Predicting Chronic Kidney Disease with AI models

Ethan Sam and Jeffrey Wang
Northeastern University
sam.et@northeastern.edu
wang.jef@northeastern.edu
Link to code: https://github.com/jeff-d-wang/CKD_Predictor

Abstract

Chronic Kidney Disease (CKD) is a serious condition that, if detected early, can be treated and managed to prevent life-threatening complications such as kidney failure later on. We aim to demonstrate how machine learning can support the earlier diagnosis of this illness and help patients receive better care.

In this project, we developed two models to predict the presence of CKD in patients based on 24 medical attributes, including age, blood pressure, blood sugar, sodium levels, and more. Using a dataset from the UCI Machine Learning Repository, we approached this problem as a binary classification problem. The two algorithms we implemented are a Multilayer Perceptron (MLP) and a Random Forest (RF). Both methods are good for the structured nature of our medical data and have been used previously in healthcare for similar purposes.

We achieved over 80% accuracy for both models, and the RF model provided insight into the most influential features for CKD prediction.

Introduction

Chronic Kidney Disease (CKD) is a major global health issue and the cause of a large number of deaths annually. Detection of CKD can be complex, as patients often experience little to no expression of symptoms in the early stages of the disease. This often leads to delayed detection in patients, and we must rely on costly clinical tests. More cost-effective screening solutions can significantly mitigate patient risks through the intervention of healthcare professionals. Through early detection, we also reduce the need for future, expensive treatments like dialysis or kidney transplants.

Data-driven approaches offer valuable opportunities for early detection since CKD is not yet fully understood in medicine. In our project, we implemented two models to predict CKD: Random Forest (RF) and Multilayer Perceptron (MLP). RF is a tree-based ensemble method that does well at handling structured data and capturing complex, non-linear interactions between features. It allows us to identify which medical attributes most influence CKD predictions through feature importance. Additionally, RF is less prone to overfitting compared to single decision trees, which we hope will impact the generalization of the model.

The MLP model complements RF by leveraging deep learning techniques to model more intricate patterns in our data. MLPs are effective for binary classification tasks. Although they carry a risk of overfitting, we addressed this using regularization, dropout, and hyperparameter tuning.

Through this study, we aim to evaluate the effectiveness of both models in predicting CKD and identifying key features related to the disease. While our primary goal is to build an accurate, low-cost screening tool, we also seek to gain deeper insights into the most influential factors in CKD diagnosis.

Background

Dataset and Preprocessing

The dataset of the original study from Rubini et al. was obtained from the UCI Machine Learning Repository, a publicly available resource library. Donated in July 2015, the dataset contains 24 medical attributes for 400 patients collected over approximately 2 months from a hospital in India. Information on each feature in the dataset is provided in (Table 1). The final feature in the dataset, our y value, indicates whether the patient was diagnosed with CKD or not.

Table 1.

Feat.	Column Name	Description
age	Age	Age
bp	Blood Pressure	Blood Pressure in mm/Hg
sg	Specific Gravity	Ratio between urine density to water density
al	Albumin	Protein percentage in blood plasma
su	Sugar	Sugar level in blood plasma
rbc	Abnormal Red Blood Cells	Red blood cell percentage in blood plasma
pc	Abnormal Pus Cells	White blood cells in urine
pcc	Abnormal Pus Cell Clumps	Pus cell clump percentage in urine
ba	Bacteria	Bacterial presence in urine
bgr	Blood Glucose Random	Random test of glucose in blood in mg/dL

Table 1. cont.		
Feat.	**Column Name**	**Description**
bu	Blood Urea	Urea nitrogen percentage in blood plasma
sc	Serum Creatine	Creatine level in patient muscles in mg/dL
sod	Sodium	Sodium level in blood in mEq/L
pot	Potassium	Potassium level in blood in mEq/L
hemo	Hemoglobin	Hemoglobin levels in blood in g/dL
pcv	Packed Cell Volume	Volume of blood cells in a blood sample
wbcc	White Blood Cell Count	Count of white blood cells in cells/cmm
rbcc	Red Blood Cell Count	Count of red blood cells in millions/cmm
htn	Hypertension	Condition in which the blood vessels endure continuous high pressure
dm	Diabetes Mellitus	Condition in which the production or response to glucose is impaired
cad	Coronary Artery Disease	Condition in which the heart's main blood channels are impaired
appet	Appetite	If the patient has a poor appetite or not
pe	Pedal Edema	Swelling due to an injury or inflammation
ane	Anemia	Insufficient healthy red blood cells
class	CKD Label	Positive or negative chronic kidney disease diagnosis

Preprocessing the data was a necessary step before applying any machine learning algorithms. Numerical features were already in a usable format, while categorical features (such as "normal" vs "abnormal" or "yes" vs "no") were converted to binary values. Several features contained missing values, with some, like abnormal red blood cells, missing in over a third of the records. With this being a small dataset, we wanted to retain as many samples as possible. To handle this, we used K-Nearest Neighbors (KNN) imputation with $k = 3$, filling in missing values based on the most similar patients in the dataset. This technique preserves underlying patterns and reduces the risk of introducing bias from global averages.

We plotted the feature correlations in a heatmap (Figure 1) to get a first glance at its structure, which revealed strong associations between the CKD label and features like specific gravity, albumin, hemoglobin, and packed cell volume. Although these strong correlations might infer their clinical significance, they also raise concerns about the dataset's simplicity and thus, possible overfitting.

In search of more data to both validate and force the models to generalize better, we turned to a follow-up study conducted by Islam et al., which introduced a second dataset

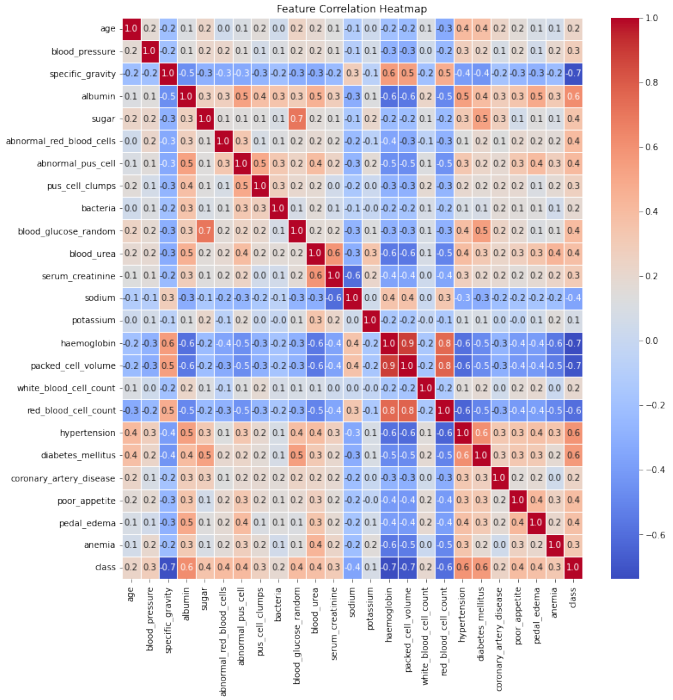

Figure 1: Feature correlation heatmap

of 200 patients collected at Enam Medical College in Bangladesh. This dataset, published in August 2023, includes the same 24 features and CKD labels. However, many of its values were expressed as string-based ranges (e.g. '$<$ 48.1' or '48-86.2') rather than precise numeric values. These ranges presented two challenges: they differed in format from the original dataset and represented intervals rather than specific values, potentially obscuring the true patient measurements.

To harmonize the datasets and generate usable inputs, we employed Kernel Density Estimation (KDE) to model the distribution of each feature from the original dataset. By placing small Gaussian kernels over each data point and summing them, KDE allowed us to estimate a smooth probability distribution for each variable. We then applied rejection sampling to synthesize values: randomly sampling points and accepting only those that satisfy the target range from the new dataset and fall under the probability curve of the original KDE-estimated distribution. This ensured that our synthetic values were statistically consistent with original observations while honoring the constraints of the new dataset.

To evaluate the effectiveness of this method, we plotted overlaid feature distributions for both datasets (Figure 2). These visualizations showed that our KDE-based sampling process succeeded in approximating the original data's characteristics, while adding slight variation to reflect the broader patient pool.

The class distribution in both datasets was also reasonably balanced. The original dataset included 250 patients with CKD and 150 without, while the newer dataset had 128

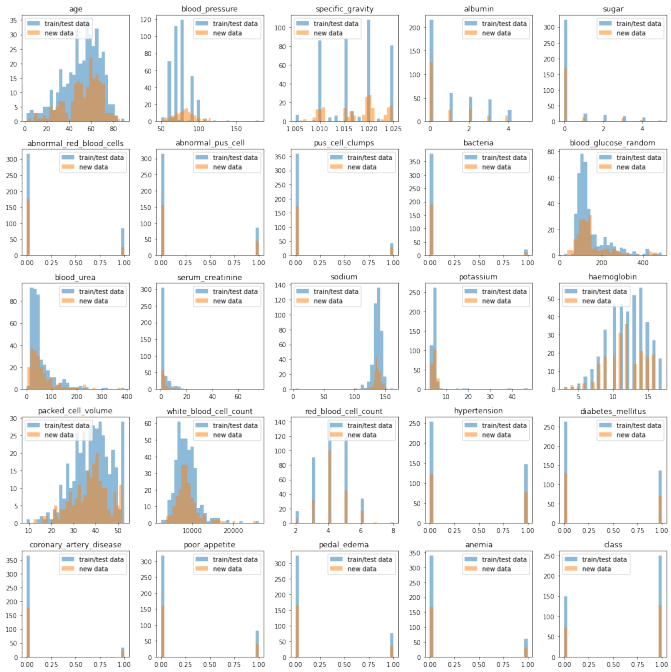

Figure 2: Feature distributions of original and new study data

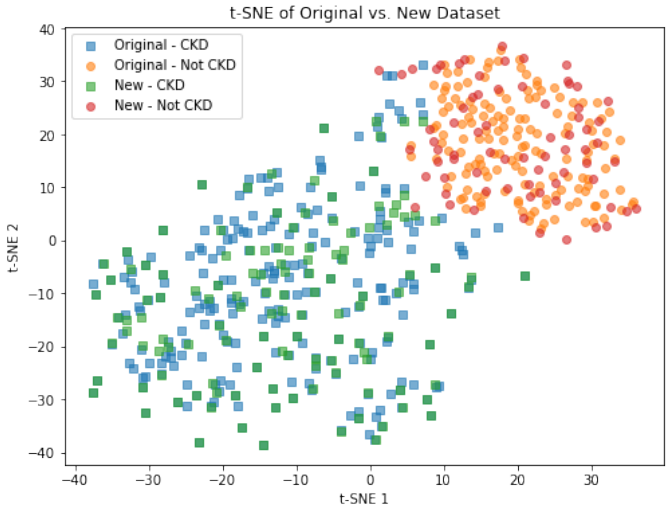

Figure 3: t-SNE of original and new study data

with CKD and 72 without, both yielding a roughly 60/40 split. To further assess whether our preprocessing introduced structural biases, we applied t-Distributed Stochastic Neighbor Embedding (t-SNE), a dimensionality reduction technique, to project all data into two dimensions (Figure 3). The resulting plot showed consistent clustering patterns between CKD-positive and CKD-negative patients across both datasets, suggesting that our synthesis and integration processes preserved meaningful diagnostic boundaries.

Random Forest

Random forest (RF) is an ensemble learning algorithm that constructs a collection of decision trees during training and outputs the mode of the classes (for classification) or mean prediction (for regression) from individual trees. Each tree in the forest is trained on a random subset of the data with replacement (bootstrap sampling) and considers a random subset of features when splitting nodes. This randomness helps reduce overfitting and increases generalization performance. We will be using the random forest for binary classification to determine whether or not a given sample indicates the presence of CKD. By tuning hyperparameters such as the number of trees, depth, and feature selection strategy, we aim to build a powerful model that can capture complex decision boundaries and accurately classify whether a sample has CKD or not.

Multilayer Perceptron

A multilayer perceptron (MLP) is a feedforward neural network that is comprised of an input layer, one or more hidden layers, and an output layer. Each layer has one or more neurons, which, upon being given a feature vector, will add weights on to each other to output a prediction. Through extensive hyperparameter tuning and architecture development, we aim to develop a model that can learn nonlinear relationships between features without overfitting to deduce if a sample has CKD or not.

Simpler Algorithms

In traditional machine learning workflows, simpler algorithms such as Logistic Regression or Decision Trees may be used as baselines. However, due to the potential complexity in medical data and interactions between features, our models of choice (RF and MLP) are better suited to capture the nuances of CKD prediction.

Together, these models provide a balance between interpretability (via Random Forests) and modeling complexity (via MLPs). By comparing their performance and examining which features each model relies on, we can better understand both the strengths of our models and the underlying structure of the data.

Related Work

Prior research has tackled the problem of CKD detection using a range of machine learning techniques. Sharma et al. adopted an ensemble learning (EL) strategy for early CKD identification (Sharma et al., 2023), while Rashed-Al-Mahfuz et al. employed a suite of machine learning models but without collecting them in an ensemble framework (Rashed-Al-Mahfuz et al., 2021).

In the approach by Sharma et al., five individual classifiers that assume linear relationships between features were combined using a majority voting mechanism (Sharma et al., 2023). Each model independently predicts the presence or absence of CKD, and the final prediction is determined by the most frequent class among them. For example, if a potential output from the stacked model looks like the following: 0, 1, 0, 0, 0 (where 0 indicates absence and 1 indicates

presence of CKD). In this case, the ensemble will classify this patient as NOT CKD. The strength of this method lies in the diversity of the models in the ensemble, which make different assumptions about the data, theoretically negating the weaknesses each model has and thus errors. Moreover, the aggregation mechanism helps mitigate overfitting, a very apparent problem with the CKD dataset due to its small size and potential for class imbalance.

Rashed-Al-Mahfuz et al. also utilized five machine learning models. Still, they instead focused on solving the issue of accurate early diagnosis of CKD by identifying key pathological categories and optimizing datasets around these selected attributes (Rashed-Al-Mahfuz et al., 2021). This led them to use Random Forest (RF), Gradient Boosting (GB), XGBoost (XGB), Logistic Regression (LR), and Support Vector Machine (SVM) over six different datasets. The six different datasets included different permutations of blood, urine, and other (excluding blood and urine) tests.

Rashed-Al-Mahfuz et al. approached the problem from a feature engineering and data segmentation perspective. Their work involved training five different models: Random Forest (RF), Gradient Boosting (GB), XGBoost (XGB), Logistic Regression (LR), and Support Vector Machines (SVM) (Rashed-Al-Mahfuz et al., 2021). These models were trained and tested across six distinct datasets. Each dataset focused on a different set of medical attributes, such as blood test results, urine metrics, or a combination of both. Their emphasis was on optimizing diagnostic performance for early-stage CKD by identifying and selecting key pathological features that were most relevant.

In comparison, we developed a Random Forest (RF) classifier and a custom-implemented Multi-Layer Perceptron (MLP). While we considered ensemble and boosting-based methods like those used by Sharma et al. and Rashed-Al-Mahfuz et al., we chose to focus on RF for its robustness against overfitting and interoperability, and a neural network (MLP) for its flexibility in modeling complex, non-linear relationships among features. We avoided ensemble methods primarily to maintain model interpretability and deeper neural networks, which posed more of a risk of overfitting without significant performance gains due to our dataset size.

Project Description

Random Forest

Key Concepts:

- Bootstrap Aggregation (Bagging): Each tree in the forest is trained on a randomly sampled subset (with replacement) of the training data.
- Feature Randomness: At each split in the tree, a random subset of features is considered rather than evaluating all features. This introduces more diversity among trees and reduces correlation.
- Voting: For classification tasks, the forest takes a majority vote among the trees to determine the final class prediction.

Mathematical Equations

Gini Impurity: For a dataset D with K classes, the Gini impurity is defined as:

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} p_k^2$$

where p_k is the proportion of samples belonging to class k within the node.

At each split in the decision tree, the Gini impurity of the left and right branches is calculated, and the weighted average is used to determine the quality of the split:

$$\text{Gini}_{\text{split}} = \frac{n_{\text{left}}}{n} \cdot \text{Gini}(D_{\text{left}}) + \frac{n_{\text{right}}}{n} \cdot \text{Gini}(D_{\text{right}})$$

Here, n is the total number of samples at the node, and n_{left} and n_{right} are the number of samples in the left and right child nodes, respectively.

Majority Voting (Final Prediction): In a random forest, the final prediction $\hat{y}$ for an input x is made using majority voting across all T trees:

$$\hat{y} = \text{mode}\left(\{h_t(x)\}_{t=1}^{T}\right)$$

where $h_t(x)$ is the prediction from the t-th decision tree in the ensemble.

Pseudocode:

Algorithm 1 Decision Tree

function FIT($X, y,$ max_depth)
    Store number of classes and feature names
    Build tree using GROWTREE($X, y, 0$)
end function
function GROWTREE($X, y,$ depth)
    Count samples per class; predict most frequent class
    if depth $<$ max_depth **then**
        Find best split feature j^* and threshold s^* via BESTSPLIT(X, y)
        if split is found **then**
            Partition data into X_L, y_L and X_R, y_R
            Create left and right subtrees recursively
        end if
    end if
    return node with predicted class, split info, left, and right
end function
function BESTSPLIT(X, y)
    for each feature j **do**
        Sort (X_j, y) by X_j
        for each threshold s **do**
            Split y into left/right groups
            Compute Gini impurity for split
            if impurity is best so far **then**
                Update best split
            end if
        end for
    end for
    return best feature index j^* and threshold s^*
end function
function PREDICT(x)
    Traverse tree nodes using feature/threshold splits
    return predicted class at leaf node
end function

Algorithm 2 Random Forest

function FIT($X, y, T,$ max_depth)
    for $t = 1$ to T **do**
        Sample bootstrap dataset $(X^{(t)}, y^{(t)})$ from (X, y)
        Train decision tree $h_t =$ DecisionTree()
        h_t.FIT($X^{(t)}, y^{(t)},$ max_depth)
        Add h_t to forest
    end for
end function
function PREDICT(x)
    Let $P = \{h_t.\text{PREDICT}(x)\}_{t=1}^{T}$
    return majority class in P
end function

Multilayer Perceptron

Our MLP consists of

- Input layer $\in \mathbb{R}^{24}$, $A^{(1)} = X$
- Hidden layer #1 $\in \mathbb{R}^{16}$, $A^{(2)} = ReLU(Z^{(2)})$
- Hidden layer #2 $\in \mathbb{R}^{8}$, $A^{(3)} = ReLU(Z^{(3)})$
- Output layer $\in \mathbb{R}^{1}$, $A^{(4)} = \sigma(Z^{(4)})$

$Z^{(l)}$ and how it is calculated are detailed in the **mathematical equations** section. A visual representation of the MLP is shown in (Figure 4).

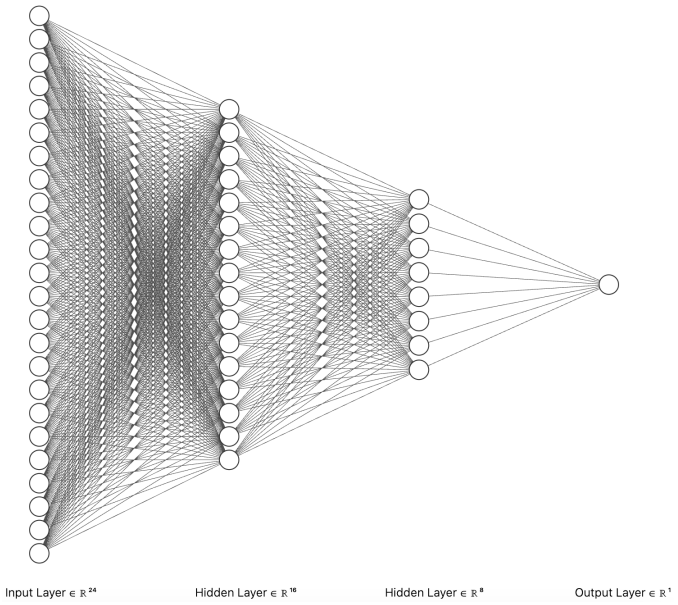

Figure 4: MLP architecture

Mathematical Equations

Forward propagation: Forward propagation computes the predicted output $\hat{y}$ from input X through connected layers of neurons, each with an activation function detailed above.

- **Post-activation**
  For each layer l, given its weight matrix $W^{(l)}$, bias vector $b^{(l)}$, and input activation $A^{(l-1)}$ (specified above) for $l > 1$, we calculate
  $Z^{(l)} = A^{(l-1)}W^{(l)} + b^{(l)}$
- **Dropout Regularization**
  We applied dropout to each hidden layer, which randomly disables $p = 0.5$ of the neurons during each training epoch. This prevents the model from relying on specific neurons, forcing it to generalize and thus mitigate overfitting. We update
  $A^{(l)} = \frac{A^{(l)} \odot D^{(l)}}{1-p}, D^{(l)} \sim \text{Bernoulli}(1-p)$
- **Loss Function**
  We used binary cross-entropy loss with L2 penalty to quantify model accuracy and mitigate overfitting. For truth values y, predictions $\hat{y}$, and L2 penalty $\lambda = 0.001$, we calculate
  $L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i + \epsilon) + (1-y_i) \cdot \log(1-\hat{y}_i + \epsilon)] + \lambda \sum_l \|W^{(l)}\|^2$

where the first term is the binary cross-entropy loss formula and the second term is the L2 penalty.

Backpropagation: After calculating the L, the MLP needs to compute the gradient of the loss function relative to each parameter.

- **Gradient Computation**
  It begins at the output layer, where we calculate the initial error
  $\delta^{(3)} = \frac{\partial L}{\partial A^{(3)}} \cdot \sigma'(Z^{(3)}) = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})+\epsilon} \cdot A^{(3)}(1-A^{(3)})$
  Then the hidden layers where we continue to identify the weights that contributed to the error
  $\delta^{(2)} = (\delta^{(3)}(W^{(3)^T})) \cdot ReLU'(Z^{(2)})$
  $\delta^{(1)} = (\delta^{(2)}(W^{(2)^T})) \cdot ReLU'(Z^{(1)})$
  Where $ReLU'(Z^{(l)}) = \begin{cases} 1 & Z^{(l)} > 0 \\ 0 & Z^{(l)} \leq 0 \end{cases}$
- **Weight Gradient and Update**
  Upon computing the gradients, we need to update the weights in the direction that reduces the loss
  $\frac{\partial L}{\partial W^{(l)}} = (A^{(l-1)})^\top \delta^{(l)} + 2\lambda W^{(l)}$
  $W^{(l)\prime} = W^{(l)} - \alpha \frac{\partial L}{\partial W^{(l)}}$ for $\alpha = 0.01$
- **Bias Gradient and Update**
  As well as the biases
  $\frac{\partial L}{\partial b^{(l)}} = \sum_{i=1}^{N} \delta_i^{(l)}$
  $b^{(l)\prime} = b^{(l)} - \alpha \frac{\partial L}{\partial b^{(l)}}$ for $\alpha = 0.01$

Pseudocode:

Algorithm 3 Multilayer Perceptron

function TRAIN(X, y)
  Initialize weights $W^{(l)}$ and biases $b^{(l)}$ randomly
  for epoch = 1 to 1000 **do**
    Forward pass: compute activations $A^{(l)}$ for each l
    Apply dropout to hidden layers during training
    Compute loss: $L = \text{BinaryCrossEntropy}(y, \hat{y}) + \lambda \sum_l ||W^{(l)}||^2$
    Compute output error: $\delta^{(3)} = \hat{y} - y$
    for layer l = 2 down to 1 **do**
      $\delta^{(l)} = (\delta^{(l+1)}(W^{(l+1)})^T) \cdot ReLU'(Z^{(l)})$
    end for
    Update weights: $W^{(l)\prime} = W^{(l)} - \alpha \frac{\partial L}{\partial W^{(l)}}$
    Update biases: $b^{(l)\prime} = b^{(l)} - \alpha \frac{\partial L}{\partial b^{(l)}}$
  end for
  return trained weights $W^{(l)}$, biases $b^{(l)}$
end function
function PREDICT(X)
  Perform forward propagation on X
  return class prediction based on threshold of 0.5
end function

Experiments

We began by using an original CKD dataset, which we split into training and testing subsets (typically an 80/20 split). After training our model on the training set, we evaluated its accuracy on the testing set. Surprisingly, both of our models achieved an exceptionally high accuracy (e.g., roughly 98%). While this initially appeared promising, it raised concerns about potential overfitting. To investigate this probability, we searched for additional datasets to validate our model's performance in a more realistic setting.

To assess our models' generalization capability, we obtained a separate CKD dataset from a different source and cleaned it using the methods mentioned in the background section. We retrained our model on the original dataset and tested it on the new dataset.

Random Forest

For the RF model, the accuracy dropped significantly (e.g., from 98% to 75 %) when testing on the new dataset, indicating that the original performance was inflated by overfitting and a lack of dataset diversity. We can see some of that overfitting in (Figure 5). While Random Forests are generally resistant to overfitting, they can still memorize patterns specific to the training data when that data lacks variability or includes features that correlate strongly within a narrow sample. In our case, both datasets contain patient data from different populations, whose underlying risk factors for CKD may differ significantly.

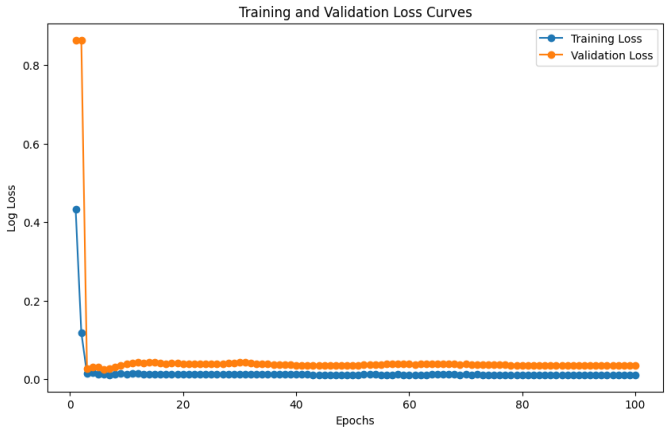

Figure 5: Training and validation loss of RF

Age was initially one of the most heavily weighted features in the Random Forest model (Figure 6), indicating that it played a dominant role in the model's decision-making process. This reflects a real-world correlation, since age is often associated with increased risk of chronic kidney disease (CKD). However, an over-reliance on a single feature can lead to overfitting, so we experimented with penalizing the influence of age during training. This hopefully encourages the model to consider a broader set of features and reduce its dependency on any single variable.

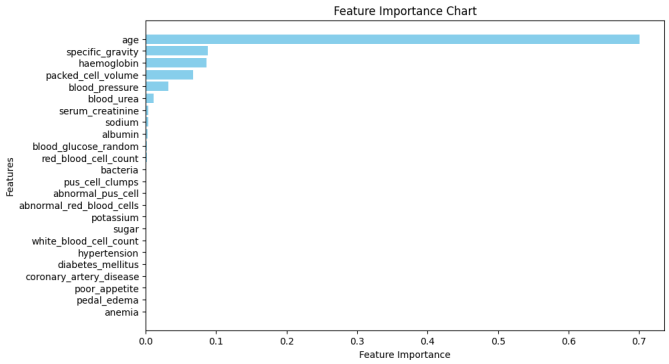

Figure 6: Feature importance chart before age penalty

However, this approach did not result in any noticeable improvement in the model's accuracy; in fact, it dropped it by about 4-5%, as shown in (Figure 7). One possible explanation is that age captures a significant portion of the predictive signal in the original dataset, and reducing its influence without introducing equally informative features may have weakened the model overall. It's also possible that the method of penalization wasn't effective enough to redistribute the model's focus to other meaningful patterns in the data. This outcome suggests that we should try not to artificially constrain the model and instead look for more natural ways to improve our dataset, such as collecting a wider array of samples.

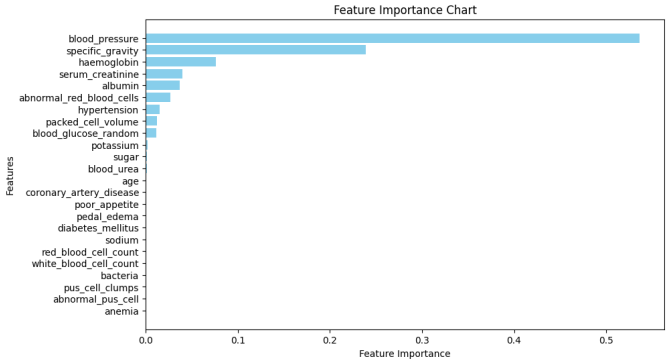

Figure 7: Feature importance chart after age penalty

Multilayer Perceptron

We started building from a simple MLP with a single hidden layer of 8 neurons with ReLU, then added complexity and features when necessary. It overfitted very early into training, however, so we began adding more neurons, dropout, and tuning hyperparameters. We had hoped that increasing neurons would force the model to generalize better, but this model's loss curve graph in (Figure 8) is indicative of overfitting despite the harsh dropout rates and ridge penalty applied to the design. It achieved a test accuracy of 98.75%, with it mistakenly predicting a false positive on a single patient.

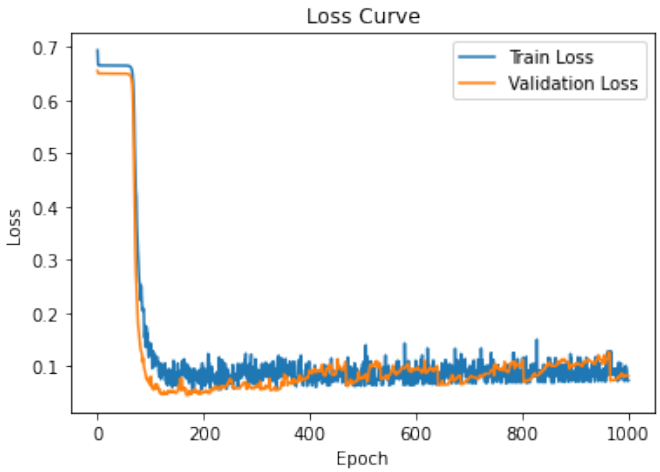

Figure 8: Training and validation loss of MLP

Upon validation with the new dataset, it achieved an accuracy of 97%, mistakenly predicting 6 false positives. We believe the model fails to predict CKD when the given feature vector has values stratifying away from popular values according to the feature distribution graphs. The variance and randomness of the new dataset from our data imputation methods contribute to this, but we believe this is more reflective of sampling from populations, as values aren't perfect. The original dataset had features with neat, integer values like being only multiples of 5, but the model seems to be lenient on small noise.

Through permutation feature importance, a process by which we randomly shuffle values of a single feature to observe its effect on the fitted model's accuracy, we have identified the top 5 key features with medical implications that the MLP relies on for prediction in (Table 2).

Table 2.

Key Features	**Feature Importance Score**
Specific Gravity	0.100
Hypertension	0.088
Pedal Edema	0.038
Haemoglobin	0.025
Blood Pressure	0.013

Conclusion

Overall, our models performed well on the datasets we tested them on, and the insights we gathered from our feature importance analyses aligned with established medical knowledge about Chronic Kidney Disease. Features such as age, blood pressure, and specific gravity consistently ranked high in importance. Individuals aged 65 and above are at much greater risk for CKD due to the aging of their kidneys aging which weakens their ability to filter waste (CDC, 2023). High blood pressure is another feature that can lead to a greater risk for CKD because high blood pressure constricts and damages blood vessels throughout the body, including the kidneys. When the kidneys' blood vessels are damaged, the kidneys may stop functioning, leading to more

waste and fluids in the blood and ”creating a dangerous cycle, and cause more damage leading to kidney failure” (NIDDK, 2025). Finally, specific gravity refers to the density of urine compared to the density of water. We believe this is a significant predictor due to the kidneys' role in filtering waste from the blood to produce urine, which directly affects urine concentration. These medical attributes we extracted can be readily measured and evaluated through routine patient checkups, which are clinically relevant. To strengthen our case, future work on this study would be centered around increasing the geographic and demographic diversity of patients in training data and validating our results.

Through this project, we learned how machine learning models can effectively support medical diagnosis by identifying patterns and risk factors associated with chronic illnesses. We also gained hands-on experience with preprocessing real-world clinical data, handling missing values, combining data sampled differently, and working to improve model accuracy. We demonstrated the potential of AI in healthcare, especially for early detection and risk prediction of CKD. Thoughtful feature selection, data quality, and model transparency when building models can aid healthcare providers in making well-informed decisions for their patients.

References

CDC Chronic kidney disease in the United States, 2023. In: Centers for Disease Control and Prevention. https://www.cdc.gov/kidney-disease/php/data-research/index.html. Accessed 21 Apr 2025

Islam MA, Akter S, Hossen MS, et al (2023) Risk factor prediction of chronic kidney disease. In: UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/857/risk+factor+prediction+of+chronic+kidney+disease. Accessed 21 Apr 2025

NIDDK High blood pressure & kidney disease. In: National Institute of Diabetes and Digestive and Kidney Diseases. https://www.niddk.nih.gov/health-information/kidney-disease/high-blood-pressure. Accessed 21 Apr 2025

Rashed-Al-Mahfuz, M., Haque, A., Azad, A., Alyami, S. A., Quinn, J. M. W., and Moni, M. A. 2021. Clinically Applicable Machine Learning Approaches to Identify Attributes of Chronic Kidney Disease (CKD) for Use in Low-Cost Diagnostic Screening. IEEE Journal of Translational Engineering in Health and Medicine, 9: 1–11. Article No. 4900511. https://doi.org/10.1109/JTEHM.2021.3073629. Accessed 21 Apr 2025

Rubini L, Soundarapandian P, Eswaran P (2015) Chronic kidney disease. In: UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease. Accessed 21 Apr 2025

Sharma, D. 2023. A Machine Learning Methodology for Classifying Chronic Kidney Diseases. Retrieved from https://www.ubishops.ca/wp-content/uploads/sharma20240106.pdf. Accessed 21 Apr 2025