53      0.30      0.38        30

    accuracy                           0.73       108
   macro avg       0.65      0.60      0.61       108
weighted avg       0.70      0.73      0.70       108

Precision:  0.7692307692307693
Recall:  0.8974358974358975
f1 score:  0.8284023668639053
```

The best values for the hyperparameters are:

- C = 7, which is considered a strong regularization. This makes the model less sensitive to outliers and noise, preventing overfitting
- Class_weight = None, the classes have equal weights during training (no imbalance in data)

- Gamma = scale, the algorithm calculates the gamma value based on the inverse of the input data's variance, meaning gamma is proportional to variance of the space feature and adapts to the dataset's characteristics
- Kernel = rbf (Radial Basis Function), which works on several datasets types such as non-linear relations between features and target variables

SVM shows the highest accuracy among the models, along with KNN, with a score of 73.14%. It also has balanced values of precision (76.92%), recall (89.74%), and F1 score (0.82) meaning it performed well. This indicates that it can efficiently identify true positive instances while keeping false positives relatively low.

Conclusion

SVM appears to be the best-performing model out of the 4 we trained, considering both accuracy and the balance between precision and recall.

In our scenario, where features such as Aspartate Aminotransferase, Alanine Aminotransferase, and Alkaline Phosphatase show a variety of degrees of importance, the complex relationships among these features require models capable of capturing complex patterns. SVM and KNN, known for their adaptability to high dimensional feature spaces, are particularly suited for such tasks, thus explaining their higher accuracy and performance over Logistic Regression and Naive Bayes in our dataset.

In terms of F1-score, SVM surely outperforms KNN where KNN has a higher possibility of classifying non-diseased cases as diseased, while SVM proved to efficiently identify true positive instances and keeps false positives relatively low.

Nevertheless, all models have room for optimization techniques in order to achieve better results. Also, the observed average accuracies in all 4 tuned models may be due to the characteristics of our dataset. Therefore, improving the dataset could also lead to better results.