Big Data Project - Diabetes Diagnosis

Stan Eduard-George, Sîrbu Oana-Adriana
Group 407 AI

1 Short Dataset Description

This project aims to analyze and predict readmission rates of diabetic patients, using a large dataset collected from 130 US hospitals over a ten-year period (1999-2008). The dataset, sourced from the UCI Machine Learning Repository (`https://archive.ics.uci.edu/dataset/296/diabetes+130-us+hospitals+for+years+1999-2008`), contains rich information on patient demographics, medical diagnoses, laboratory results, medications, and hospitalization records. The goal is to apply various preprocessing techniques and dimensionality reduction methods to enhance our understanding and predictive modeling of diabetes diagnosis.

2 Exploratory Data Analysis

We start by loading the dataset and performing an initial inspection to understand its structure and content. The first 10 rows of data look like this:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	...
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1	...
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3	...
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2	...
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2	...
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1	...
5	35754	82637451	Caucasian	Male	[50-60)	?	2	1	2	3	...
6	55842	84259809	Caucasian	Male	[60-70)	?	3	1	2	4	...
7	63768	114882984	Caucasian	Male	[70-80)	?	1	1	7	5	...
8	12522	48330783	Caucasian	Female	[80-90)	?	2	1	4	13	...
9	15738	63555939	Caucasian	Female	[90-100)	?	3	3	4	12	...

10 rows × 50 columns

Figure 1: Example of the first 10 rows from the dataset

We observe that the dataset contains several columns with missing values, as the first thing we notice is that NaN values for feature ‘weight‘ are represented by ”?”. Secondly, feature ‘age‘ is represented as a series of lists with values from 0 to 100 with step 10. We may consider converting ‘age‘ to integer. We need to handle these appropriately to ensure the quality of our analysis.

We continued with printing the unique values that a feature can take along with their counts. All the results are stored in the "unique_values_counts.txt" from the **conclusions** folder that we attached to this report. We separated the features such that the ones with too many unique values will be ignored.

```
Columns with too many unique values:
encounter_id 101766
patient_nbr 71518
```

Figure 2: Output of printing columns with too many unique values

These columns are identifiers and have a high cardinality, meaning they have a large number of unique values. High cardinality columns are typically unique identifiers like encounter IDs or patient numbers, which are not useful for predictive modeling or statistical analysis because they do not provide informative features about the data. This is why we decided to eliminate them when training the models.

Next, we see that the values of the label (column "readmitted") are imbalanced.

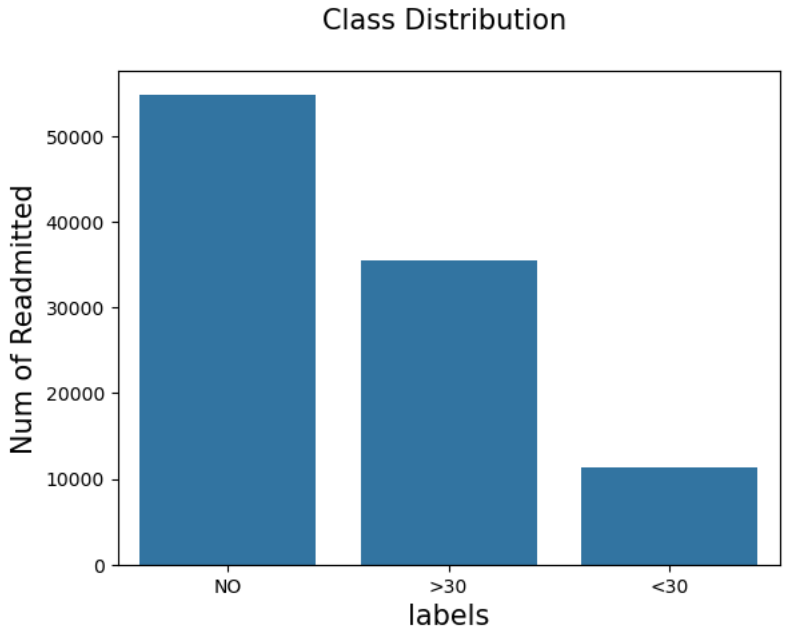


Figure 3: Emphasizing the imbalance found for the "readmitted" column

We can see that we have 3 categories and we want this to be a binary classification problem. Will the patient be readmitted? yes/no Therefore, we will group the "> 30" and "< 30" into the same 'YES' category. This will later be encoded into 0 and 1.

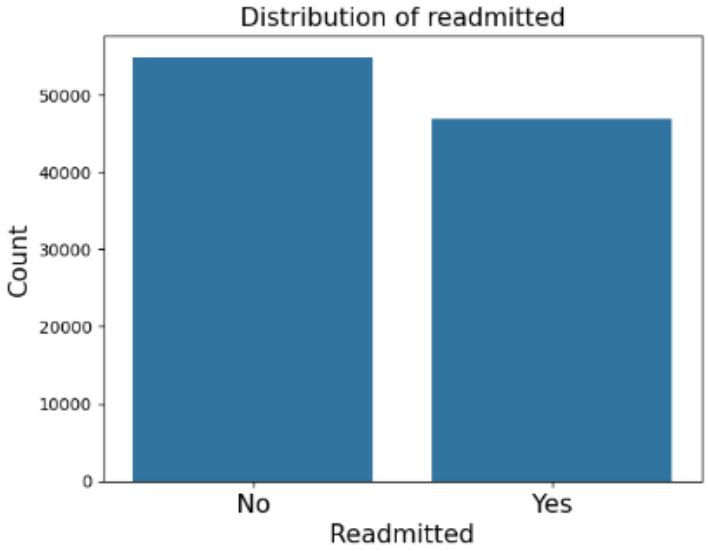


Figure 4: The new labels distribution for the "readmitted" column

We want to further inspect the "weight" column, as it appear to have many missing values, described by "?".

```
count     101766
unique        10
top            ?
freq       98569
Name: weight, dtype: object
```

Figure 5: The description of the "weight" column values

Due to the fact that from 101766 instances, the most frequent one is indeed "?", having 98569 values, we will drop this column in one of the preprocessing methods.

Usually gender is really important when determining a medical diagnose, as the health parameter varies for females and males. There were some people whose gender was unknow, and we dropped these rows as there are only 3 instances.

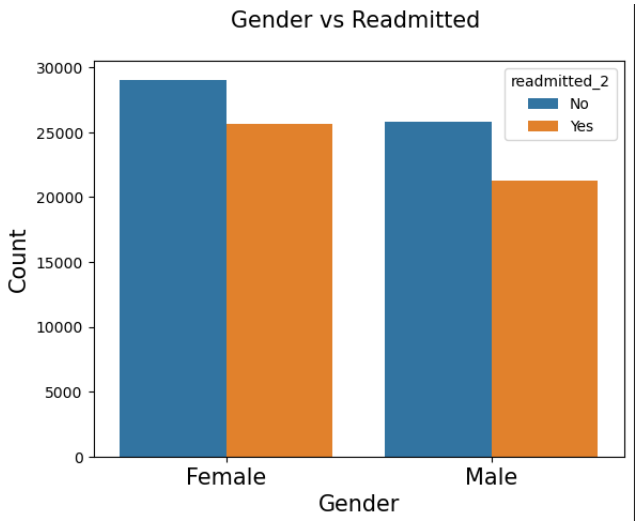


Figure 6: The comparison between gender and readmitted values

We also visualized the distribution of the gender variable and compares it across the readmitted variable. We concluded that being a female or a male does not influence the readmission possibility. Age could be a really crucial factor to the final prediction. We will inspect its distribution.

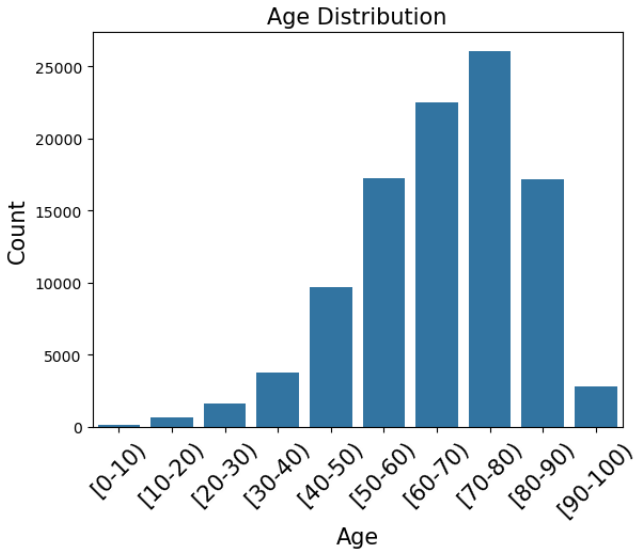


Figure 7: The distribution of the "age" column values

As expected, elderly people have a higher change of having health problems. Therefore, this column may really help during our predictions.

For the "payer_code" it appears to be almost the same situation as for the "weight" column. It will also be dropped, otherwise it will introduce noise.

We also visualized the relationship between the time stayed in the hospital (time_in_hospital) and readmission status.

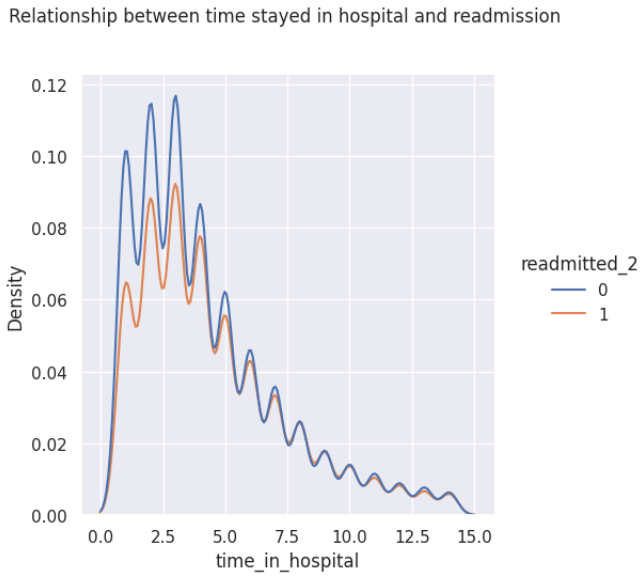


Figure 8: The relationship between time spend in hospital and the readmission status

The similarity in the shapes of the two curves suggests that, on average, the hospital stay duration alone

might not be a strong distinguishing factor for predicting readmission.

The columns 'admission_type_id', 'admission_source_id', 'discharge_disposition_id' may not directly provide clinical information about the patient's condition, treatment, or health outcomes, but they can serve as important contextual features. For example, the type of admission (e.g., emergency vs. elective) might impact the severity of the patient's condition or the urgency of treatment. Therefore we will keep them.

We can say that the number of lab procedures doesn't add much value to our training process (demonstrated by the figure below).

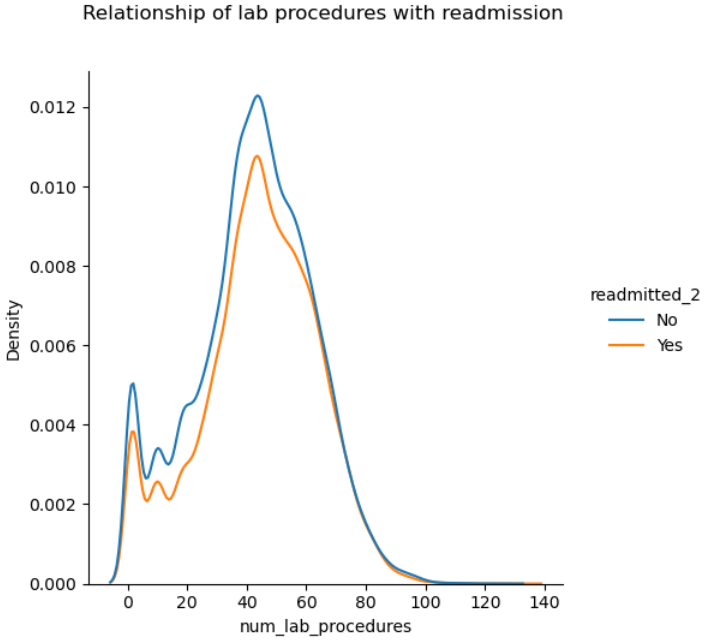


Figure 9: The relationship between lab procedures and the readmission status

We will examine another interesting feature, namely the number of emergency visits.

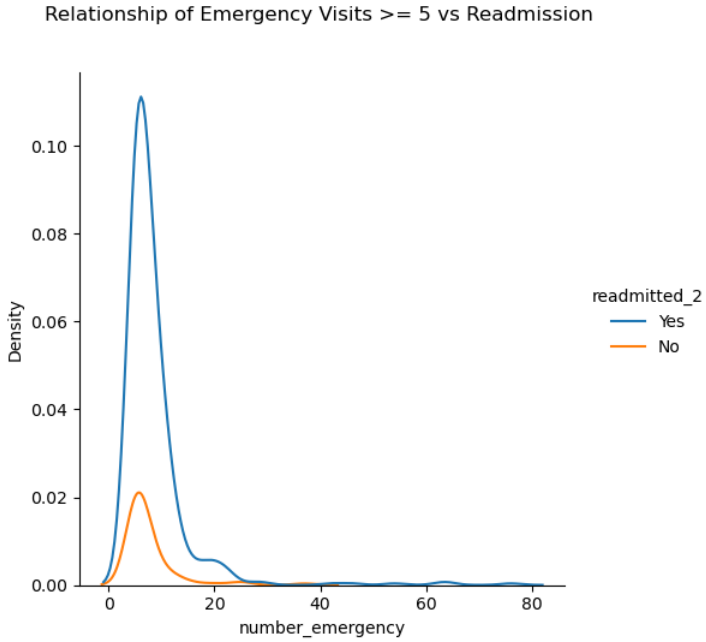


Figure 10: The relationship between the number of emergency visits (≥ 5) and the readmission status

We observed that the majority of the patients that have near 10 emergency visits and are more likely to be readmitted to hospital. We can conclude that, if the number of emergency visits increases, the patient most likely will be remitted to the hospital, so this is a relevant feature.

We further analysed the columns "diag_1", "diag_2" and "diag_3".

```
num_question_marks = (raw_data['diag_1'] == '?').sum()
percent_question_marks_diag_1 = (num_question_marks / num_instances) * 100

print("Number of values containing '?' in the 'diag_1' column:", num_question_marks)
print("Percentage of '?' values in the 'diag_1' column:", percent_question_marks_diag_1)
```

```
Number of values containing '?' in the 'diag_1' column: 21
Percentage of '?' values in the 'diag_1' column: 0.020636184074761945
```

```
num_question_marks_2 = (raw_data['diag_2'] == '?').sum()
percent_question_marks_diag_2 = (num_question_marks_2 / num_instances) * 100

print("Number of values containing '?' in the 'diag_2' column:", num_question_marks_2)
print("Percentage of '?' values in the 'diag_2' column:", percent_question_marks_diag_2)
```

```
Number of values containing '?' in the 'diag_2' column: 358
Percentage of '?' values in the 'diag_2' column: 0.35179780470308464
```

```
num_question_marks_3 = (raw_data['diag_3'] == '?').sum()
percent_question_marks_diag_3 = (num_question_marks_3 / num_instances) * 100

print("Number of values containing '?' in the 'diag_3' column:", num_question_marks_3)
print("Percentage of '?' values in the 'diag_3' column:", percent_question_marks_diag_3)
```

```
Number of values containing '?' in the 'diag_3' column: 1423
Percentage of '?' values in the 'diag_3' column: 1.3983471399231548
```

Figure 11: Inspecting the columns regarding diagnosis

We can easily see that only 0.02%, 0.35% and 1.39% respectively of datapoints contain '?'. This values will be imputed by using the most common diagnostic.

```
Value counts for 'examide' column:
examide
No    101763
Name: count, dtype: int64
```

Figure 12: Inspecting the "examide" column

We will also drop the "examide" column as it only contains the same value for each instance, so it won't help in the training process.

Same thing happened for "acetohexamide" and "citoglipton" columns.

More columns will be dropped in the preprocessing phase, as it appears that the vast majority of medicines are not given to the patients. This fact is justified by the following graphs:

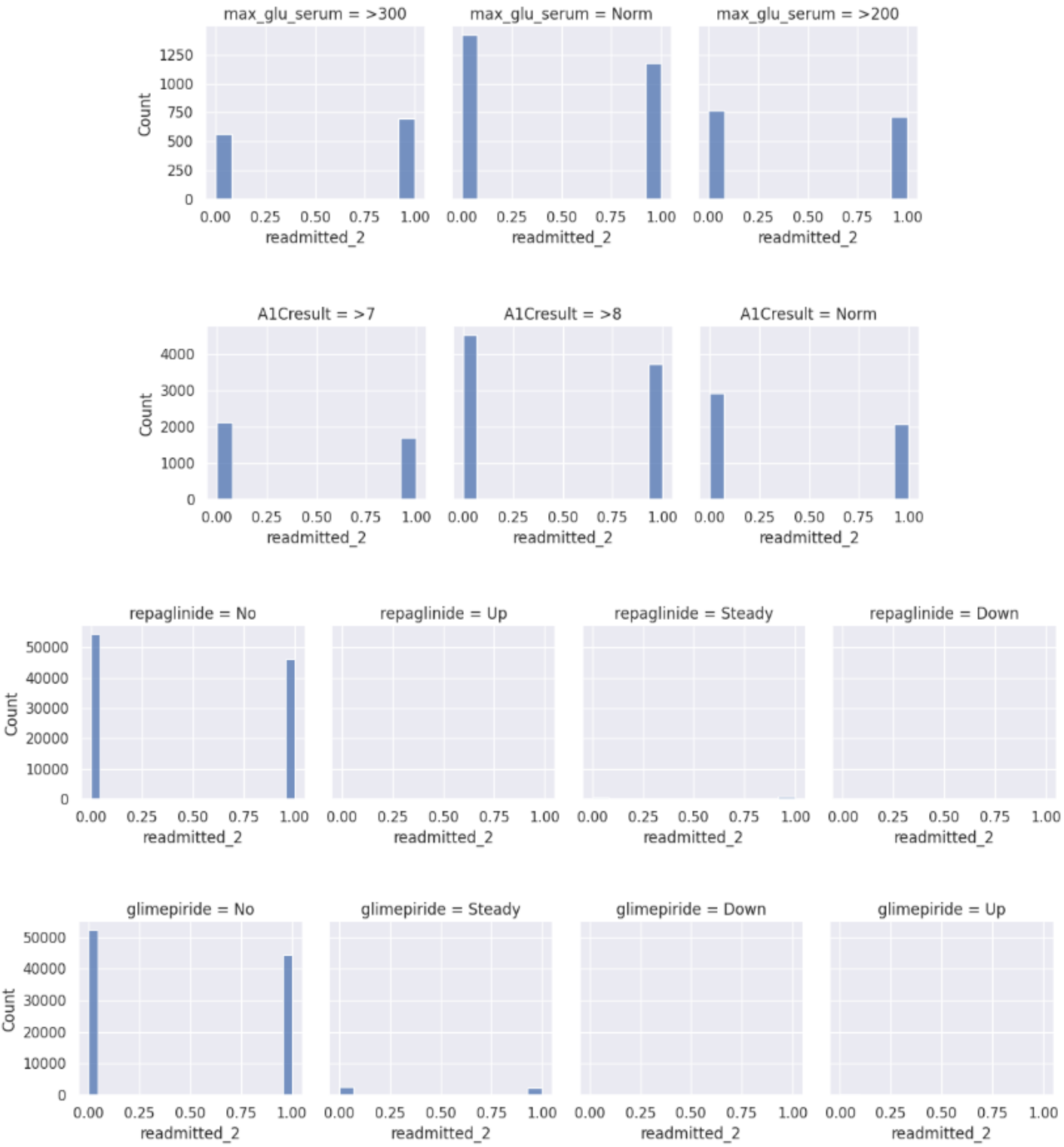


Figure 13: Correlation between medicines prescribed and the readmission status

We double checked that there are no more columns having missing columns or containing irrelevant data for our purpose and then we proceeded with the actual data preprocessing.

3 Data Preprocessing

As stated in the previous section, the dataset contains several columns with missing values. We need to handle these appropriately to ensure the quality of our analysis. Therefore, we dropped the problematic columns using:

```
df = df.drop(columns=['encounter_id', 'patient_nbr', 'weight', 'payer_code', 'medical_specialty',
'readmitted'])
```

We also removed the columns related to some medications that do not offer significant information for the analysis, as previously described.

```
df.drop(columns=['acetohexamide', 'tolbutamide', 'troglitazone', 'tolazamide', 'examide',
'citoglipton', 'glipizide-metformin', 'glimepiride-pioglitazone', 'metformin-rosiglitazone', 'metformin-p
inplace=True)
```

We defined a function for a simple imputation method for replacing missing values represented by question marks with the mode of each respective column. This was applied for the 'diag_1', 'diag_2', 'diag_3' columns.

Furthermore, we also defined the **encode_object_features** function encodes all categorical features in the dataset into numerical values using label encoding. This transformation is essential for preparing the data for machine learning algorithms, which typically require numerical input. The categorical columns discovered in the EDA part were these:

categorical_features = ['race', 'gender', 'age', 'admission_type_id', 'discharge', 'admission_source_id', 'diag_1', 'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'glipizide', 'glyburide', 'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'insulin', 'glyburide-metformin', 'change', 'diabetesMed', 'readmitted_2']

We iterated over each categorical feature and applies label encoding to transform the values into numerical codes.

As a summary, we ensured that the dataset is clean, standardized, and ready for machine learning models. The **preprocess_data** function handles missing values, drops irrelevant columns, and standardizes specific features, while the **encode_object_features** function transforms categorical variables into numerical format suitable for further analysis and modeling.

4 Dimension Reduction Methods

For the purpose of this project, we computed data with a simple preprocessing method (simple encoding) and also with three other dimesionality reduction methods:

- Simple encoding - just encoding the object-typed features.
- PCA - apply Principal Component Analysis over the encoded features.
- Random Projection - apply a dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional subspace using randomly generated matrices.
- UMAP (Uniform Manifold Approximation and Projection) - nonlinear technique for visualizing high-dimensional data in lower dimensions, capturing both local and global structures effectively.

5 Models trained

On the preprocessed data (with/without specific dimension reduction methods) we trained 5 distinct models:

- **K-nearest neighbors (KNN)** - predicts the label of a data point by considering the labels of its nearest neighbors in the dataset.
- **Gaussian Naive Bayes** - calculates the probability of a data point belonging to a particular class based on the features' probabilities and assuming independence between features.
- **Random Forest Classifier** - builds multiple decision trees during training and outputs the class that is the mode of the classes.
- **XGB Classifier** - an implementation of gradient boosting algorithm that constructs an ensemble of decision trees sequentially, aiming to minimize the prediction error by combining the predictions from multiple weak learners.
- **CatBoostClassifier** - a gradient boosting algorithm optimized for handling categorical features, using new techniques such as oblivious trees (type of decision trees where each node makes a decision based on the value of only one feature, without considering interactions between features) to enhance performance.

6 Results

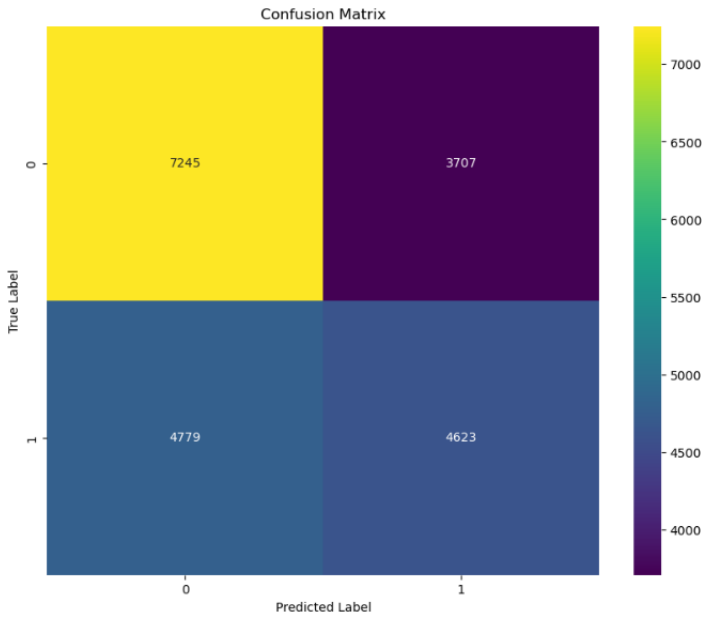
I will present the results achieved by each model across all dimension reduction methods employed. The analysis will start with training the models on data preprocessed using a basic label encoder.

6.1 Simple encoded data

For all the models a Grid Search was conducted in order to tune the models' hyperparameters.

6.1.1 KNN

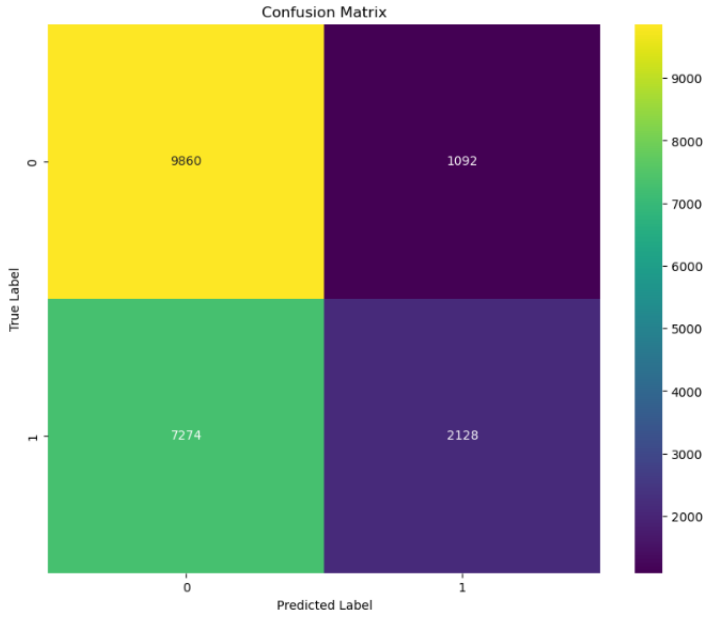
- Confusion matrix



- Best parameters: 'n_neighbors': 7, 'weights': 'distance'
- Accuracy: 0.58, Macro avg F1-score: 0.58

6.1.2 Gaussian NB

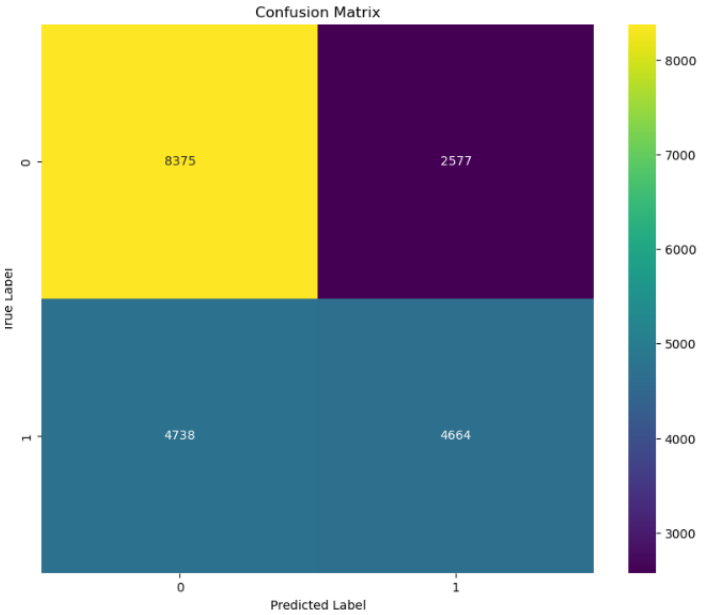
- Confusion matrix



- Can't perform Grid Search on it;
- Accuracy: 0.59, Macro avg F1-score: 0.52

6.1.3 Random Forest

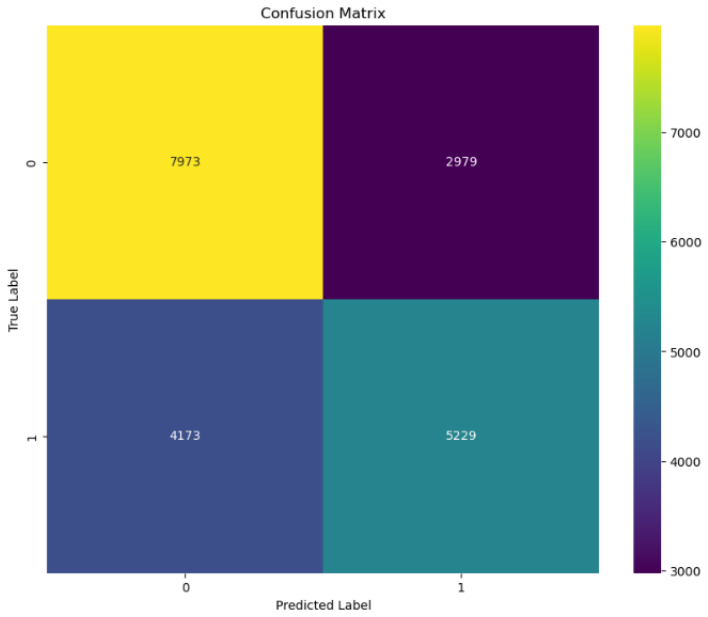
- Confusion matrix



- Best parameters: 'max_depth': 15, 'n_estimators': 200
- Accuracy: 0.64, Macro avg F1-score: 0.63

6.1.4 XGB Classifier

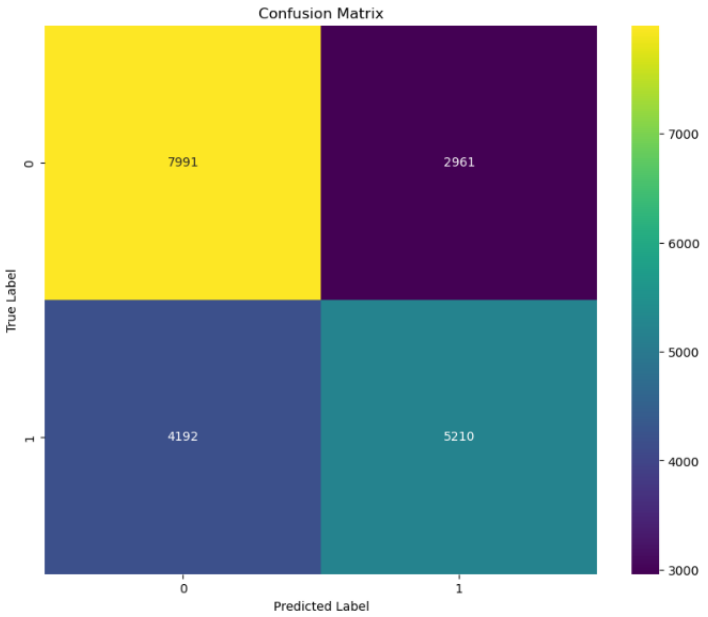
- Confusion matrix



- Best parameters: 'learning_rate': 0.1, 'n_estimators': 200
- Accuracy: 0.65, Macro avg F1-score: 0.64

6.1.5 CatBoostClassifier

- Confusion matrix

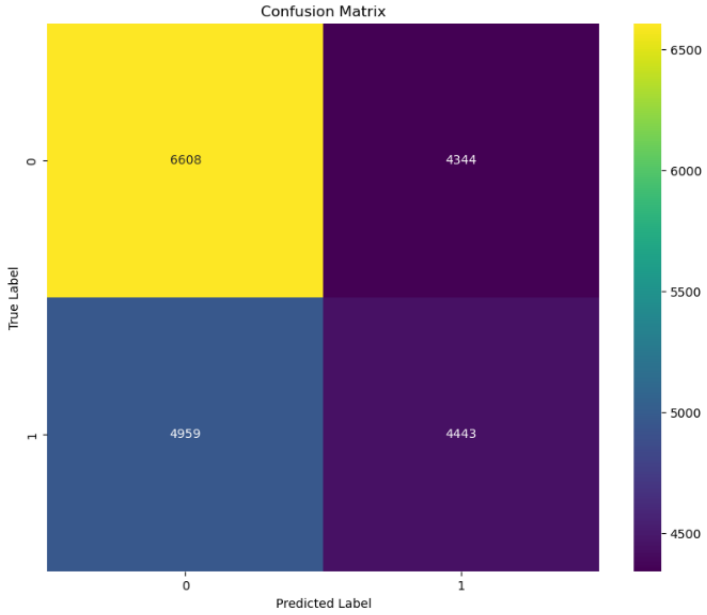


- Best parameters: 'bootstrap_type': 'MVS', 'iterations': 500, 'learning_rate': 0.1
- Accuracy: 0.65, Macro avg F1-score: 0.64

6.2 PCA

6.2.1 KNN

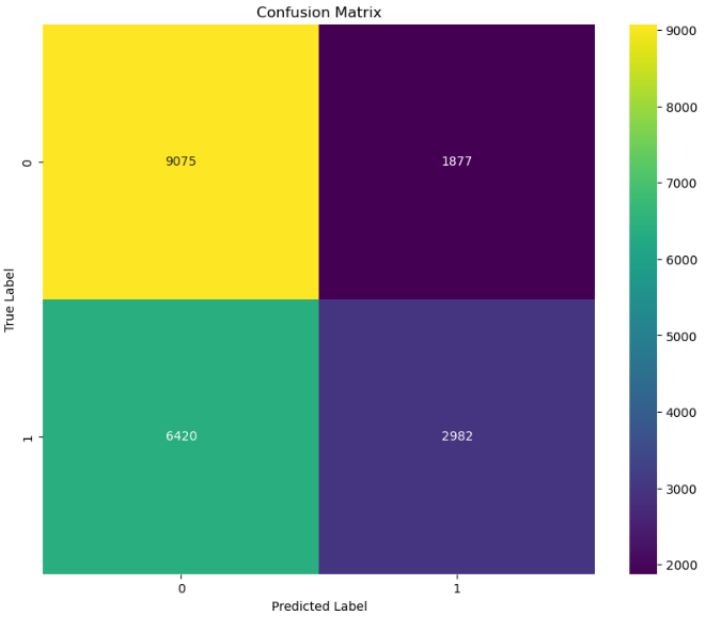
- Confusion matrix



- Best parameters: 'n_neighbors': 7, 'weights': 'distance'
- Accuracy: 0.54, Macro avg F1-score: 0.54

6.2.2 Gaussian NB

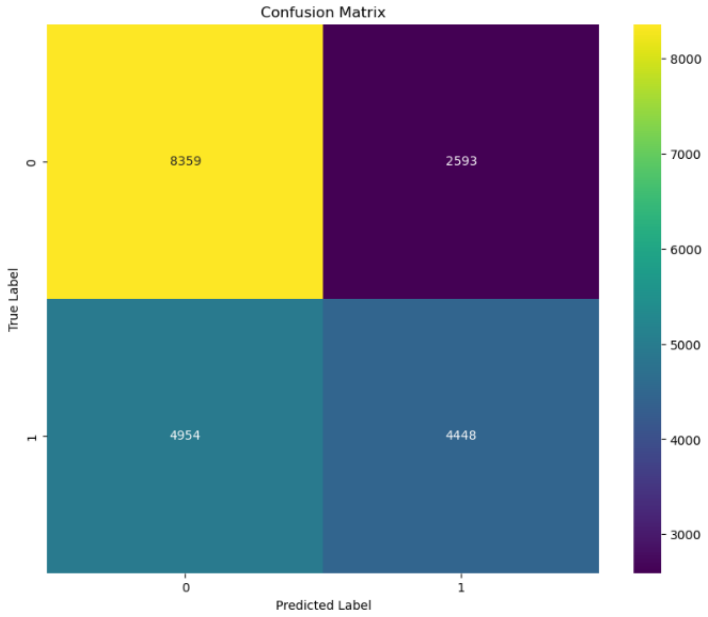
- Confusion matrix



- Can't perform grid search on it
- Accuracy: 0.59, Macro avg F1-score: 0.55

6.2.3 Random Forest

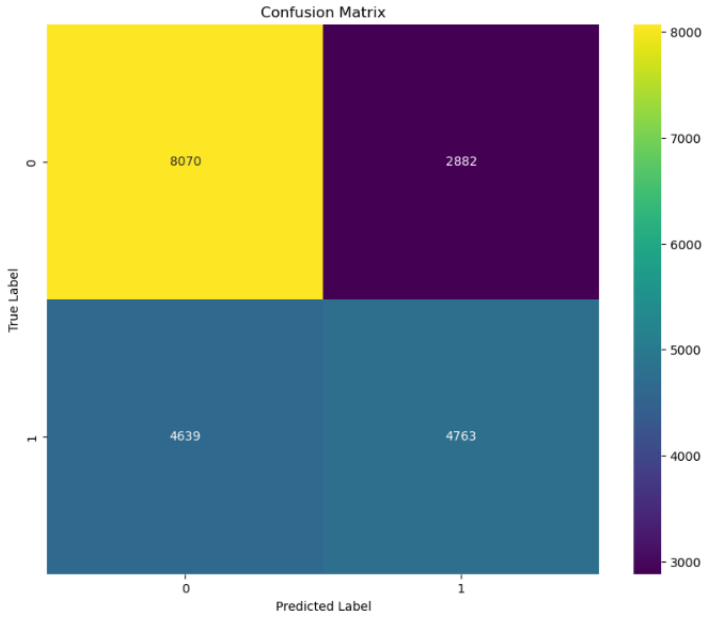
- Confusion matrix



- Best parameters: 'max_depth': 15, 'min_samples_split': 5, 'n_estimators': 300
- Accuracy: 0.63, Macro avg F1-score: 0.61

6.2.4 XGB Classifier

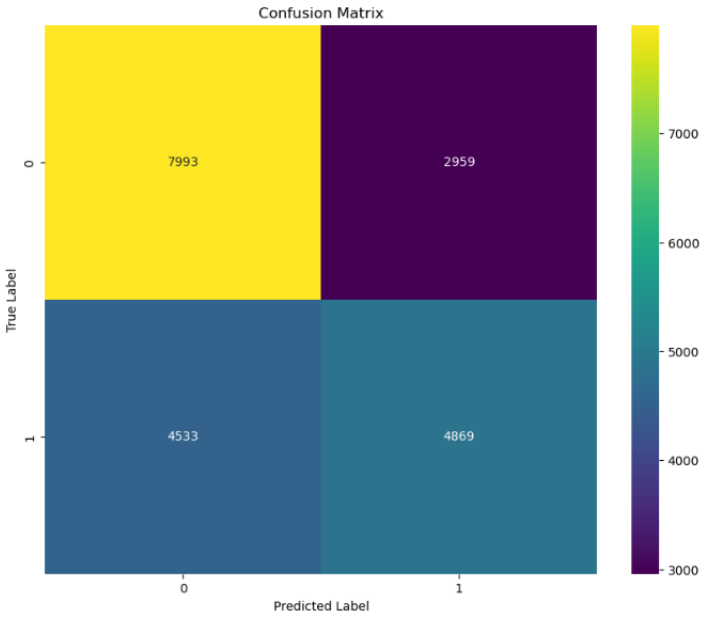
- Confusion matrix



- Best parameters: 'learning_rate': 0.1, 'n_estimators': 300
- Accuracy: 0.63, Macro avg F1-score: 0.62

6.2.5 CatBoostClassifier

- Confusion matrix

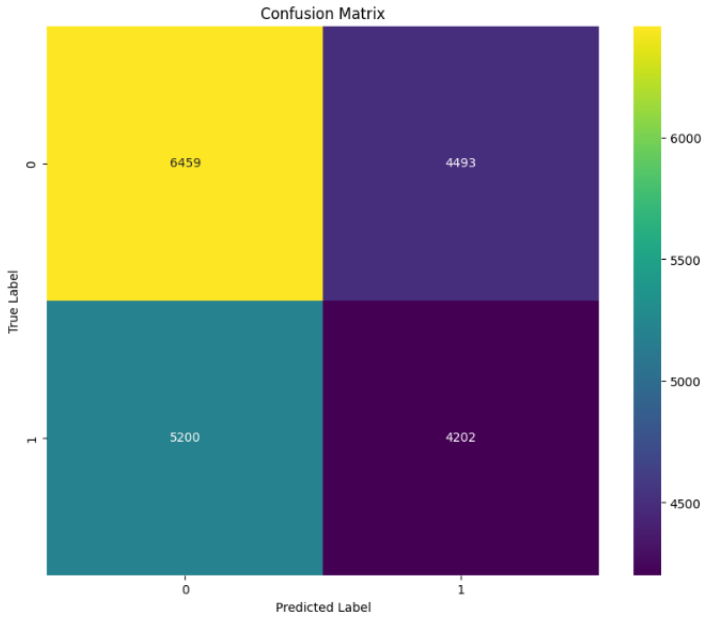


- Best parameters: 'boosting_type': 'Plain', 'bootstrap_type': 'MVS', 'iterations': 500, 'learning_rate': 0.1
- Accuracy: 0.63, Macro avg F1-score: 0.62

6.3 Random Projection

6.3.1 KNN

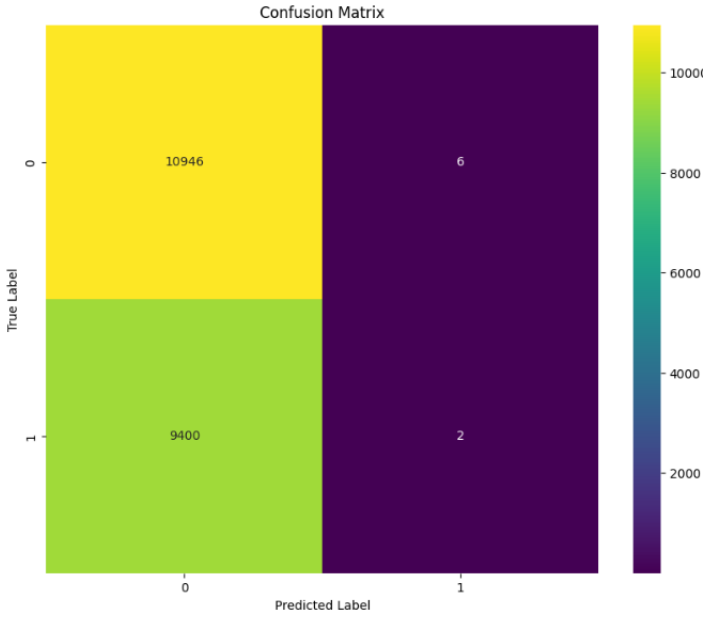
- Confusion matrix



- Best parameters: 'n_neighbors': 7, 'weights': 'uniform'
- Accuracy: 0.52, Macro avg F1-score: 0.52

6.3.2 Gaussian NB

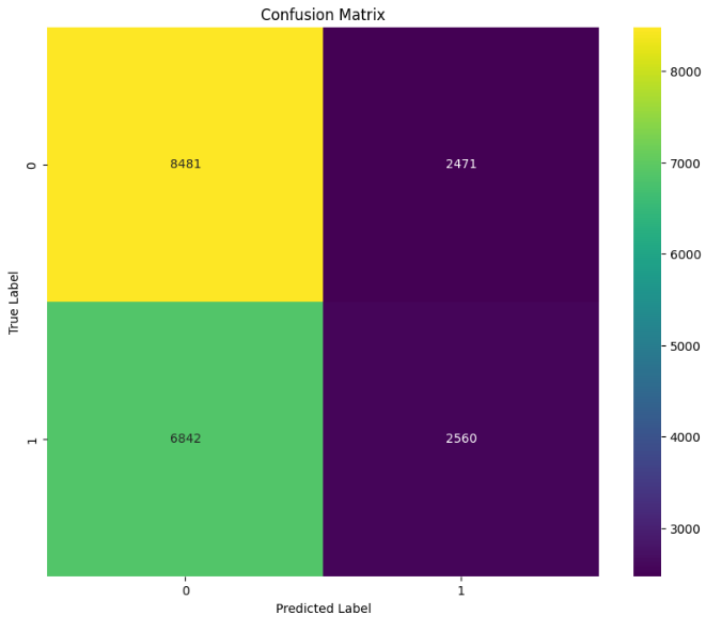
- Confusion matrix



- Can't perform grid search on it
- Accuracy: 0.54, Macro avg F1-score: 0.35

6.3.3 Random Forest

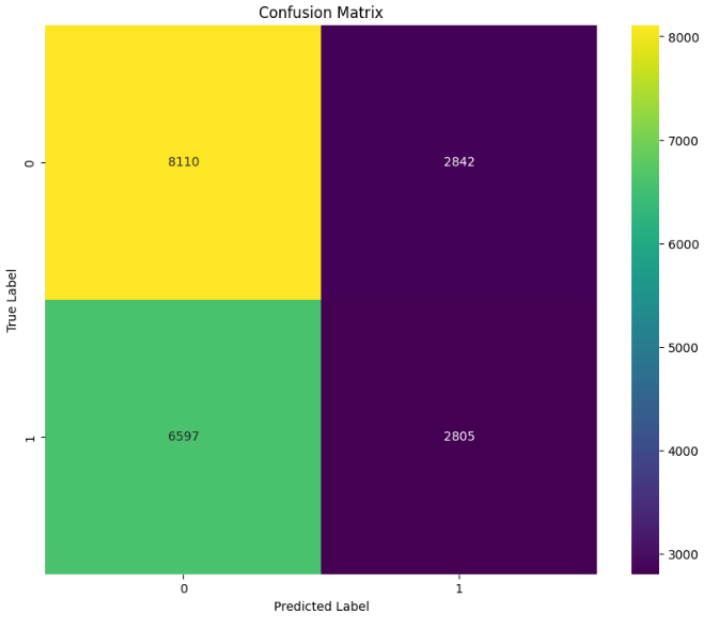
- Confusion matrix



- Best parameters: 'max_depth': 15, 'min_samples_split': 2, 'n_estimators': 300
- Accuracy: 0.54, Macro avg F1-score: 0.50

6.3.4 XGB Classifier

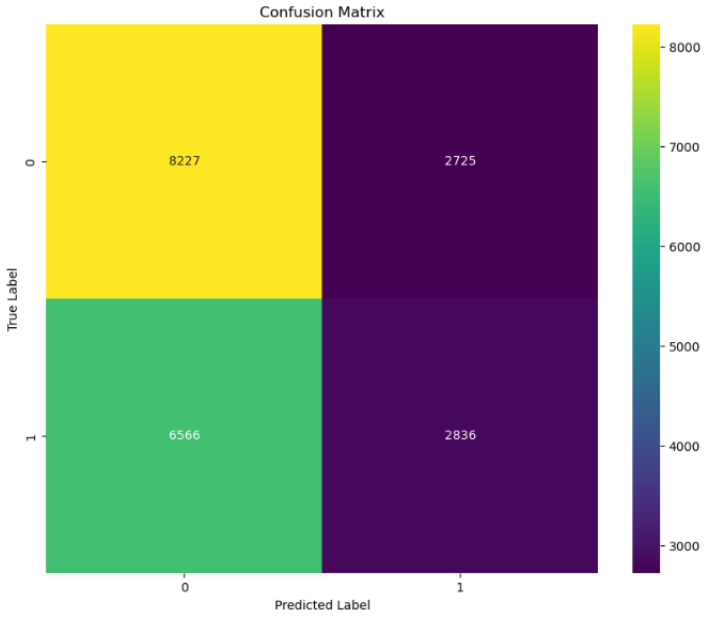
- Confusion matrix



- Best parameters: 'learning_rate': 0.2, 'n_estimators': 300
- Accuracy: 0.54, Macro avg F1-score: 0.50

6.3.5 CatBoostClassifier

- Confusion matrix

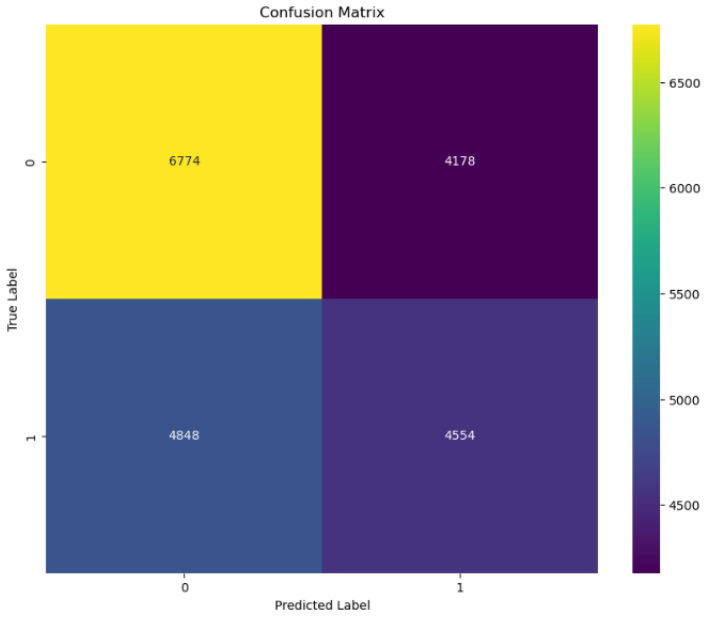


- Best parameters: 'boosting_type': 'Plain', 'bootstrap_type': 'MVS', 'iterations': 1000, 'learning_rate': 0.1
- Accuracy: 0.54, Macro avg F1-score: 0.51

6.4 Uniform Manifold Approximation and Projection

6.4.1 KNN

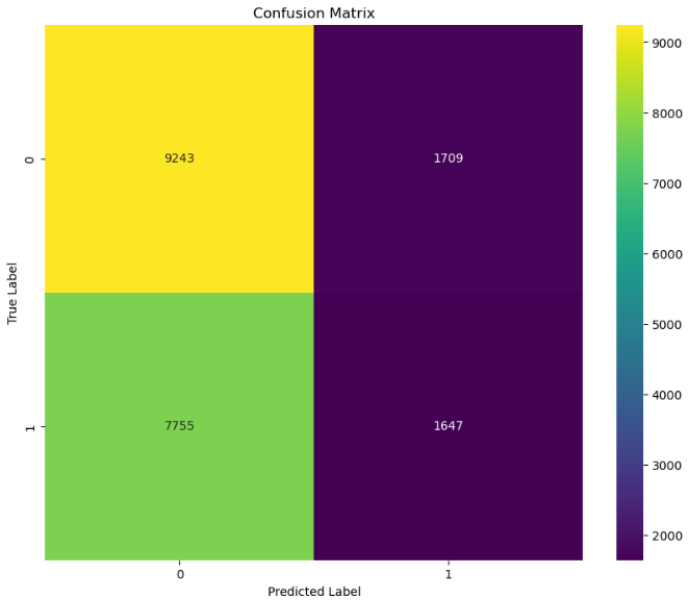
- Confusion matrix



- Best parameters: 'n_neighbors': 7, 'weights': 'uniform'
- Accuracy: 0.56, Macro avg F1-score: 0.55

6.4.2 Gaussian NB

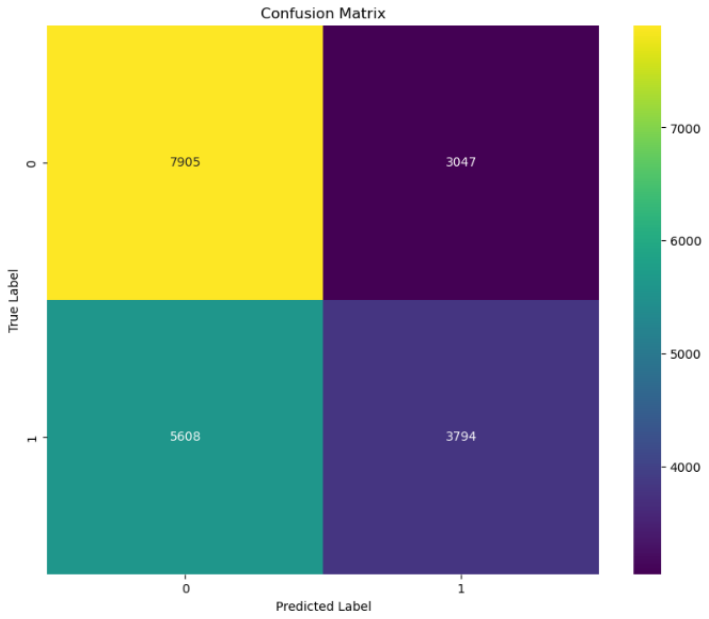
- Confusion matrix



- Can't perform grid search on it
- Accuracy: 0.54, Macro avg F1-score: 0.46

6.4.3 Random Forest

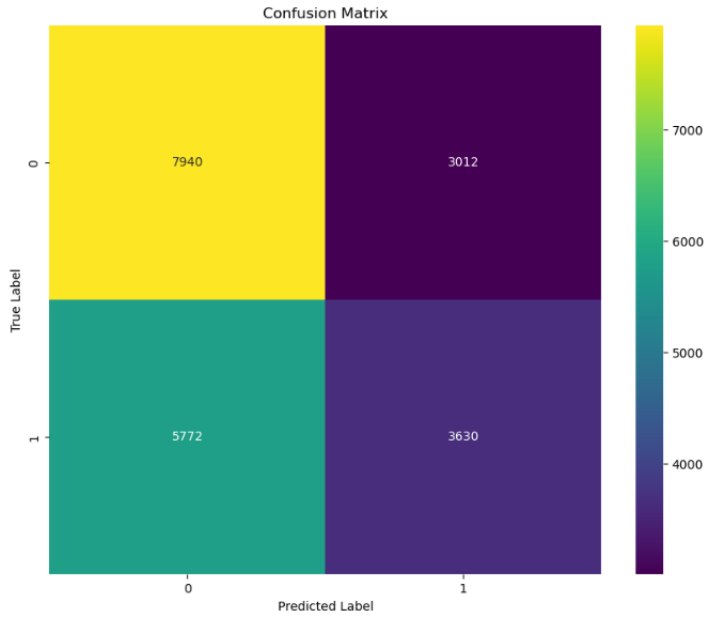
- Confusion matrix



- Best parameters: 'max_depth': 15, 'n_estimators': 200
- Accuracy: 0.57, Macro avg F1-score: 0.56

6.4.4 XGB Classifier

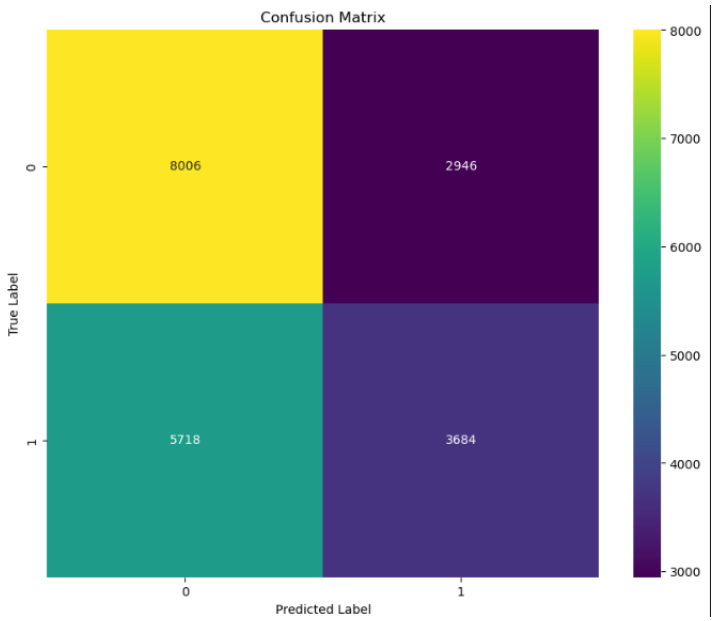
- Confusion matrix



- Best parameters: 'learning_rate': 0.2, 'n_estimators': 200
- Accuracy: 0.57, Macro avg F1-score: 0.55

6.4.5 CatBoostClassifier

- Confusion matrix



- Best parameters: 'bootstrap_type': 'MVS', 'iterations': 1000, 'learning_rate': 0.1
- Accuracy: 0.57, Macro avg F1-score: 0.55

7 Comparison between approaches

In our code, we automated the process to compute all the evaluation scores in some specific tables for an easier comparison between the models and the dimension reduction techniques. We will present our results.

Reduction Method	RandomForest	GaussianNB	KNN	XGBClassifier	CatBoostClassifier
simple_encoding	0.6406	0.5890	0.5831	0.6486	0.6486
PCA	0.6292	0.5924	0.5429	0.6305	0.6319
random_projection	0.5424	0.5379	0.5238	0.5363	0.5435
UMAP	0.5748	0.5350	0.5565	0.5684	0.5743

Table 1: Accuracy of Different Models Across Dimension Reduction Methods

Reduction Method	RandomForest	GaussianNB	KNN	XGBClassifier	CatBoostClassifier
simple_encoding	0.6283	0.5197	0.5760	0.6421	0.6419
PCA	0.6150	0.5522	0.5377	0.6205	0.6230
random_projection	0.5001	0.3499	0.5179	0.5025	0.5091
UMAP	0.5567	0.4598	0.5512	0.5482	0.5542

Table 2: F1-scores of Different Models Across Dimension Reduction Methods

8 Conclusions

- Different models show different degrees of sensitivity to dimension reduction techniques. While some models, like RandomForestClassifier and CatBoostClassifier, maintain good performance across different preprocessing methods, others, like GaussianNB and KNeighborsClassifier, may experience more significant fluctuations in performance.
- PCA offers reasonably good performance across both accuracy and F1-score metrics, suggesting its effectiveness in capturing essential information during dimensionality reduction while preserving classification performance.
- Random projection and UMAP have lower performance compared to other methods, indicating challenges in keeping crucial information during dimensionality reduction, which leads to decreased classification accuracy and F1-scores.
- GaussianNB lags behind other models in terms of both accuracy and F1-score, indicating limitations in handling complex datasets and balancing precision and recall effectively.

Another important aspect to note is the impressive performance of the PCA dimension reduction method compared to simple encoding. While we lost almost 2% in accuracy and macro-average F1-score evaluation metrics, we gained significantly in terms of dimensionality and computational resources, as PCA considers less than half of the features. More details on all three dimension reduction methods are analyzed below:

- Simple encoding used, in total, 34 features (out of the initial 47, though some were later dropped during preprocessing).
- PCA considered only 12 features, which is just 35% compared to the initial set of features.
- Random projection, although it showed lower overall performance, computed only 3 features (just 8% of the preprocessed data).
- UMAP, last but not least, also demonstrated good performance while retaining only 3 features.

We should also note that, without any cleaning of the data, the initial results for the best models combined with the best dimension reduction method reached an accuracy of only 20%, compared to the 65% achieved in the final computations.

The varying performance of models across different dimension reduction techniques highlights the importance of considering the specific characteristics of the dataset, such as its complexity, dimensionality, and feature distributions, when selecting the most suitable preprocessing method and model. Therefore, we consider that by using medical knowledge, we can make our data preprocessing better. This might help us find more important patterns in the data, leading to better results in classifying it.