Predictive Patterns: Data Modeling Insights into Chronic Kidney Disease Progression

Data Science II

COSC 4337

Submitted to

Dr. Ricardo Vilalta

Submitted by

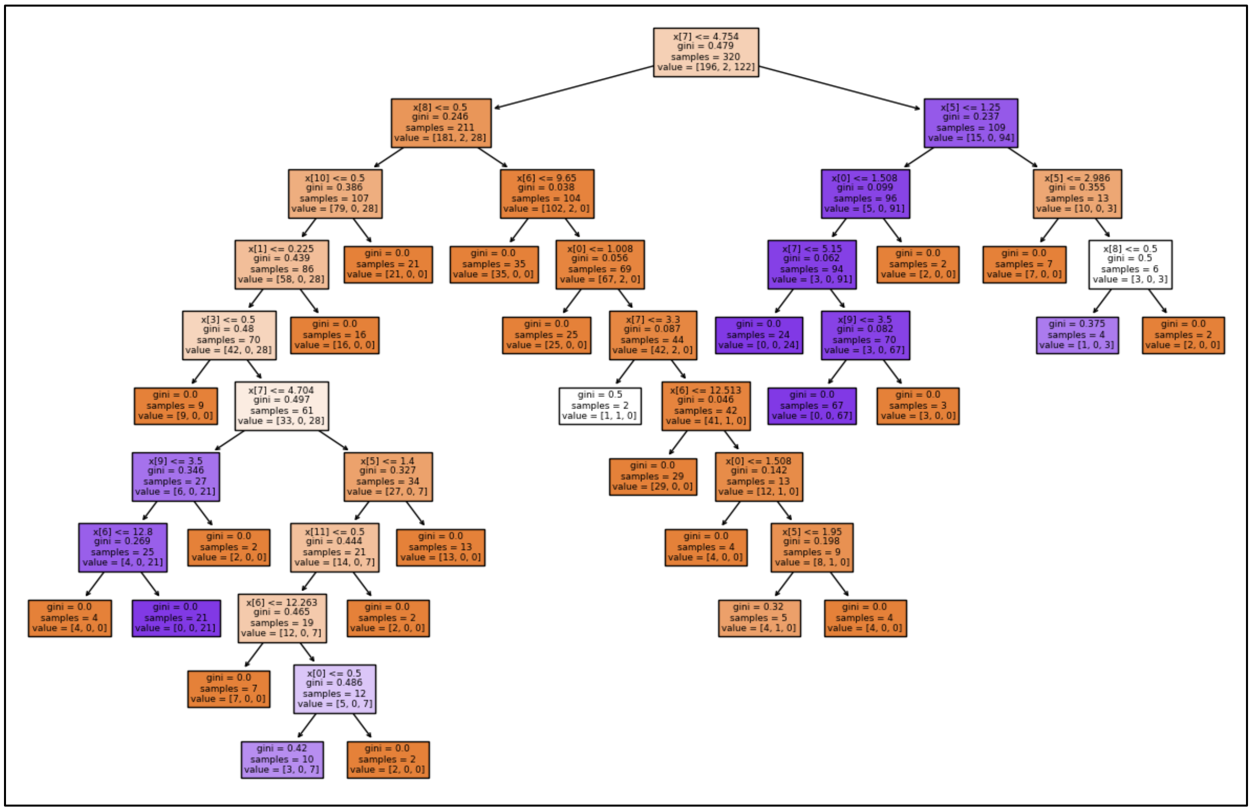
Joseph Irving (1766731)

Daniel Emami (1390157)

Ryan Nguyen (1897135)

Decision Tree Classifier

In the realm of machine learning, decision trees are a robust tool employed for both classification and regression tasks. They offer the unique advantage of interpretability, as the decision-making process can be visualized and comprehended. This study harnesses a decision tree model to analyze a dataset related to kidney disease. The dataset comprises several features including patient age, blood pressure, specific gravity, albumin, sugar, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, white blood cell count, and red blood cell count. The decision tree model was trained using the scikit-learn library in Python. The hyperparameters of the model were tuned via GridSearchCV, an exhaustive search method that iterates over a predefined grid of hyperparameters. This method selects the combination of hyperparameters that yields the highest performance measured by a specified scoring metric. In this case, that would be accuracy.


x[7] <= 4.754 gini = 0.479 samples = 320 value = [196, 2, 122]
x[8] <= 0.5 gini = 0.246 samples = 211 value = [181, 2, 28]
x[5] <= 1.25 gini = 0.237 samples = 109 value = [15, 0, 94]
x[10] <= 0.5 gini = 0.386 samples = 107 value = [79, 0, 28]
x[6] <= 9.65 gini = 0.038 samples = 104 value = [102, 2, 0]
x[0] <= 1.508 gini = 0.099 samples = 96 value = [5, 0, 91]
x[5] <= 2.986 gini = 0.355 samples = 13 value = [10, 0, 3]
x[1] <= 0.225 gini = 0.439 samples = 86 value = [58, 0, 28]
gini = 0.0 samples = 21 value = [21, 0, 0]
gini = 0.0 samples = 35 value = [35, 0, 0]
x[0] <= 1.008 gini = 0.056 samples = 69 value = [67, 2, 0]
x[7] <= 5.15 gini = 0.062 samples = 94 value = [3, 0, 91]
gini = 0.0 samples = 2 value = [2, 0, 0]
gini = 0.0 samples = 7 value = [7, 0, 0]
x[8] <= 0.5 gini = 0.5 samples = 6 value = [3, 0, 3]
x[3] <= 0.5 gini = 0.48 samples = 70 value = [42, 0, 28]
gini = 0.0 samples = 16 value = [16, 0, 0]
gini = 0.0 samples = 25 value = [25, 0, 0]
x[7] <= 3.3 gini = 0.087 samples = 44 value = [42, 2, 0]
gini = 0.0 samples = 24 value = [0, 0, 24]
x[9] <= 3.5 gini = 0.082 samples = 70 value = [3, 0, 67]
gini = 0.375 samples = 4 value = [1, 0, 3]
gini = 0.0 samples = 2 value = [2, 0, 0]
gini = 0.0 samples = 9 value = [9, 0, 0]
x[7] <= 4.704 gini = 0.497 samples = 61 value = [33, 0, 28]
gini = 0.5 samples = 2 value = [1, 1, 0]
x[6] <= 12.513 gini = 0.046 samples = 42 value = [41, 1, 0]
gini = 0.0 samples = 67 value = [0, 0, 67]
gini = 0.0 samples = 3 value = [3, 0, 0]
x[9] <= 3.5 gini = 0.346 samples = 27 value = [6, 0, 21]
x[5] <= 1.4 gini = 0.327 samples = 34 value = [27, 0, 7]
gini = 0.0 samples = 29 value = [29, 0, 0]
x[0] <= 1.508 gini = 0.142 samples = 13 value = [12, 1, 0]
x[6] <= 12.8 gini = 0.269 samples = 25 value = [4, 0, 21]
gini = 0.0 samples = 2 value = [2, 0, 0]
x[11] <= 0.5 gini = 0.444 samples = 21 value = [14, 0, 7]
gini = 0.0 samples = 13 value = [13, 0, 0]
gini = 0.0 samples = 4 value = [4, 0, 0]
x[5] <= 1.95 gini = 0.198 samples = 9 value = [8, 1, 0]
gini = 0.0 samples = 4 value = [4, 0, 0]
gini = 0.0 samples = 21 value = [0, 0, 21]
x[6] <= 12.263 gini = 0.465 samples = 19 value = [12, 0, 7]
gini = 0.0 samples = 2 value = [2, 0, 0]
gini = 0.32 samples = 5 value = [4, 1, 0]
gini = 0.0 samples = 4 value = [4, 0, 0]
gini = 0.0 samples = 7 value = [7, 0, 0]
x[0] <= 0.5 gini = 0.486 samples = 12 value = [5, 0, 7]
gini = 0.42 samples = 10 value = [3, 0, 7]
gini = 0.0 samples = 2 value = [2, 0, 0]

Our findings reveal that the features represented by x[7] and x[8] play a significant role in the model's decision-making process. These features appear at the top of the decision tree, suggesting that they're key determinants in the classification of kidney disease. Given the mapping of our dataset, x[7] corresponds to *sc* (serum creatinine) and x[8] corresponds to *sod* (sodium). This indicates that these physical and biochemical parameters could be crucial indicators of kidney function and disease progression. Each node in the decision tree represents a decision rule based on a feature's value. For instance, the root node splits the data based on whether the serum creatinine level is less than or equal to 4.754. This suggests that patients with a serum creatinine level below this threshold have different outcomes than those with a higher level. The Gini impurity at each node, a measure of class purity, is also displayed. A lower Gini impurity indicates a purer node where a majority of samples belong to a single class. This provides insight into the model's decision-making process at each stage.

The depth of the tree, or the maximum length from the root to a leaf, provides insight into the model's complexity. In this case, the decision tree is relatively deep with a maximum depth of 10, which allows the model to capture more intricate patterns in the data. However, a deeper tree also runs the risk of overfitting. Overfitting occurs when the model learns the training data too well, capturing noise along with the underlying pattern, which can lead to poor performance on unseen data. This is where the concept of bias-variance tradeoff comes into play. A model with high bias (underfitting) oversimplifies the data and has high error on both training and test data. On the other hand, a model with high variance (overfitting) performs well on training data but poorly on unseen data. The goal is to find a balance between bias and variance, where the model performs well on both training and unseen data.

In the context of this decision tree analysis for kidney disease classification, the bias-variance tradeoff is particularly relevant. The tuning of hyperparameters to a max_depth of 10 suggests a model complexity that aims to capture detailed patterns in the data, while min_samples_leaf of 2 and min_samples_split of 5 prevent the model from creating overly specific rules based on the training data alone, which would increase variance. The high training accuracy of 0.95625 coupled with a perfect test score of 1.0 initially raises concerns about potential overfitting; however, the consistent performance across training and testing suggests that the model is not merely memorizing the training data but rather generalizing well to new data. This balance implies that an optimal bias-variance tradeoff has likely been achieved, with the model being complex enough to learn the underlying patterns (low bias) and general enough to apply these patterns to unseen data (controlled variance), though caution is advised to ensure the test data is truly representative to confidently validate the model's performance.

The hyperparameter tuning process identified the optimal parameters for the Decision Tree classifier as 'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5. The model achieved an accuracy score of 0.95625 on the training data and a perfect score of 1.0 on the test data, indicating robust performance. This suggests that the model is capable of accurately classifying kidney disease based on the given features.

Our analysis underscores the utility of decision trees in medical diagnosis and highlights the importance of features such as serum creatinine and sodium in the prediction of kidney disease. It also emphasizes the importance of considering the bias-variance tradeoff when building machine learning models. Future work could explore the application of more complex models or ensemble methods to further improve predictive performance. This could include techniques like boosting or bagging, or models like random forests or gradient boosting machines, which can capture more complex patterns and offer improved performance.

Random Forrest Classifier

The Random Forest Classifier is a popular machine learning algorithm know for it's simplicity and notable predictive capabilities that combines the output of multiple decision trees to reach a single result. It does this by creating multiple decision tree's and providing the output that occurs most often among these trees. A Random Forest has an advantage over standard decision trees in that it has the ability to mitigate overfitting while also being able to handle a large number of features. We chose this as one of our models as it is a supervised learning algorithm which is appropriate for our use case given we are working with a labeled dataset. Accuracy is a commonly used performance metric for evaluating classification tasks as it tells us the proportion of correctly predicted classes.

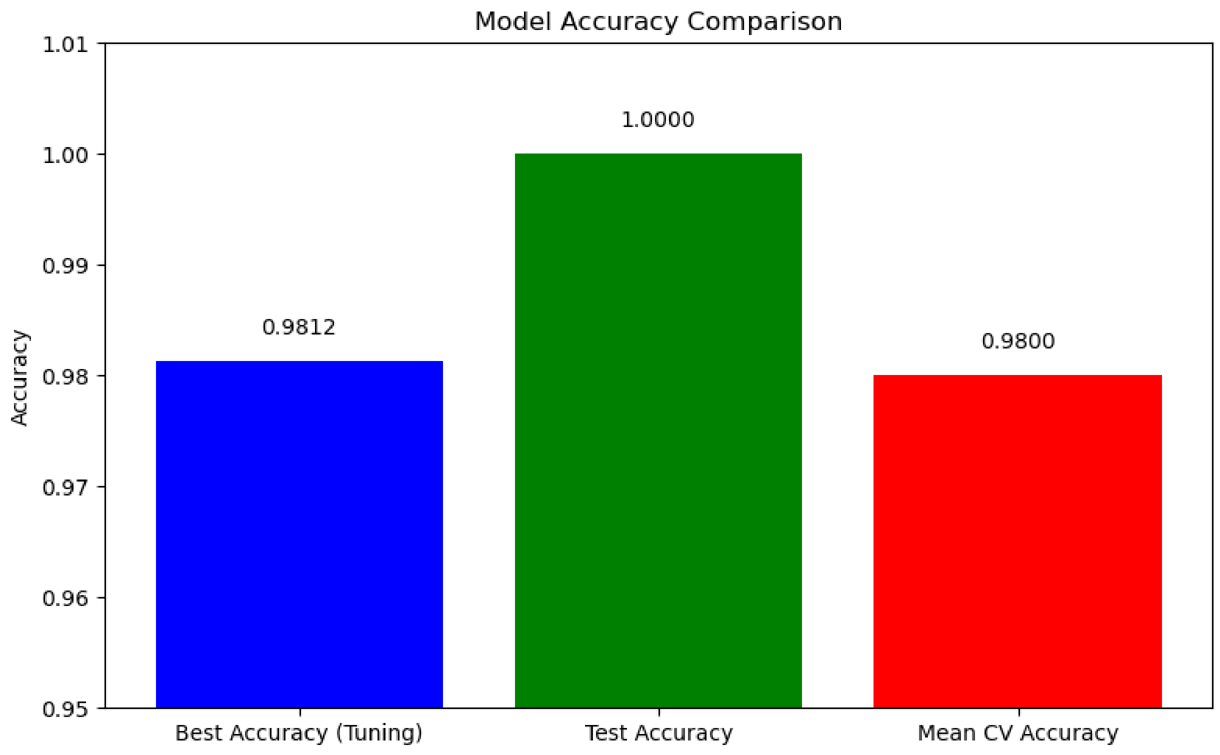
For the hyperparameters of this model, we have chosen to focus on tuning ‘n_estimators’, ‘max_depth’, ‘min_samples_split’, ‘min_samples_leaf’, and ‘max_features’. Tuning these parameters influences model complexity and the ability of the model to learn. These specific parameters were chosen as they have a significant impact on the model’s ability to generalize. Increasing ‘n_estimators’ results in an increase in the robustness of the model, meaning it’s ability to perform well across different datasets or variations in the data. This in turn should reduce variance without significantly increasing bias. Meanwhile parameters like ‘max depth’ and ‘min_samples_leaf’ help reduce overfitting by controlling the size and depth of trees.

In order to determine the best hyperparameters we employed the use of RandomizedSearchCV from Scikit learn. This tool evaluates the model using a set number of hyperparameter combinations using cross-validation. This method was chosen over grid search because it is computationally less expensive while still offering an good combination of hyperparameters in less time. As mentioned, we opted to tune our hyperparameters to optimize for accuracy given the straightforward interpretation of model performance it provides via the proportion of correct predictions. Accuracy is especially useful when the dataset is balanced which ours is.

Our Random Forest model achieved an accuracy of 0.9812 during tuning with a perfect accuracy of 1.0 on the test set. Initially we were suspicious of these results and the potential for overfitting. To address these concerns, we opted to perform 10-fold cross-validation which resulted in a mean score of 0.9800, suggesting that the model performs well in classifying these instances. These scores were achieved with hyperparameters of 'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': 'auto', and 'max_depth': 10.

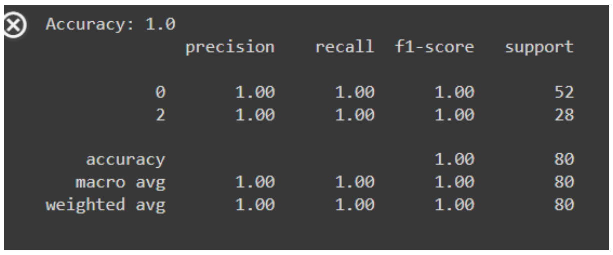
Bias-variance tradeoff is a significant issue to consider when dealing with supervised learning. Bias is the error that comes from creating a model that is too simple for the more complex real world problem it’s being used for. Variance on the other hand measures how much each prediction for a point vary between different iterations of the model.

With our results, it seems fair to say the model has low bias given the consistently high accuracy. The perfect score on the test data, however, did raise some concerns as to the possibility of overfitting which would suggest high variance. As mentioned, this is somewhat addressed by the high cross-validation score received with additional testing which suggests the model will generalize well to unseen data. It could also be the case that our dataset is overly simple or contains repetitive patterns that is making it easy for the model to predict outcomes. However, assuming that is not the case this model shows an excellent balance between bias and variance with high accuracy scores across the board. The only other way to confirm that the perfect test score is not a red flag for overfitting would be to test the model on additional data which unfortunately we do not have.



In summary, this Random Forest model is well-tuned with high accuracy and an excellent balance in the bias-variance tradeoff although concern for overfitting is legitimate and testing on additional data would be ideal. The model accuracy plot above shows the different accuracy measures providing a useful visual representation for how well the model is expected to perform on new data.

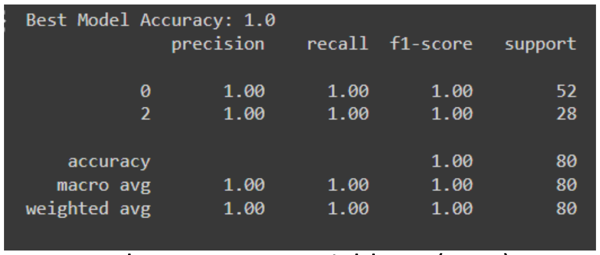
K-Nearest Neighbors:



```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        52
           2       1.00      1.00      1.00        28

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```



```
Best Model Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        52
           2       1.00      1.00      1.00        28

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

The K-nearest neighbors (KNN) algorithm is a non-parametric classification technique used for pattern recognition and predictive modeling. It classifies a data point based on the majority class of its k nearest neighbors in the feature space. KNN is a simple but effective algorithm that does not make strong assumptions about the underlying data distribution which makes it suitable for various classification tasks: predicting chronic kidney disease progression.

In our study, we used KNN classifier to predict the progression of chronic kidney disease based on a dataset comprising several clinical and biochemical features. The dataset was divided into training and testing sets using a standard 80-20 split (80 training, 20 testing), with the training set used to train the model and testing set used to evaluate the performance.

Before training the KNN model we had to standardize the features using StandardScaler to ensure each feature contributes equally to distance calculation between data points. Standardization is crucial for KNN as it relies on the distance metric to determine the nearest neighbors.

The KNN classifier achieved impressive accuracy of 100% on the testing set, which indicates that it accurately classified instances of chronic kidney disease progression with precision, recall, and F1-score metrics also excellent across both classes. This demonstrates the model’s ability to correctly identify positive and negative cases of kidney disease. The precision, which measures proportion of true positive predictions out of all instances predicted as positive (true and false), in this case was 1.0 for both classes. This indicates that all instances predicted as belonging to classes 0 and 2 are indeed correct.

Recall, also known as sensitivity, measures the proportion of true positive predictions out of all actual positive instances. Similar to precision, the recall for both classes is 1.0, indicating that the mode correctly identified all actual instances of classes 0 and 2.

F1-score, the harmonic mean of precision and recall and provides a balance measure of model's performances, for both precision and recall were 1.0 for both classes, so the score is also 1.0 for both classes.

The support, number of actual occurrences of each class in test set, in this case there were 52 instances of class 0 and 28 instances of class 2.

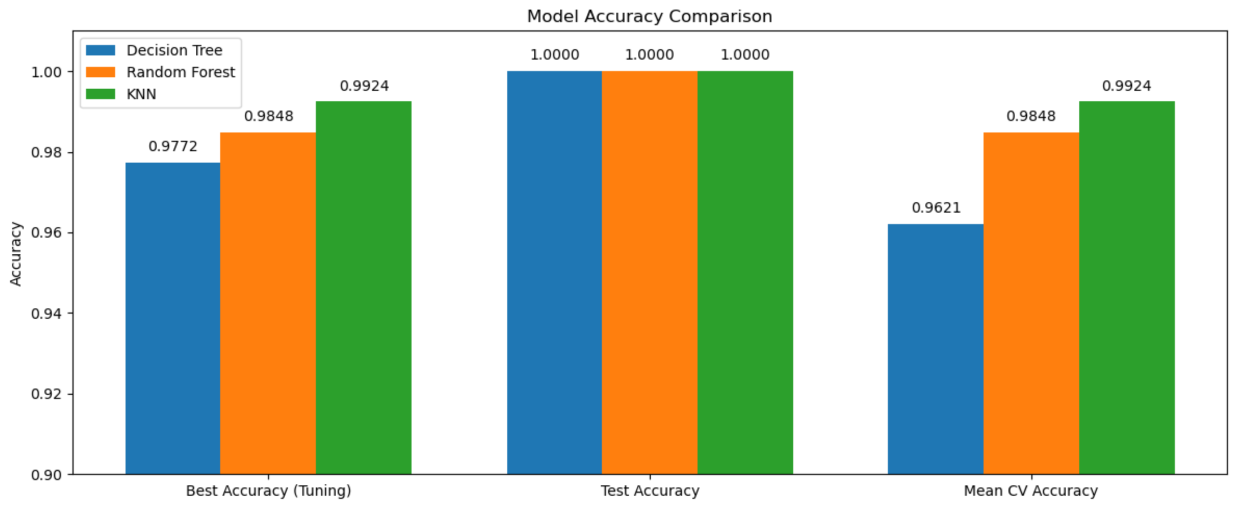
Finally, accuracy measures overall correctness of model across all classes. With accuracy of 1.0, this indicates that the model correctly classified all instances in the test set. The KNN model overall achieved perfect performance on the test data, accurately classifying all instances. This can result from a few things: well-balanced dataset, appropriate feature selection, effective hyperparameter tuning.

While the KNN classifier performed exceptionally well in my dataset, it's essential to consider potential limitations and avenues for improvement. KNN is sensitive to choice of number of neighbors (k) and distance metric used for classification. Further experimentation with different values of k and alternative distance metrics such as Manhattan or Mahalanobis distance could provide insights into improving model performance.

Additionally, future research could explore ensemble methods like weighted KNN or distance weighted KNN to incorporate strengths of multiple KNN models and potentially enhance predictive accuracy further.

In conclusion, KNN classifier demonstrates promising results for predicting chronic kidney disease progression based on clinical and biochemical features for simplicity, interpretability, and excellent performance.

Model Comparison



With this report we compared three supervised learning models for predicting chronic kidney disease – Decision Trees, Random Forests, and K-Nearest Neighbors. The models were tuned, trained, and evaluated for accuracy using our dataset the contained various clinical features. All three models performed extremely well, scoring excellently during tuning and receiving perfect accuracy on the test set. This initially raised suspicions of overfitting, and as such we ran additional cross-validation on each model to confirm the model would perform well against unseen data and again, all models performed exceptionally well. Particularly notable was the KNN model which had the highest accuracy on both tuning, at 0.9924, and on mean cross-validation accuracy, at 0.9924. Furthermore, the simplicity and interpretability of a KNN models is an additional perk over the other models we compared. While all models performed exceptionally well, these scores suggest the KNN model performed the best for this dataset. However, the differences in these scores are minimal and as such any of these models would be highly effective at predicting chronic kidney disease.