DATA1030 Final Project

Data Science Initiative
BROWN

Predicting Chronic Kidney Disease

Prepared By:

Bowei Sun

A Report Prepared For:

Data Science Institute at Brown University

Link to Github:

https://github.com/EricccS/FinalProject1030.git

Introduction

Chronic kidney disease (CKD) is a critical health condition that requires timely diagnosis and risk management to prevent severe complications. It is important to explore some prediction techniques to classify patients as having CKD or not, so as to enable early intervention, effective risk management, and efficient resource allocation. This project aims to employ and compare the performance diverse machine learning (ML) models to enhance CKD detection, support clinical decision-making, and optimize healthcare resource utilization. The dataset, sourced from the UCI repository, contains patient information collected over two months in a hospital setting. And it has 400 data points and 26 columns. One of the columns is for the patient's id , and all the IDs are independent and identically distributed. So the dataset is a small, iid dataset. Someone on Kaggle solved this prediction problem and after the similar EDA and preprocessing process, he tried some deep learning techniques: Artificial Neural Network (ANN) with accuracy score 0.95, long short-term memory (LSTM) with accuracy score 0.96, convolutional neural network (CNN) with accuracy score 0.99. Whereas in this report, though more machine learning models are trained and explored, the result of predictive power is almost the same as previous work.

Exploratory Data Analysis (EDA)

The target variable is binary and has two classifications: 250 patients are diagnosed with chronic kidney disease and 150 are not diagnosed with chronic kidney disease. Therefore, the dataset is a balanced dataset. And all the features can be classified into 11 continuous features, 10 categorical features and 3 ordinal features. All features contain some percent of missing values, ranging from 0.001 to 0.327, and the target variable does not contain any missing values.

After broadly reviewing all the histograms for continuous features, it is interesting to see that some features have distinct outliers and heavy tails on the patients who have CKD, and no outliers nor heavy tails happening on the patients who do not have CKD. The box plot to illustrate this is shown in Fig1. And some continuous features have clear mean differences in distribution by different classification, which is shown in Fig2. It is plausible that those features are more important in predicting CKD or not.

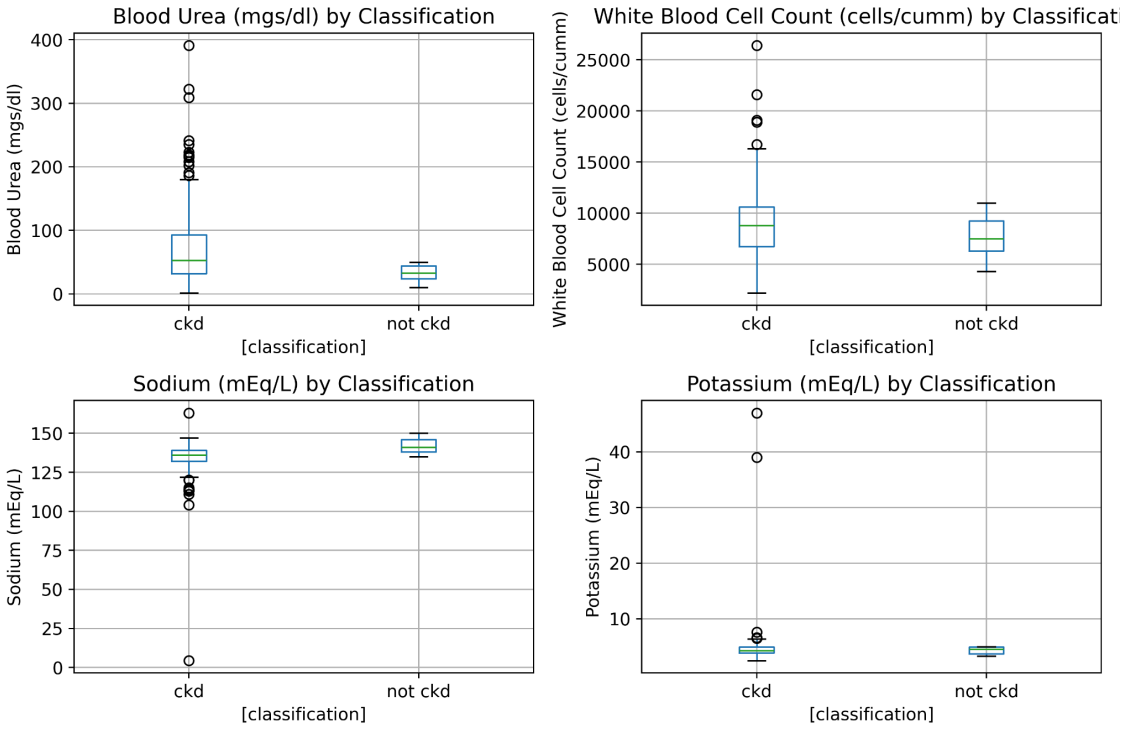

Fig1. Box plots of 4 Different Features Grouped by Classification

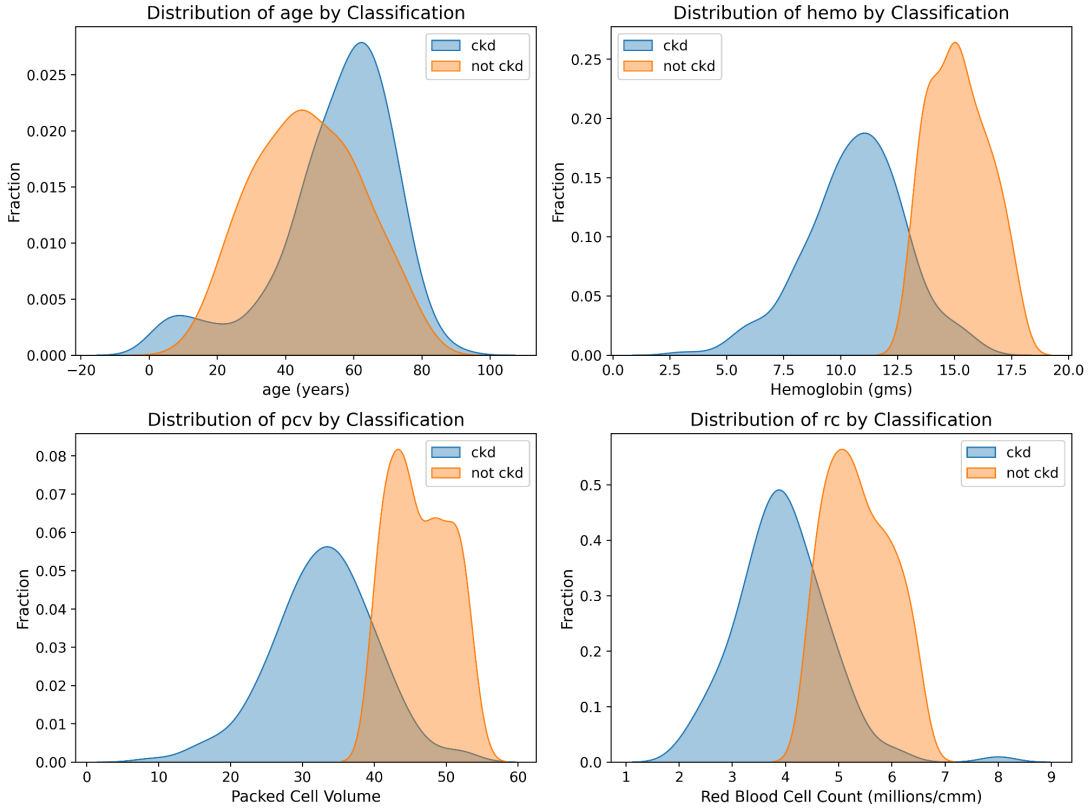

Fig2. Histograms of 4 Different Features Grouped by Classification

Then, to avoid multicollinear and information redundancy between features, after drawing the correlation matrix for all continuous variables as shown in Fig3, it can be found that there is strong correlation between Hemoglobin (hemo) and Packed Cell Volume (pcv). So the feature hemo is dropped for future analysis.

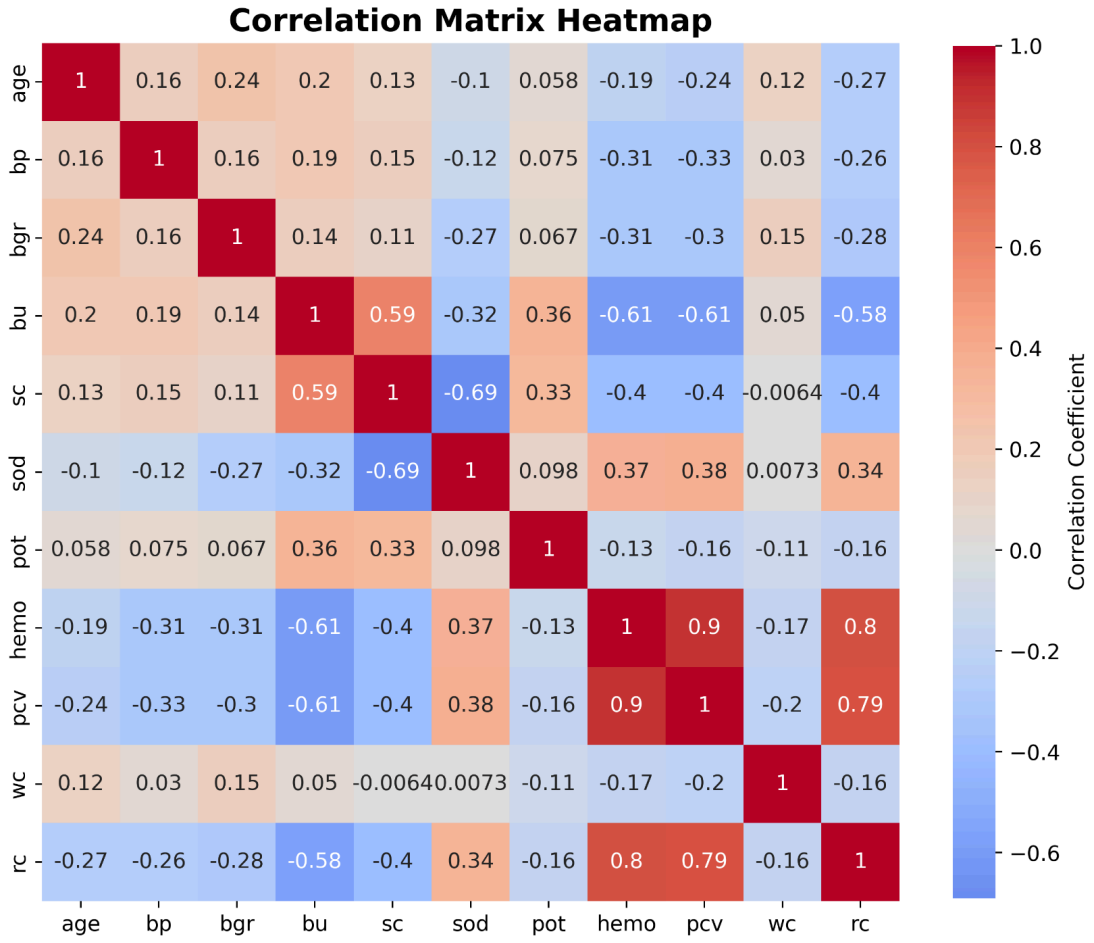

Fig3. Correlation Matrix Heatmap

Then for all the categorical features, the stacked bar plots are shown in Fig4. It is clear to see that all categorical features are binary and the not CKD situation only happens in one category of each categorical feature.

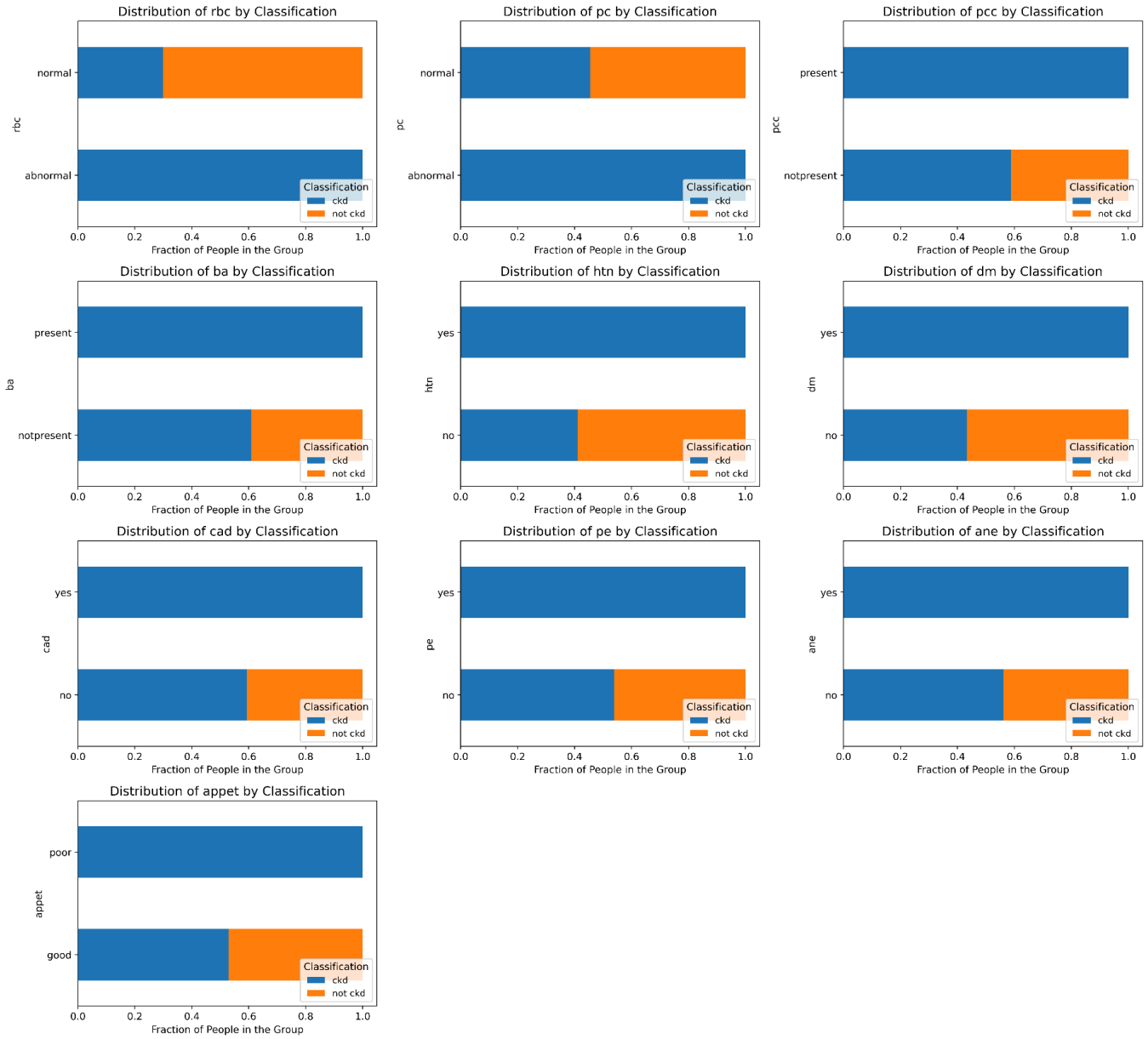

Fig4. Stacked Bar Plots for Categorical Features by Classification

Methods

Since the dataset is small, iid, and balanced, the choice of splitting the dataset is basic split. And the training set size is 0.6, validation and test sets size are 0.2. Also, as the features have been classified into 3 different types, OneHotEncoder, OrdinalEncoder are applied to categorical

features and ordinal features respectively with treating missing values as another category, and StandardScaler is employed on all continuous features in the pipeline. Then, at the last step of preprocessing, StandardScaler is applied again on all the features to ensure they are on the same scale, with a mean of 0 and standard deviation of 1. This prevents features with larger magnitudes from dominating the model and improves the performance of algorithms sensitive to feature scaling. Note that fit_transform is used for the training set and transform is used for the validation and test sets. After the preprocessing, all the missing values in categorical and ordinal features are imputed but still left many missing values on continuous features. Then, the ML algorithm is implemented and the whole pipeline is shown below:

- Initialize empty variables for storing values
- For each different random state:
    - Split the dataset into training, validation and test sets
    - Preprocess the training, validation and test sets
    - Impute missing values for continuous features
    - Define hyperparameter grid for tuning
    - For each hyperparameter combination:
        - Initialize ML model with certain combination of hyperparameter
        - Fit the model into training set
        - Evaluate model using validation set
        - Store the model and its validation score
    - Select the best model by optimizing validation score
    - Use the best model with its hyperparameter combination to get the test score

And there are a few things worthy to note about this pipeline:

1. Since the data size is relatively small and computationally fast, to get a more generalized result , the number of different random states was chosen to be 10.
2. As the dataset has many features with missing values, choosing reduced-features model technique to impute would be very time consuming. Hence, the multivariate Imputation (IterativeImputer) was chosen to impute missing values in continuous features, and the estimator is RandomForestRegressor since all the features have been encoded as numeric and standardized.
3. XGBoosting is exempted from the step of imputation of missing values.

In total, 5 different ML models were trained and tested, and the summary is shown below:

ML Algorithm	Sklearn Package	Parameter Tuning
XGBoosting	XGBClassifier(n_estimators = 10000, early_stopping_rounds=30)	'max_depth': [1, 3, 5, 8], 'reg_alpha': [1e-4, 1e-3, 1e-2, 1e-1, 0e0], 'reg_lambda': [1e-4, 1e-3, 1e-2, 1e-1, 0e0]
Random Forest	RandomForestClassifier()	'max_depth': [1, 3, 5, 8, 13], 'max_features': [1, 3, 5, 8, 13, 18]
SVM	SVC()	'C': [1e-1, 1e0, 1e1, 1e2, 1e3, 1e4], 'gamma': [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1]
Logistic Regression	LogisticRegression('penalty' = 'elasticnet')	'l1_ratio': np.linspace (0, 1, 8), 'C': [0.01, 0.05, 0.1, 0.5, 1, 1.5, 2]
KNN	KNeighborsClassifier()	'n_neighbors': [1, 3, 5, 10], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan', 'minkowski']

Table1. Summary of 5 Different ML Models

Also, there are a few things worthy to note:

1. n_estimators is a very large number in XGBoosting and early_stopping_rounds is used for automatically determining the best number of trees based on a validation set.
2. Logistic regression uses elastic net regularization, where the l1_ratio parameter needs to be tuned to determine the proportion of the Lasso component.
3. Some ML techniques offer separate packages for regression problems. Since this project focuses on classification, it is important to carefully choose the correct package names.

Results

For this classification project with a balanced dataset, the baseline accuracy is calculated by predicting a single category for all test data points, resulting in an accuracy score of 0.63125. Figure 5 displays the mean and standard deviation of test scores across five different ML algorithms, all of which achieve mean accuracy scores above 0.95, significantly outperforming the baseline. The small standard deviations indicate consistent test scores across different random

states, further confirming the models' strong predictive performance. To compare the ML algorithms more precisely, the Table2 proves the most predictive ML algorithm is XGBoosting since it has the highest mean test score 0.995.

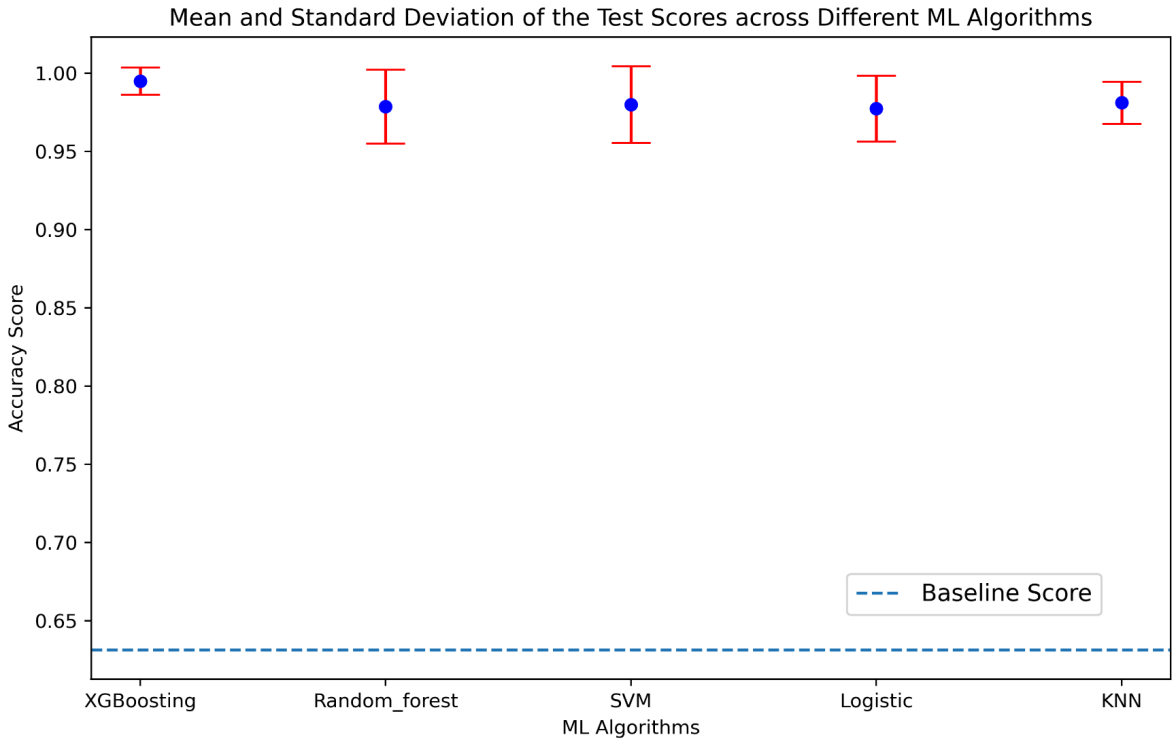

Fig5. Mean and Standard Deviation of the Test Scores across Different ML Algorithms

	XGBoosting	Random Forest	SVM	Logistic Regression	KNN
Mean Test Score	0.99500	0.97875	0.98000	0.97750	0.98125

Table2. Mean Test Scores of 5 Different ML Models

The best model in XGBoost across different random states consistently has the same optimized tuning parameters: 'reg_lambda' = 0.0001, 'reg_alpha' = 0.0001, and 'max_depth' = 1. These parameters ensure simplicity and prevent overfitting. The shallow trees focus on clear patterns, while minimal regularization reflects low noise and relevant features, enabling a well-generalized, efficient model.

To evaluate the model's performance, a confusion matrix for the XGBoost model was generated, as shown in Fig6. The matrix highlights that only a very small percentage of predictions fall into the False Positive (FP) and False Negative (FN) categories, demonstrating the model's strong predictive accuracy. Further performance metrics calculated from the 2x2 confusion matrix show a recall score of 0.992, a precision score of 1.0, and an F1 score of 0.996. These results indicate that the XGBoost model performs exceptionally well, likely requiring no further adjustments.

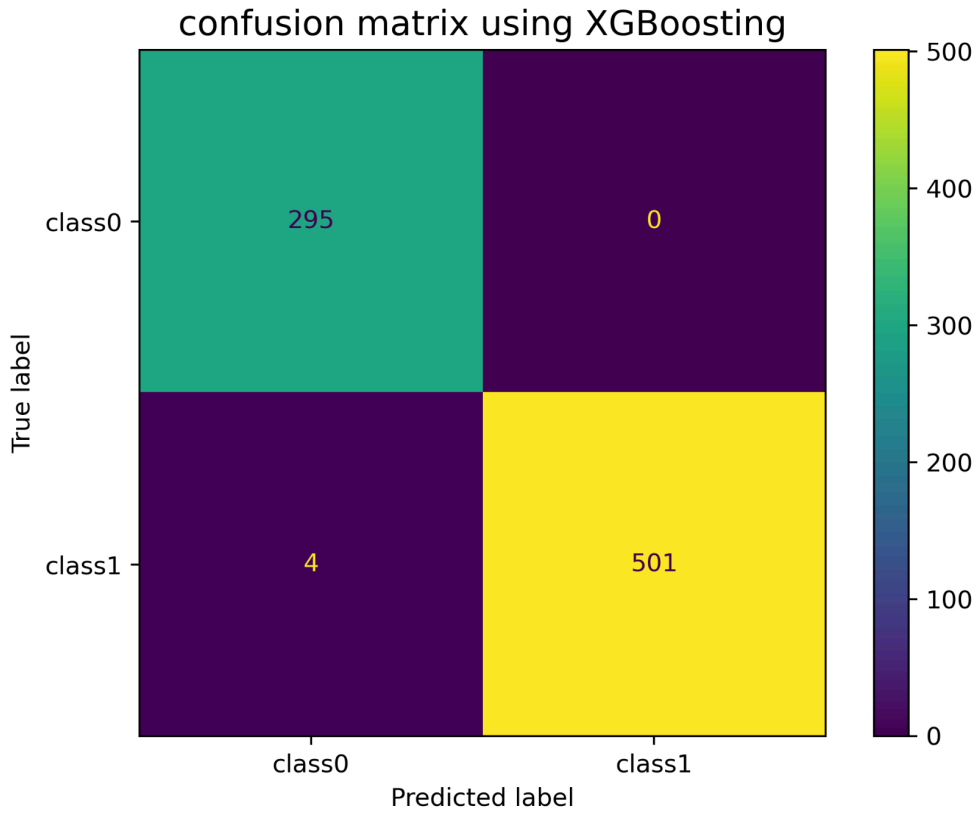

Fig6. Confusion Matrix Using XGBoosting

To investigate the global feature importance, 5 different methods are used as shown in Fig7 and Fig8 below. Perturbation, SHAP values with global feature, embedded “importance_type” metric in XGBoosting with 3 different types of measure: gain, cover and weight. And some features that are important across all the methods or plots: rc (red blood cell counts), al (albumin), sg (specific gravity), sc (serum creatinine) and etc. And there is no one or few features that dominate the importance, almost all the features have some predicted contribution to the target variable, which is also a proof of why the model can predict so well.

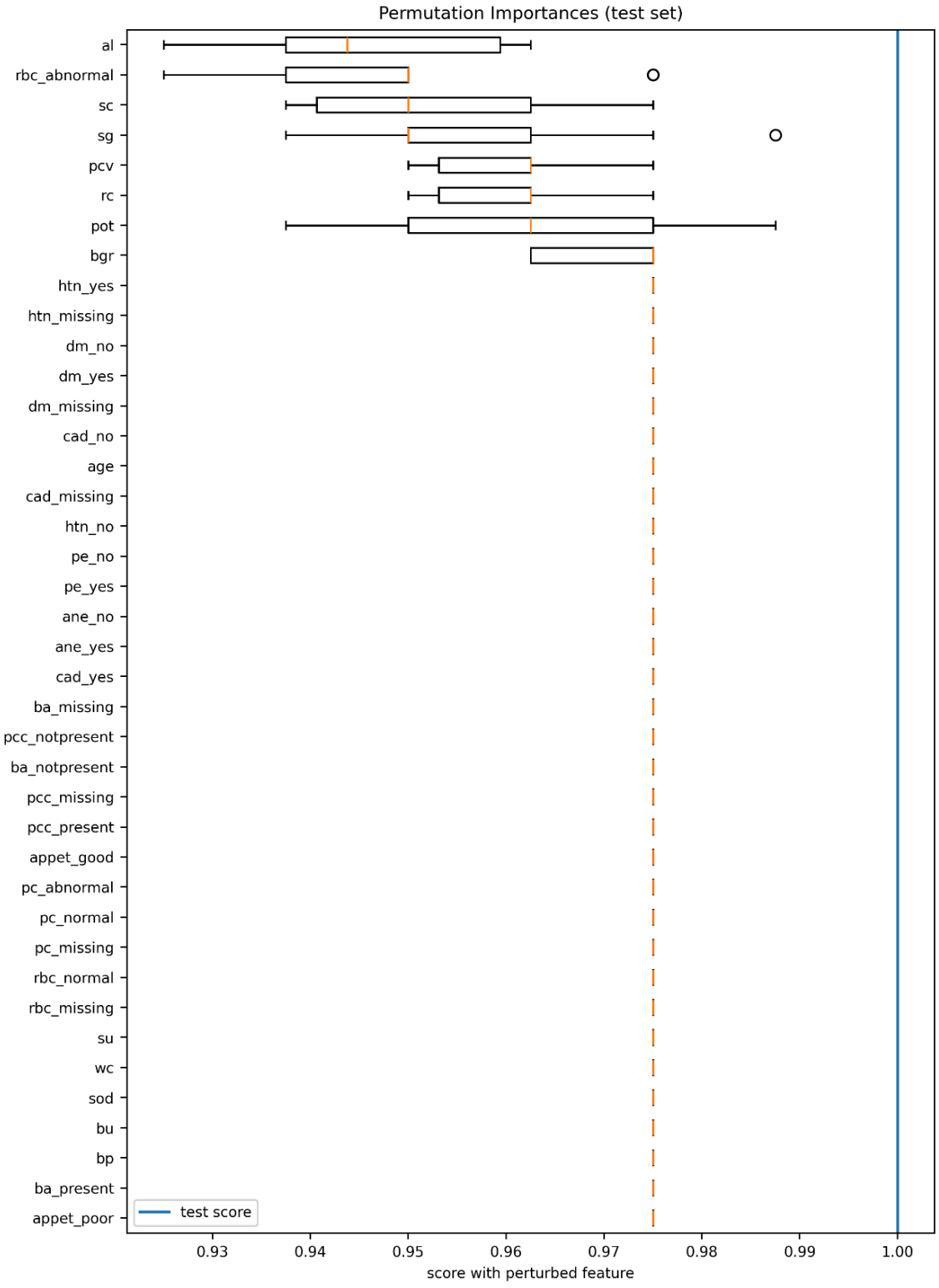

Fig7. Iterative Permutation for Global Feature Importance

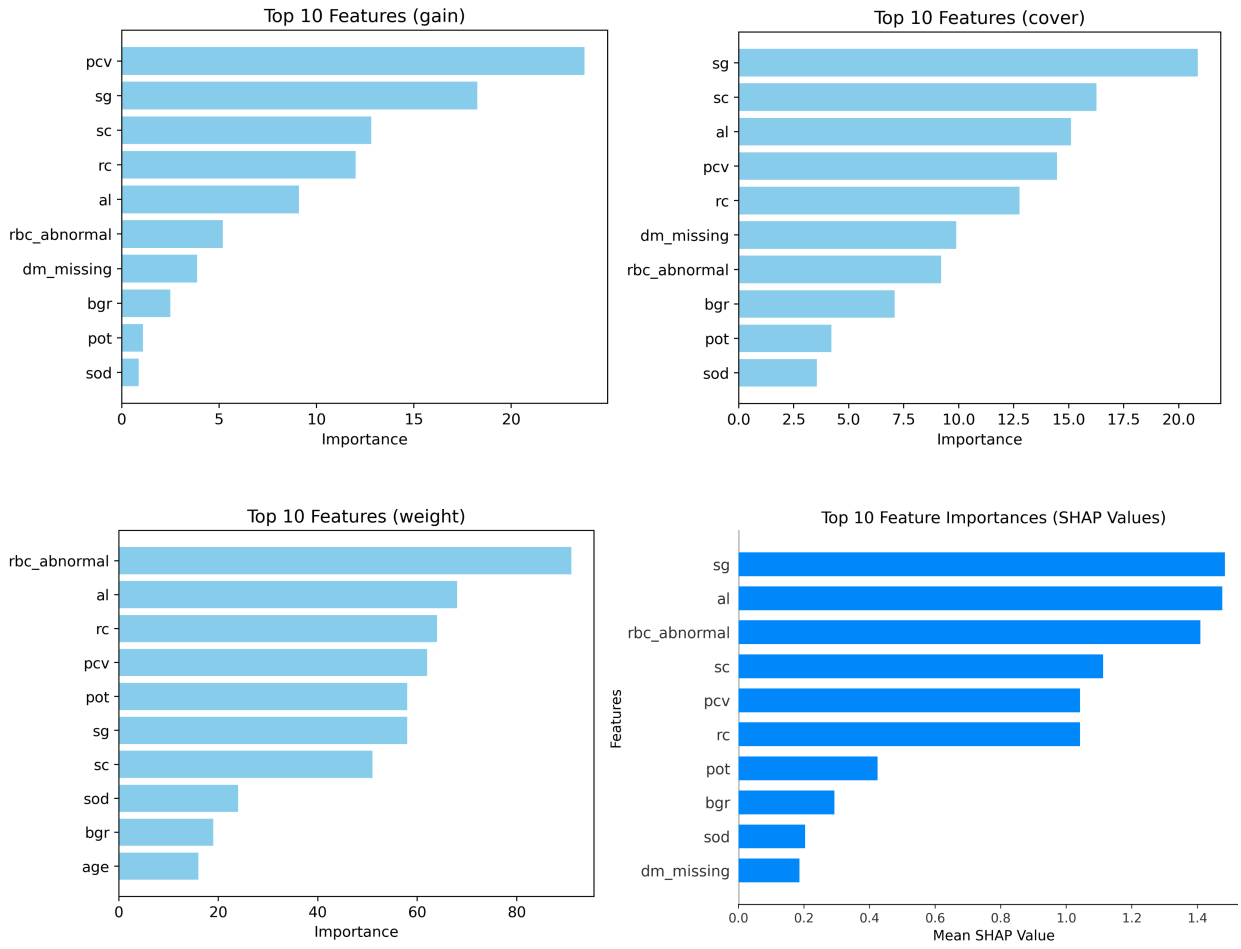

Fig8. Four More Different Plots for Global Feature Importance

To examine local feature importance, row index 68 was selected for a prediction of class 0. As shown in the force plot in Fig. 9, the base value is 0.629. Features such as rbc_normal and pcv contributed to lowering the prediction, while features like al and sc had negative SHAP values, suggesting that these features strongly influenced the model toward predicting class 0. Conversely, row index 17 was selected for a prediction of class 1, as illustrated in Fig. 10. In this case, features like al and sg pushed the prediction upward, indicating that these features contributed significantly to moving the prediction from the base value toward class 1.

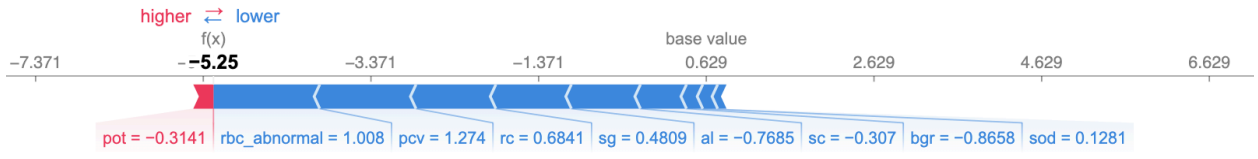

Fig9. Local Feature Importance for Index 68

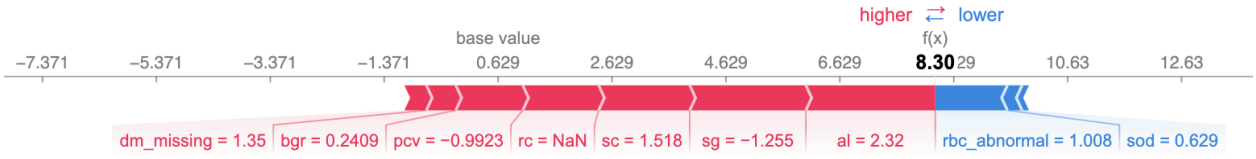

Fig10. Local Feature Importance for Index 17

Outlooks

To improve the model's predictive power, we can implement a K-Fold cross-validation strategy, where the dataset is split into n subsets (folds) for each parameter combination. The model is trained and validated on different subsets in each fold, which helps in assessing its performance more reliably and ensures better generalizability. This reduces the risk of overfitting to specific data points, providing a more robust estimate of the model's ability to perform on unseen data.

For model interpretability, performing a confusion matrix analysis across different groups of patient IDs will allow us to investigate how well the model predicts for various subpopulations. This analysis can help identify potential biases, such as the model performing better or worse for certain groups based on demographics or health conditions. By understanding these biases, targeted adjustments can be made, such as modifying the training data or refining model parameters to improve predictions for underrepresented groups.

Combining this with SHAP can provide deeper insights into which features are driving the predictions, and whether their influence differs across patient groups. This approach will help not only improve the accuracy of the model but also ensure that it is fair and interpretable, which is crucial for real-world healthcare applications.

References

Data source: https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease

Previous work:

https://www.kaggle.com/code/yasiralibhutto/chronic-kidney-disease-prediction-99-accuracy/notebook#Display-accuracy-scores%F0%9F%96%A5%EF%B8%8F