# Supervised Learning Modeling to Predict Chronic Kidney Disease

1st Nathan Theng
*Department of Computer Science*
*California State University, Fresno*
Fresno, California USA
theng_nathan@mail.fresnostate.edu

***Abstract*—The use of AI, specifically classification and prediction has gained significant attention in recent years in medicine and healthcare as it is crucial to predict early diagnosis of adverse disease and condition. With continuous traditional specific laboratory testing for diseases such as MRI and complete blood panels can be expensive and not as widely available to an individual rural area, supervised learning techniques can be employed based on previous lab data to predict if an individual in prone to a certain disease. This paper presents four supervised learning techniques and models to predict specifically Chronic Kidney Disease.**

**The four supervised learning techniques and models are Decision Tree, Random Forest, Support Vector Machine, and Deep Neural Network (MultiLayer-Perceptron). These four proposed models were trained on a data set specifically geared toward Chronic Kidney Disease with 24 different features and one target classification that will be pre-processed beforehand and split into training, testing, and validation data for the models to train and test on.**

**The results of the models were shown via a confusion matrix in which the accuracy, precision, recall, and F1-Score were calculated along with the execution time for each model. The results demonstrated that the the deep neural network had the best performance with an F1 Score of 97.87% and 96.97% for predicting if individuals had chronic kidney disease and individuals who did not have chronic kidney disease respectively. The neural network also had an execution time of roughly two seconds. The models that are used from the chronic kidney data set can be applied to other diseases and varying amounts of features to further advance AI in medicine.**

## I. Introduction

Chronic Kidney Disease is defined as a long-term condition where the kidney functionality erodes over time leading to unfiltered waste to build up and fluid to remain in the bloodstream. This can not only lead to kidney failure but also to heart disease and strokes. However, Chronic Kidney Disease is very difficult to predict early and the symptoms develop slowly and aren't specific to the disease. This is the case that even some people have no symptoms at all and are usually diagnosed with specific blood and urine tests that can be expensive on top of previous lab testing and results. The main question that this paper will try to answer is if we are able to predict if an individual has chronic kidney disease based on previously tested lab results. By doing so, individuals would not need to request expensive specific lab tests to be diagnosed with chronic kidney disease.

However, while it is difficult to recognize chronic kidney disease from just the symptoms as the symptoms are individualist to each person. Thus, the features for previous lab results are used as the features that the models will take in as inputs. The research design will start with obtaining the data set for chronic kidney disease with different features. The data set will need to be data pre-processed by converting the data, filling in the empty and null values, and splitting the data into training, testing, and validation data sets. These different data sets will be fed to the four different classification models: Decision Tree, Random Forest, Support Vector Machine, and Neural Network. Once these

models are created and executed, the results will be computed in forms of accuracy, precision, recall, and F1 Score. Finally, a comparison will be made between all of the different models to determine which model worked the best for this data set and disease

## II. Data

### A. Dataset

The Chronic Kidney Disease Dataset [1] was taken from the University of California, Irvine Machine Learning Repository. The data included 400 different instances and individuals and had 24 different features, and 1 target variable: CKD/NotCKD (CKD stands for Chronic Kidney Disease). Out of the 24 features, 14 numerical features included age, sugar, specific gravity, albumin, blood pressure, blood glucose random, blood urea, serum creatinine, sodium potassium, hemoglobin, packed cell volume, and white blood cell count. The remaining 10 features are categorical features which include: red blood cells, pus cells, pus cell clumps, bacteria, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema, and anemia. However, it should be noted that out of the 400 cases, 250 individuals have chronic kidney disease and the remaining 150 individuals do not have chronic kidney disease. However, it should be noted that the data set has many missing values for different features that will need to be filled in later on.

### B. Pre-processing and Training/Testing Split

For all categorical features, the data was labeled encoded such as giving it numerical values like 0 and 1 so that the models can better interpret them

It is important to remember that there are missing values throughout the data set. There are only 158 instances where all of the features are fully completed. This represents only 39.5% of the entire data set. This represents a small portion of the data so the missing values in the features must be filled.

If the missing value belongs to a numerical feature, the missing data is replaced by the mean of that specific feature. By using the mean, it helps to preserve the central tendency of the data. Thus, if the mean of the average value of the feature represents the central tendency, imputing the missing values with the mean could represent that the imputed values are similar to the observed values. This also helps retain all the observations within the data set rather than cleaning and deleting the rows with missing values, which, in this case, will lead to a significant loss of 60.5%. However, by using the mean, it assumes the data is normally distributed, even though it is not, and can introduce biases if that feature is related to other features in the data set.

if the missing value belongs to a categorical feature, the missing data was replaced by the median of that specific feature. The reason why the mode was used was because it help maintain the distribution of the categorical features, By using the most frequent category, we are assuming that the missing value most likely belongs to the prevalent category. The mode is also less sensitive to outliers making it more robust in situations where imputation methods are heavily influenced by outliers. However, this means we are assuming that the occurrence of the missing value is independent of the values itself.

Listed below is the number of missing values for the numerical features and categorical features.

| Feature | # Missing Values |
|---|---|
| Age | 9 |
| Blood Pressure | 12 |
| Specific Gravity | 47 |
| Albumin | 46 |
| Sugar | 49 |
| Blood Glucose Random | 44 |
| Blood Urea | 19 |
| Serum Creatinine | 17 |
| Sodium | 87 |
| Potassium | 88 |
| Hemoglobin | 52 |
| Packed Cell Volumne | 71 |
| White Blood Cell Count | 106 |
| Red Blood Cell Count | 131 |

TABLE I: Missing Values for Numerical Value

| Feature | # Missing Values |
|---|---|
| Red Blood Cell | 152 |
| Pus Cell | 65 |
| Pus Cell Clumps | 4 |
| Bacteria | 4 |
| Hypertension | 2 |
| Diabetes Mellitus | 2 |
| Coronary Artery Disease | 2 |
| Appetite | 1 |
| Peda Edema | 1 |
| Anemia | 1 |

TABLE II: Missing Values for Categorical Value

It should be noted that for future reference within the models, a 0 represents that the individuals have Chronic Kidney Disease (CKD) and a 1 represents that the individuals do not have Chronic Kidney Disease (NotCKD).

### C. Training, Testing, Validation Split

In order to run the models, the data must be split into training, testing, and validation data sets. However, the validation set was only needed for the neural network model. The training and testing split for Decision Tree, Random Forest, and Support Vector Machine was a 70/30 split. This meant out of the total data set 70% of the data set was used for training and 30% of the data set was used for the testing. The training, validation, and testing split for Neural Network was a 50/20/30 split. This meant out of the total data set 50% of the data set was used for training, 20% of the data set was used for validation, and 30% of the data set was used for testing.

## III. Models

In order to create these models various libraries we used. Specifically in order to create the Decision Tree, Random Forest, and Support Vector Machine Model, the sklearn library was used. In order to create the deep Neural Network, MultiLayer Perceptron, model, the tensorflow and keras libraries were used.

### A. Decision Tree Model

A Decision Tree Model is a supervised Learning algorithm that depicts decisions and their potential outcome. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Out of the 24 different features, it will pick one of the features to be the root node ( the most important feature) it will then split to the next important feature by priority determined by the Gini Impurity Index. The Gini Impurity is one of the various methods that decide the optimal split from a root node to subsequent splits. The Gini Index varies between 0 and 1 where 0 represents a very pure classification, 0.5 represents an equal distribution of elements across the classes and 1 represents a random distribution of the elements between the classes. Below is an equation to calculate the Gini Impurity Index.

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{1}$$

Below is the Decision Tree that was produced by the model

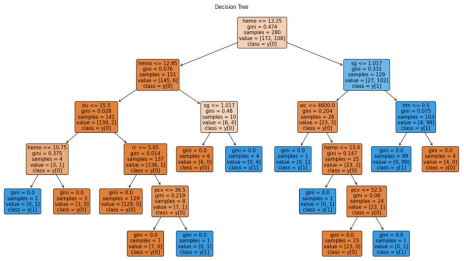


Fig. 1: Decision Tree Model

The accuracy, precision, recall, and F1 score was calculated as well.

| Measure | CKD | NotCKD |
|---|---|---|
| Accuracy | 92.50% | 92.50% |
| Precision | 97.26% | 85.01% |
| Recall | 91.02% | 95.24% |
| F1 Score | 94.14% | 89.83% |

TABLE III: Decision Tree Results

The Decision Tree Model has a sub-zero execution time as well. A confusion matrix was also created to visualized the model's performance as seen below:

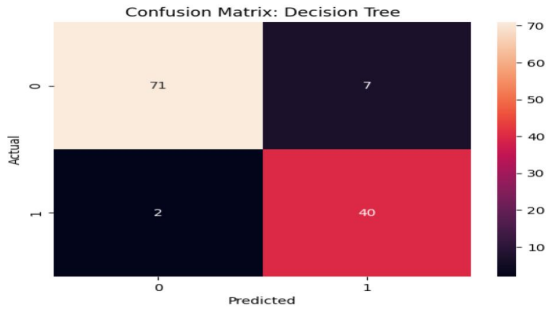


Fig. 2: Decision Tree Confusion Matrix

This confusion matrix shows that the Decision Tree model was able to predict individuals who had chronic kidney disease correctly with individuals who actually had chronic kidney disease 71 times. The model was also able to predict individuals who did not have chronic kidney disease correctly with individuals who did not have chronic kidney disease 40 times. It predicted an individual would not have chronic kidney disease, when in fact that individual did have chronic kidney disease seven times. It also predicted an individual would have chronic kidney disease, when in fact that individual did not have chronic kidney disease twice.

### *B. Random Forest Model*

A Random Forest Model is a supervised learning model that acts like an extension of a decision tree model. Instead of one decision tree, multiple decision trees which is why it is called a forest. The model will combine the output of multiple decision trees to research one single result. Each decision tree will have a different structure and different splitting of the features. For the model, I used the following parameters: A maximum tree depth of 10, a minimum sample leaves of 10, a minimum sample split of three, and 70 different decision trees. These parameters were selected by using the GridSearchCV function within the sklearn library. This function will search through different ranges for each parameter, to find the best possible combination to achieve the highest accuracy. The ranges for each of the parameters are shown below.

- Maximum Depth Range: 1 to 20
- Minimum Sample Leaves: 1 to 10
- Minimum Sample Split: 1 to 10
- Number of Tree: 1 to 100, increments of 10

The reason the number of trees is from 10 to 100 with increments of 10 is that the run time for the function to calculate all 100 different possibilities is too high, thus only 10 different values were used. Below are the first five different decision trees in the Random Forest as well as the performance results and confusion matrix.

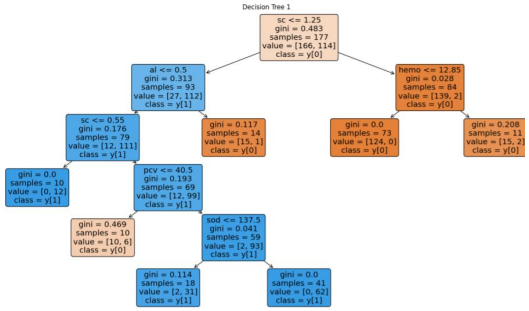


Fig. 3: Random Forest Tree 1

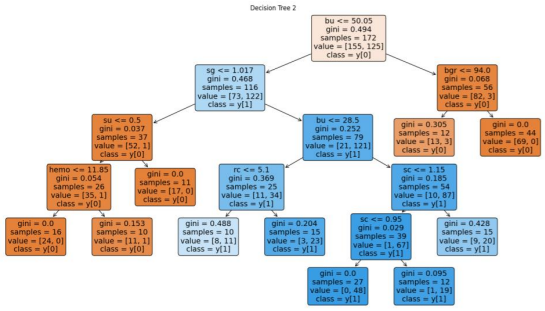


Fig. 4: Random Forest Tree 2

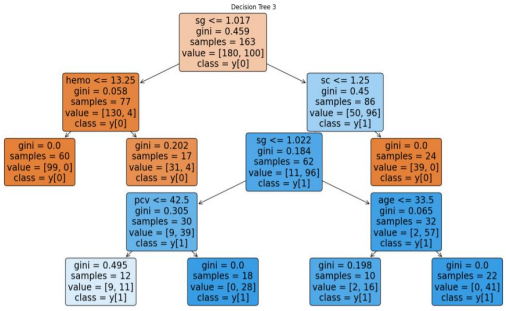


Fig. 5: Random Forest Tree 3

| Measure | CKD | NotCKD |
|---|---|---|
| Accuracy | 96.67% | 96.67% |
| Precision | 97.44% | 95.24% |
| Recall | 97.44% | 95.24% |
| F1 Score | 97.44% | 95.24% |

TABLE IV: Random Forest Results

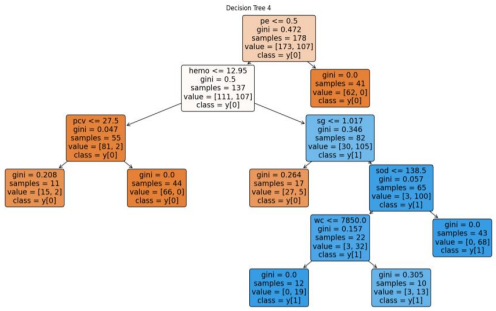


Fig. 6: Random Forest Tree 4

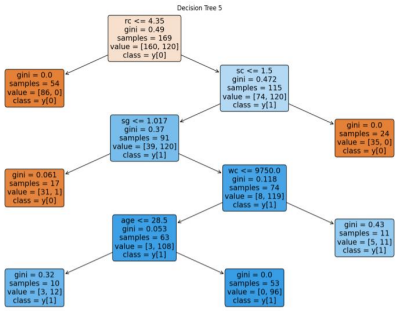


Fig. 7: Random Forest Tree 5

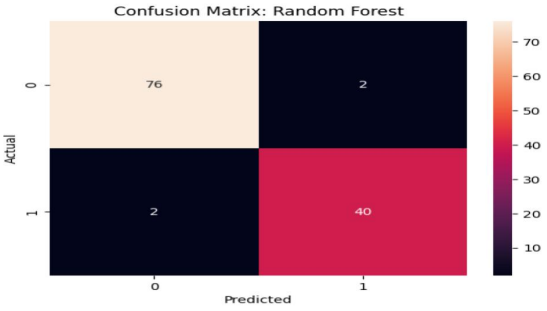


Fig. 8: Random Forest Confusion Matrix

This confusion matrix shows that the Random Forest model was able to predict individuals who had chronic kidney disease correctly with individuals who actually had chronic kidney disease 76 times. The model was also able to predict individuals who did not have chronic kidney disease correctly with individuals who did not have chronic kidney disease 40 times. It predicted an individual would have chronic kidney disease, when in fact that individual did not have chronic kidney disease twice. It also predicted an individual would not have chronic kidney disease, when in fact that individual had chronic kidney disease twice. The Random Forest Model has a sub-zero execution time as well.

### *C. Support Vector Machine*

The third model was a support vector machine. The support vector machine can classify. The machine can do this by finding the optimal hyperplane to best separate data belonging to different classes in the feature space. A hyperplane is a line or plane that will maximize the different target classifications, CKD or NotCKD. From the Seaborn Diagram see below, it is clear that to make the the decision boundary between the different classes, it will be a complex or non-linear plane.

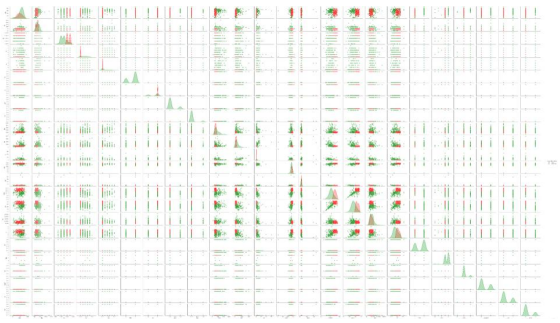

Fig. 9: Seaborn Diagram

This means that the two classifications which can be represented by the red and green dot, can not be separated by a linear line as there many data points seem to be complex and non-linear. Because of this, the support vector machine will take in a radial basis function as a parameter to create a non-linear decision boundary. Which can be seen in Figure 10 [2].

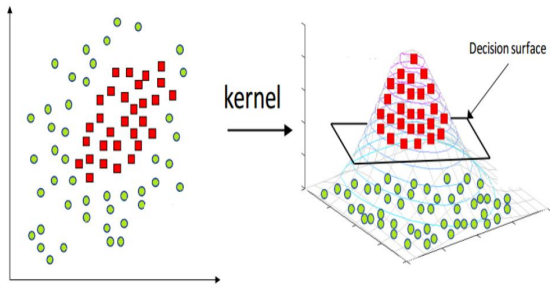


Fig. 10: Radial Basis Function Mechanism

The performance and the confusion for the Support Vector Machine can be seen below as well.

| Measure | CKD | NotCKD |
|---|---|---|
| Accuracy | 96.67% | 96.67% |
| Precision | 100% | 91.30% |
| Recall | 94.87% | 100% |
| F1 Score | 97.87% | 96.97% |

TABLE V: Support Vector Machine Results

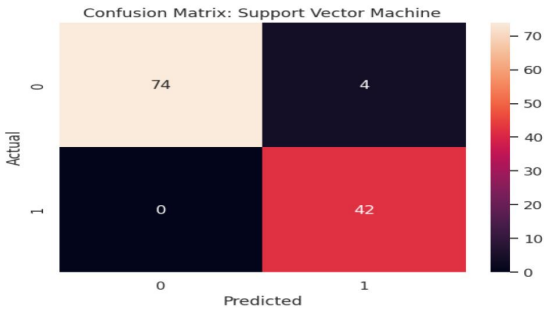


Fig. 11: Support Vector Machine Confusion Matrix

This confusion matrix shows that the Support Vector Machine model was able to predict individuals who had chronic kidney disease correctly with individuals who actually had chronic kidney disease 74 times. The model was also able to predict individuals who did not have chronic kidney disease correctly with individuals who did not have

chronic kidney disease 42 times. It predicted an individual would have chronic kidney disease, when in fact that individual did not have chronic kidney disease zero times. It also predicted an individual would not have chronic kidney disease, when in fact that individual had chronic kidney disease four times. The Random Forest Model has a 45-second execution time as well, which is significantly higher than the Decision Tree Model and Random Forest Model.

### *D. Deep Neural Network*

The fourth and final model is the Deep Neural Network Model. This is a Deep Learning technique that is inspired by the human brain. In our biological brain, every person has neurons that take inputs in the dendrites, the information is then taken by the cell body and has to pass an activation level in order to send the signal to the next neuron. A computer neuron acts very similar to that of a biological neuron. The inputs are neurons that send a signal to the transfer function which acts like the cell body. There is also an activation function associated with the neuron in which the signal must activate in order to send the signal to the next neuron. Within neural networks, there are various of these neurons in layers. From the input layer to different hidden layers with different weights and activation functions. Finally, after the hidden layers, there is an output layer that will determine the result of the model. For chronic kidney disease data sets, our model has one input layer, five hidden layers, and one output layer. The input layer has 24 different inputs each representing a feature. Within, the hidden layers, the first hidden layer has 50 hidden neurons, the second hidden layer has 40 hidden neurons, the third hidden layer has 30 hidden neurons, the fourth hidden layer has 20 hidden neurons, and the fifth hidden layer has 10 hidden neurons. All of the hidden neurons have the Rectified Linear Unit (ReLU) Activation Function. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as f(x)=max(0,x). In the output layer, there is only one neuron (either 0 or 1). Because this is a binary decision, the Sigmoid Activation Function. The Sigmoid function transforms input values to a range between 0 and 1, which is useful for binary classification problems. It's often used in the output layer of a binary classification model, where the output can be interpreted as a probability. For example, if the output is close to 1, it indicates a high probability of belonging to one class, while an output close to 0 indicates a high probability of belonging to the other class. Below is a diagram representing the neural networks.

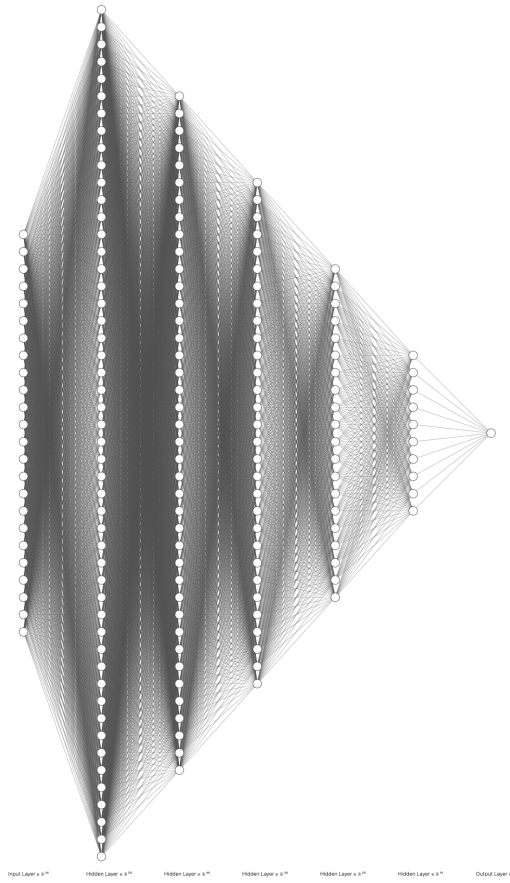

Fig. 12: Neural Network Diagram 1

This diagram clearly demonstrates that there are 24 input neurons in the input later which is fully connected to the 1st hidden layer with 50 neurons which is fully connected to the 2nd hidden layer with 40 neurons which is fully connected to the

3rd hidden layer with 30 neurons which is fully connected to the 4th hidden layer which is fully connected the 5th hidden layer which is connected to the one output neuron. The diagram below is more simplified.

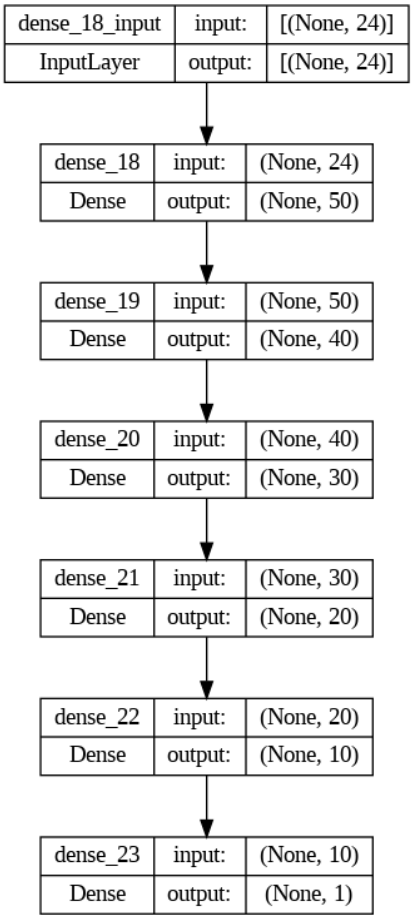


Fig. 13: Neural Network Diagram 2

Within the neural network, the model utilized the following parameters.

- Learning Rate: 0.001
- Optimizer: Adaptive Moment Estimation (Adam)
- Loss Function: Binary Cross Entropy. Measures the dissimilarity between the actual labels and the predicted probabilities of the data points being in the positive class

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

- Early Stopping Mechanism which monitor the validation loss. It has a patience of 5 epochs in which after 5 epochs if the validation decrease, it will implement and restore the previous best weights to the neurons.

Below are the diagrams that illustrate the training loss and training validation as well as the training accuracy and validation accuracy.

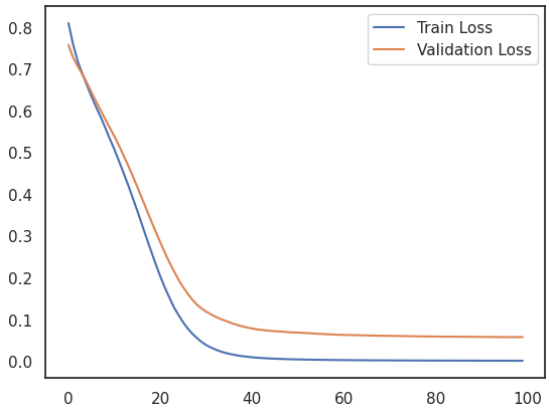


Fig. 14: Training and Validation Loss

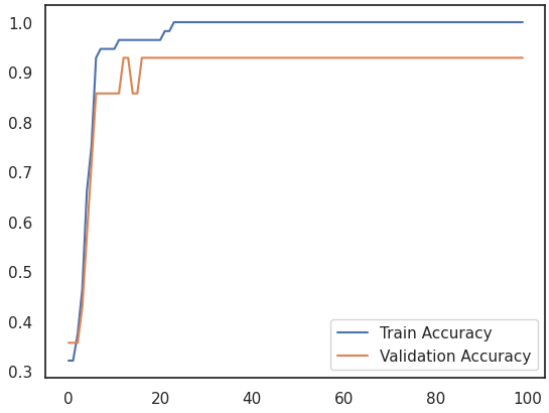


Fig. 15: Training and Validation Accuracy

Below is the performance metrics for the Neural Network and the confusion matrix as well. It should noted that the execution time is roughly two seconds.

| Measure | CKD | NotCKD |
|---|---|---|
| Accuracy | 96.67% | 96.67% |
| Precision | 100% | 91.30% |
| Recall | 94.87% | 100% |
| F1 Score | 97.87% | 96.97% |

TABLE VI: Neural Network Results

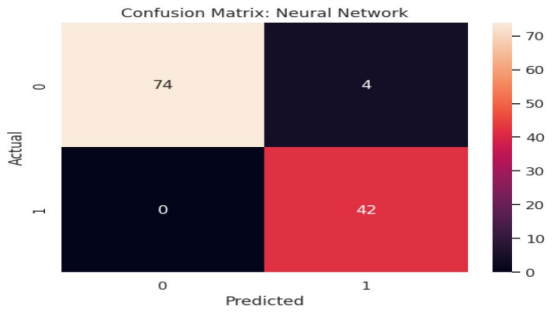


Fig. 16: Neural Network Confusion Matrix

This confusion matrix shows that the Neural Network model was able to predict individuals who had chronic kidney disease correctly with individuals who actually had chronic kidney disease 74 times. The model was also able to predict individuals who did not have chronic kidney disease correctly with individuals who did not have chronic kidney disease 42 times. It predicted an individual would have chronic kidney disease, when in fact that individual did not have chronic kidney disease zero times. It also predicted an individual would not have chronic kidney disease, when in fact that individual had chronic kidney disease four times. The Random Forest Model has a two-second execution time as well, which is slightly higher than the Decision Tree Model and Random Forest Model, but significantly lower than the Support Vector Machine.

## IV. Results and Conclusion

All four models were able to take in the training and testing data set and able to predict if an individual had chronic kidney disease or not with an accuracy of over 85% However, there is an uneven distribution of the class distribution as 250 individuals have chronic kidney disease while the remaining 150 individuals do not have chronic kidney disease. Due to the uneven distribution of the classes, the F1 Score is better. The F1 score is a measure that combines precision and recall. This is because precision and recall have their own trade-off where if you increase the precision, the recall decreases and vice versa. Thus, the F1 Score is the best metric to measure the performance along with execution time between the different models. Below is a table with the F1 Score for the Chronic Kidney Disease and Not Chronic Kidney Disease with the execution time. DT stands for Decision Tree, RF stands for Random Forest, SVM stands for Support Vector Machine, NN stands for Neural Network

| Model | F1: CKD | F1: NotCKD | Execution Time |
|---|---|---|---|
| DT | 94.14% | 89.83% | Sub-0 seconds |
| RF | 97.44% | 95.24% | Sub-0 seconds |
| SVM | 97.87% | 96.97% | 45 seconds |
| NN | 97.87% | 96.97% | 2 seconds |

TABLE VII: Model Comparison

From the table, Support Vector Machines and Neural Networks have the highest F1 score for chronic kidney disease and not chronic kidney disease. However, the execution time for the neural network model was 2 seconds and the execution time for the support vector machine was 45 seconds. Thus, the neural network model is the best in this case. The execution time for the neural network is longer compared to the decision tree and the random forest model for an increase of performance by up to 3% and 7% for chronic kidney disease and not chronic kidney disease respectively. However, in healthcare, the trade-off of an increase in performance for a slight increase in execution is worth it. Especially in cases of chronic kidney disease which can lead to heart disease and stroke which are adverse symptoms that can lead to death. Thus, taking the trade-off of higher performance to execution will be advantageous for the patient in this case.

## V. Discussion

### A. Considerations

It should be noted that a large portion of the data is technically synthetic as the majority of the missing values had to be imputed by the mean or mode if the feature was numerical and categorical respectively. By using the mean as the value imputation for the numerical value, it may not accurately represent the central tendency of the data if the feature has a skewed data set. This also doesn't account for the variability in the data. The mean imputation will assume that all the values are equally dispersed around the mean, which is not always the case. This may lead to biased results. The mean imputation also assumes that the missing values are missing completely and are random (MCAR). If the missingness is related to another feature or a pattern in the data, it can introduce artificial patterns and potentially bias as well. In mode imputation with categorical features, it also assumes that the missingness is completely at random (MCAR). If the feature is related to other features or patterns, this may also introduce bias. Mode imputation can lead to unrealistic imputed values when the most frequent category is not representative of the missing values, especially if the data values are equally split between the two categories, the mode imputation may not capture the distribution as well.

### B. Future Experimentation

A future experimentation consideration is that some of the features may not have a strong correlation to the target classification and may actually reduce the performance of the models. In future experimentation, the inputs for the models or the number of features used can be reduced to perhaps 10 features, 5 features, or even 2 features instead of all 24 different features. By doing this, it can help reduce the noise that the models will have to parse through. One of the ways to do this is to create a correlation matrix between the variables and look at which variables have the highest correlation to the target classification. Thus, when viewing a correlation matrix, one would only need to look at the row of the classification and pick the feature that has the highest value, indicating the highest correlation to the classification variables. Below is the correlation matrix from the chronic kidney disease data set.

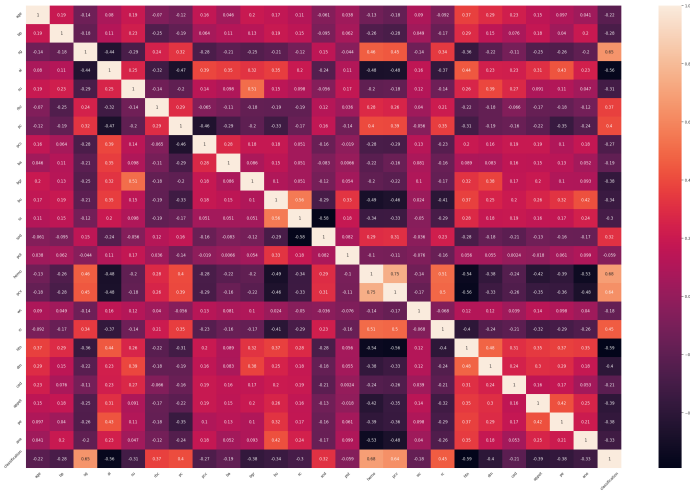

Fig. 17: Chronic Kidney Disease Correlation Matrix

The three highest features are specific gravity, hemoglobin, and pus cell volume with values of 0.65, 0.68, and 0.64 respectively. However, it is also important to note that these features have missing values. However, depending on how many missing values there are, the data set could be potentially cleaned to obtain a more accurate data set that does not have value imputation. The same four models could be run with just these variables to see if the model performance improves.

Other future experiments can be running the models on a larger data set. The chronic kidney disease data set is quite small with only 400 instances. By finding another data set for another disease could push the limitation of the models in terms of their performance metrics. Another data set that might be used is the CDC Diabetes Health Indicator Data Set [3] from the UCI Machine Learning Repository. This data set has 253,680 instances which is much larger than the 400 instances in the chronic kidney disease data set. Other models and algorithms can be incorporated as well such as the K-Nearest Neighbors (KNN) or Naive Bayes models. These models can be compared with the previously used models by their performances.

## ACKNOWLEDGMENT

This research project group thanks Professor Athanasios (Thanos) Aris Panagopoulos, the Department of Computer Science at California State University, and CSci 198: Senior Research Project

## REFERENCES

[1] Rubini,L., Soundarapandian,P., and Eswaran,P.. (2015). Chronic_Kidney_Disease. UCI Machine Learning Repository. https://doi.org/10.24432/C5G020.

[2] Jain, Apurv. "Support Vector Machine(s.v.m)Classifiers and Kernels." Medium, Medium, 25 Sept. 2020, medium.com/@apurvjain37/support-vector-machine-s-v-m-classifiers-and-kernels-9e13176c9396.

[3] CDC. U.S. Diabetes Surveillance System. Atlanta, GA: US Department of Health and Human Services, CDC; 2017. https://gis.cdc.gov/grasp/diabetes/DiabetesAtlas.html