Chronic Kidney Disease Risk Factors

Graves Sade  
School of Applied Computational Science  
Meharry Medical College Nashville, Tn. USA  
sgraves24@mmc.edu

line 1: 2nd Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address

*Abstract*— The Chronic Kidney Risk Factors project achieved an accuracy of 96.8 percent in predicting the target class variable using a support vector model. The Support Vector Machine demonstrated strong performance in determining whether a risk factor variable influences the development of chronic kidney disease.

Keywords— chronic kidney disease (CKD), Hypertension, Serum Creatine, Blood urea and glucose

# Introduction and Objectives

I wanted to pursue a project on the risk factors of chronic kidney disease due to my personal experience with End-Stage Renal Disease, which is the final stage of chronic kidney disease. I have been living with this condition since 2016. One of my goals is to raise awareness about the disease and educate as many people as possible about its effects on patients and the importance of patient support groups. Additionally, I aim to inform others about preventative measures related to this disease.

# Methods

## Dataset

The dataset used for this project was obtained from UCI Machine Learning Repository, which provides extensive, high-quality data for research purposes. The dataset can be accessed directly at <https://archive.ics.uci.edu/dataset/857/risk+factor+prediction+of+chronic+kidney+disease> and <https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease> . It includes detailed records related to both dataset description, patient electronic health records , demographic information, and the project focus target .

This project addresses a **Classification** problem, aiming to predict class variable to determine the early stage of chronic kidney disease, or numerical outcomes]. Prior to Exploratory Data Analysis (EDA), both datasets contained twenty-nine features and six hundred instances, ensuring sufficient complexity and size for robust model development and evaluation.

## Exploratory Data Analysis (EDA)

This project is funded by . (Do not delete this)

For the analysis, several preprocessing and exploratory data analysis (EDA) techniques were applied to prepare the dataset for modeling:

**Handling Missing Values**: The dataset contained one thousand and thirty-seven missing values across twenty nine features. These were handled by removing rows/columns with missing values] to ensure no loss of critical information.

**Encoding Categorical Variables**: There were four categorical variables in the dataset, including the present or not present of pus cell clumps or bacteria. These variables were encoded using method one-hot encoding, resulting in five new features for analysis. Also I used label encoding to covert categorical variables like class to nerical representation of 0 for yes and 1 for no.

**Handling Outliers**: The dataset features did not contain outliners

**Multicollinearity Analysis**: Multicollinearity was analyzed using the method of a correlation matrix. A total of eight highly correlated variables (correlation > [threshold, < 0.7) were identified, and five features were removed based on their influence on other variables and their relevance to the target variable. The multicollinearity analysis graph is shown in Figure [1], highlighting the correlated features and justifying their removal.

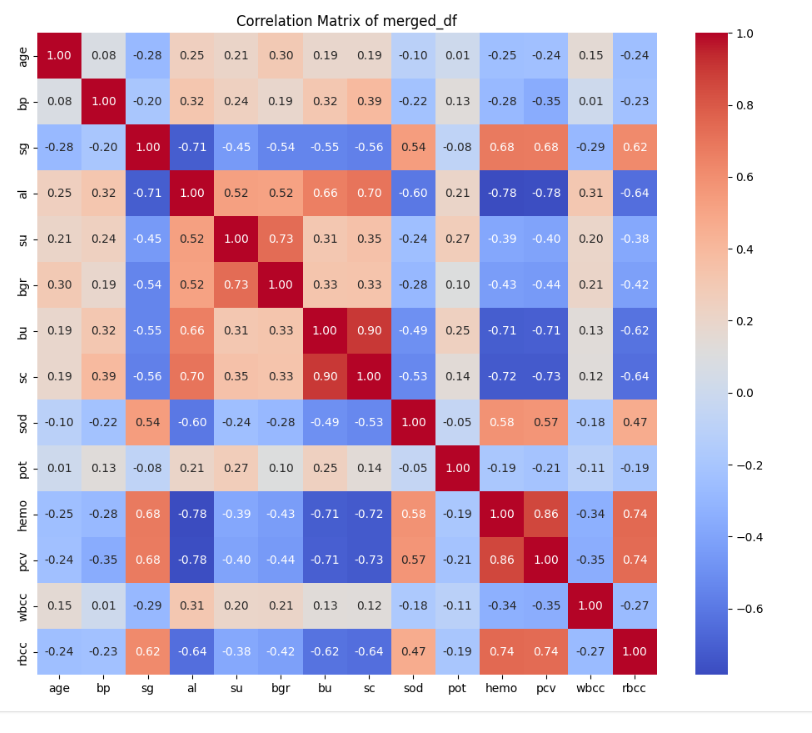


Figure [1] Correlation Matrix

**Summary of EDA Findings**: The EDA revealed key insights into the dataset, such as the high correlation between the class variable of having CKD and other essential features such as Serum creatine, blood urea nitrogen, and blood glucose protein in the blood (al) and the unbalance of the both data set due to the merging . . These findings improved the preprocessing decisions and model selection, ensuring the data was adequately prepared for robust analysis and accurate predictions.

Table [1] Final Feature:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | Feature Name | Mean and Standard Deviation | Type | Unit |
| 1 | Age | 49.563&15.51 | numerical | years |
| 2 | Blood pressure | 74.050&11.17 | Numerical | bp |
| 3 | Specific gravity | 1.019&0.005 | Numerical | sg |
| 4 | Sugar | 0.253&0.0812 | Categorical | su |
| 5 | Red Blood Cell | 0.886&0.318 | categorical | Rbc |
| 6 | Pus Cell | 0.816&0.385 | categorical | pc |
| 7 | Pus Cell clumps | 0.888&0.285 | categorical | pcc |
| 8 | Bacteria | 0.075&0.265 | categorical | ba |
| 9 | Sodium | 138.84&7.486 | numerical |  |
| 10 | Blood Urea | 52.575&47.395 | numerical | bu |
| 11 | Potassium | 4.634&3.476 | numerical | pot |
| 12 | White Blood Cell Count | 8475.9&3126.88 | numerical | wbcc |
| 13 | Red Blood Cell Count | 4.897&1.019 | numerical | rbcc |
| 14 | Hypotension | 0.215&0.412 | Categorical | htn |
| 15 | Diabetes Mellitus | 0.177&0.383 | Categorical | dm |
| 16 | Pedal Edema | 0.126&0.326 | Categorical | pe |
| 17 | Coronary Artery Disease | * 1. &0.255 | Categorical | cad |
| 18 | Appetite | 0.120 & 0.326 | Categorical | appet |
| 19 | Anemia | 0.101 & 0.302 | Categorical | ane |
| 20 | Class | 0.727 & 0.446 | Categorical | class |

## Model Selection

We have implemented the following models to train and validate our data: (From this one just remove the models you never used)

Logistic Regression: Logistic regression served as a baseline model due to its simplicity and interpretability. It models the probability of readmission as a linear combination of the features. The model’s coefficients are straightforward to interpret, providing insights into the direction and strength of each feature’s impact. However, it is limited in its ability to capture non-linear relationships in the data.

Random Forest: Random forest is an ensemble method that combines multiple decision trees, leveraging different subsets of data to enhance robustness and reduce overfitting. It performs well with non-linear data and provides feature importance metrics, making it useful for identifying key predictors. However, it can be computationally intensive and prone to overfitting if not carefully tuned.

Support Vector Machine (SVM): SVM is a powerful algorithm, particularly effective for high-dimensional datasets. It aims to find the optimal hyperplane that maximally separates classes, enabling it to model complex decision boundaries. While it performs well when the classes are clearly separable, SVMs can be sensitive to parameter tuning and computationally expensive for large datasets.

Decision Tree: A decision tree is a simple, interpretable model that splits the data recursively based on feature thresholds. Each branch represents a decision rule, making it intuitive for users to understand. However, decision trees are prone to overfitting, especially in noisy datasets, and often require pruning or parameter tuning to improve generalizability.

AdaBoost: Adaptive Boosting (AdaBoost) sequentially builds weak learners, typically decision trees, each focusing on correcting errors from the previous iteration. It excels at improving accuracy, particularly for hard-to-classify instances. However, it is sensitive to noise and outliers, which can affect its performance.

Gradient Boosting: Gradient Boosting builds models sequentially, with each tree correcting the residuals of the previous tree using gradient descent. This approach effectively captures complex relationships and often achieves high predictive accuracy with proper tuning. The trade-off is its computational expense and susceptibility to overfitting if not regularized.

XGBoost: XGBoost is an optimized version of gradient boosting designed for speed and efficiency. It incorporates regularization terms, making it less prone to overfitting and suitable for large datasets. While it offers high predictive power, it can be computationally intensive and requires careful parameter tuning.

LightGBM: LightGBM is a gradient-boosting framework optimized for efficiency and scalability. Its leaf-wise tree growth strategy reduces training time without sacrificing accuracy, making it ideal for large, high-dimensional datasets. However, it may overfit on smaller datasets and requires careful handling of categorical features.

CatBoost: CatBoost is a gradient-boosting algorithm designed to natively handle categorical data, simplifying preprocessing. It offers strong performance and faster training times with less hyperparameter tuning compared to other boosting models. While it may not excel with continuous-only data, it is generally robust and efficient.

Deep Neural Network (DNN): Finally, we used a Deep Neural Network (DNN) to leverage its ability to model complex, non-linear relationships in the data. DNNs excel in predictive accuracy when trained with sufficient data and optimized effectively, providing a robust solution for our analysis.

## Train Test Split

The dataset was divided into training and testing sets using a stratified split and grid search CV approach. This split ensured that the models had sufficient data to learn patterns while maintaining a robust evaluation set. The class distribution was carefully maintained in both sets to reflect the original dataset composition.

Within the training set, we performed five -fold cross-validation to identify the best hyperparameters for each model, ensuring optimal performance and reducing the risk of overfitting. This approach allowed us to fine-tune the models effectively while preserving the integrity of the test set for unbiased evaluation.

To evaluate the performance of the Deep Neural Network (DNN), the dataset was divided into training, validation, and testing sets using a stratified approach. A total of 70% of the data was allocated for training, 15% for validation, and the remaining 15% for testing. The training set was used to train the model, while the validation set was employed for hyperparameter tuning and monitoring performance during training to prevent overfitting. We further experimented with separately tuned models, incorporating regularization techniques such as L2 regularization and dropout, to optimize generalization performance. The final evaluation was conducted on the testing set using the best-tuned model, ensuring an unbiased assessment of its predictive accuracy.

## Model Evaluation

For the classification models, evaluation metrics included sensitivity (recall), specificity, accuracy, precision, and F1-score to provide a comprehensive assessment of performance. Sensitivity and specificity measured the model’s ability to correctly identify positive and negative cases, respectively, while accuracy provided an overall performance indicator. Precision assessed the proportion of true positive predictions among all positive predictions, and the F1-score balanced precision and recall evaluating the model’s robustness in handling imbalanced data.

For regression models, evaluation focused on Mean Squared Error (MSE), which quantified the average squared difference between predicted and actual values, and R2, which explained the proportion of variance in the target variable accounted for by the model. These metrics ensured a detailed understanding of the models’ predictive capabilities across tasks.

# Results

## Conventional Model Results

The performance of the conventional models was evaluated using metrics such as accuracy, sensitivity, specificity, precision, recall, and F1-score, as summarized in **Table [2]**. Among all the models, the highest accuracy of 0.97\% was achieved by the support vector machine . This model also demonstrated strong performance across other metrics, making it the most effective conventional approach for the classification task.

The best-performing was optimized using the following hyperparameters:  **Table [3]**. These hyperparameters were identified through cross-validation on the training set, ensuring a balance between generalization and performance. This configuration allowed the model to effectively capture patterns in the data while minimizing overfitting.

**Confusion Matrix for Test Set**: The confusion matrices for the top three models are illustrated in Figures [Y1], [Y2], and [Y3], providing a detailed breakdown of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These matrices offer insights into the classification performance of each model:

**Model 1 SVM**: As shown in Figure [Y1], Model 1 achieved the highest accuracy of 96.88%, with [TP: 23, TN: 8, FP: 51 FN: 0]. This performance demonstrates its strength in correctly identifying both positive and negative cases, making it the most robust among the evaluated models.

**Model 2 /Logistic Regression**: Shown in Figure [Y2], Model 2 achieved an accuracy of 93.7%, with [TP: 23, TN: 7, FP: 2, FN: 0]. While its performance is slightly lower than Model 1, it still performed well, particularly in minimizing false positives.

.

Model 3/ ADABoost: Illustrated in Figure [Y3], Model 3 achieved an accuracy of 91.58%, with [TP: 65, TN: 22, FP: 4, FN: 1].  Despite being competitive, it had a higher rate of misclassifications compared to the other two models, indicating potential areas for improvement, such as handling class imbalance.

These visualizations highlight the comparative strengths and weaknesses of the top models, guiding the selection of the most appropriate model for the classification task.

## Deep Learning Results

The performance of the Deep Neural Network (DNN) was evaluated using metrics such as accuracy, sensitivity, specificity, precision, recall, and F1-score, as summarized in Table [4].

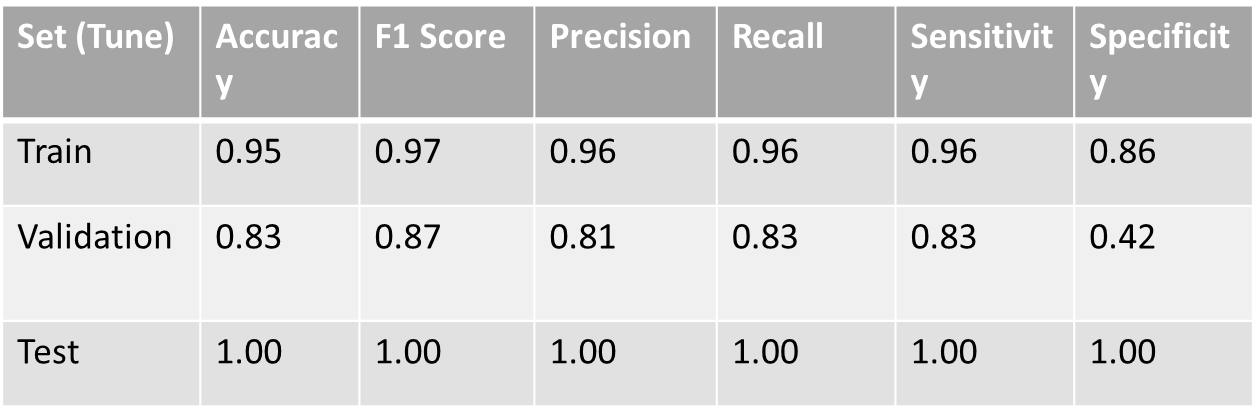
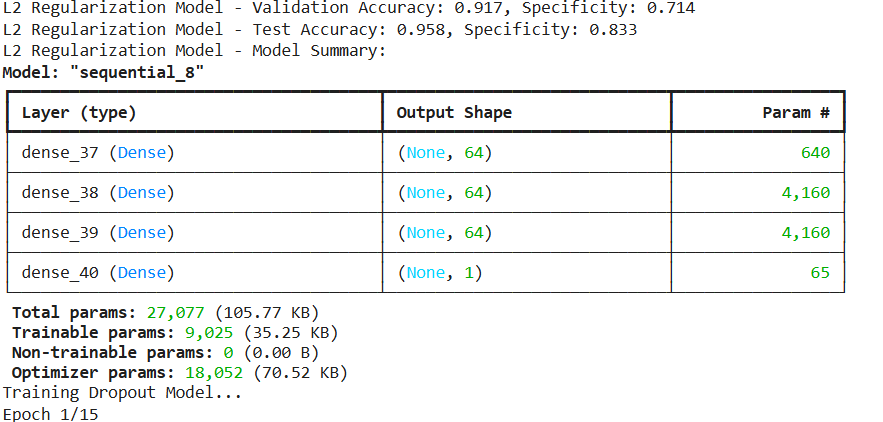


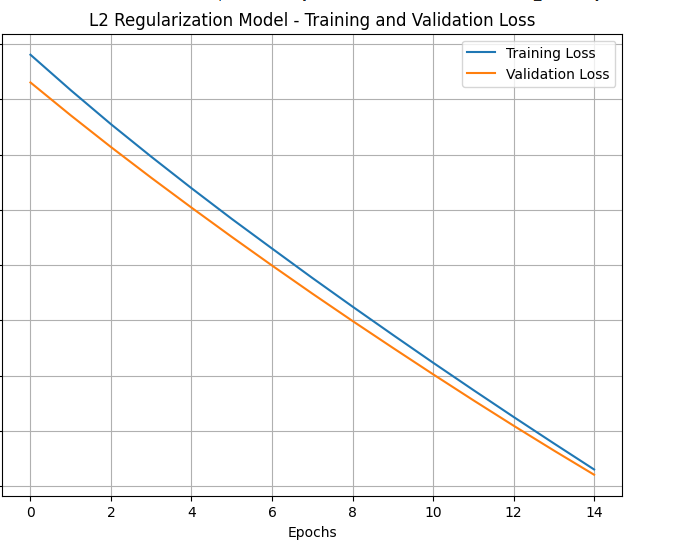
Table [4] Deep Neural Network

Figure[ 4] DNN Regularization Architectures



Among the evaluated models, the architecture attached in Figure [4] achieving the highest accuracy of 0.92% on the test 91.7/validation0.83 set consisted of an input layer with 100 neurons, followed by a Dropout layer 20 rate % to prevent overfitting. The first hidden layer had first hidden 64 neurons neurons with ReLU activation, followed by another Dropout layer dropout 20 rate %. The second hidden layer included second hidden 64 neurons] neurons with ReLU activation, and the final output layer consisted of output 1 neurons] neurons with sigmoid activation for classification. The total parameters of the model were 27,077, with 9,025 trainable parameters. The remaining parameters belong to the optimizer configuration This architecture effectively balanced complexity and generalization.

The model’s performance over the training epochs is depicted in Figure [5], which illustrates the training and validation loss trends. The training loss steadily



decreased, indicating effective learning, while the validation loss demonstrated stability, suggesting minimal overfitting. This performance highlights the robustness of the model architecture and the benefits of hyperparameter tuning and regularization strategies such as L2 regularization and dropout.

**Confusion Matrix for Test Set**: The confusion matrix for the Deep Neural Network (DNN) model is illustrated in Figure [6], providing a detailed breakdown of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

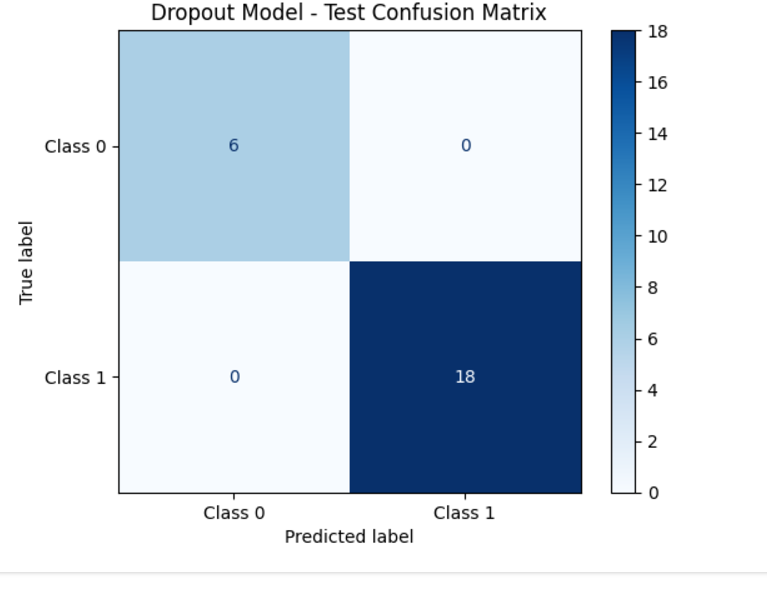


Figure [6] Deep Neural Network Model

# Discussion

Taking the machine learning class has been a truly eye-opening experience. I’ve learned about various statistical models and how to fine-tune them using hyperparameter optimization. The class also introduced me to the concept of multicollinearity and its impact on model performance. I explored techniques like one-hot encoding and deep neural networks, which gave me a deeper understanding of how to work with complex datasets. Even though I had no background in data analysis before this, the course has taught me a lot and equipped me with some really useful skills.

##### Acknowledgment *(Heading 5)*

Any Acknowledgement.

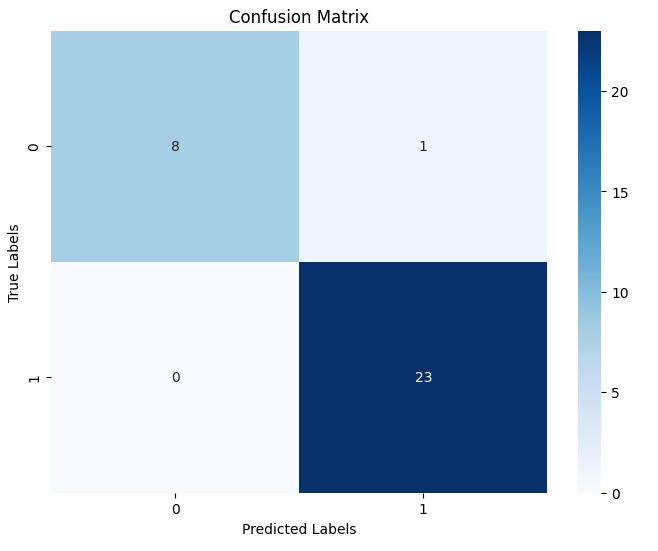
##### References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

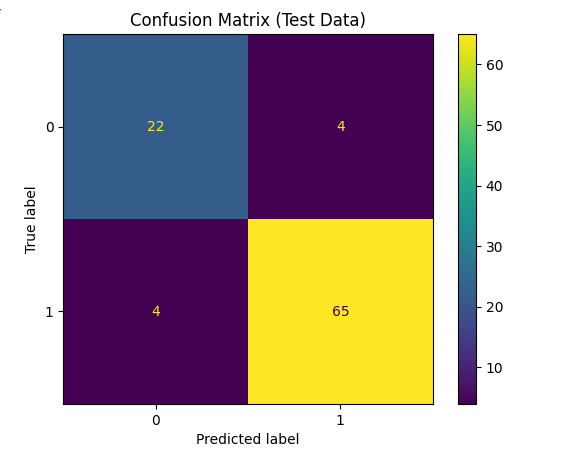
Table 2 : Conventional Model Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Models | Accuracy | F1 Score | Precision | Recall | sensitivity | specificity |
| Logistics Regression | 0.84 | CKD:0.62  NCKD:.98 | CKD:1.00  NCKD:.82 | CKD:0.44  NCKD:1.0 | CKD:0.44  NCKD:1.0 | 0.22 |
| SVM | 0.96 | CKD:0.94  NCKD:.98 | CKD:1.00  NCKD:.96 | CKD:0.89  NCKD:1.0 | CKD:0.94  NCKD:.98 | 0.88 |
| ADA Boost | 0.92 | CKD:0.89  NCKD:.92 | CKD:0.85  NCKD:.94 | CKD:0.85  NCKD:.92 | CKD:0.85  NCKD:.92 | 0.85 |

**Figures [Y1] SVM**



**Figures [Y3] ADA Boost**



**Figures [Y2]** **Logistic Regression**

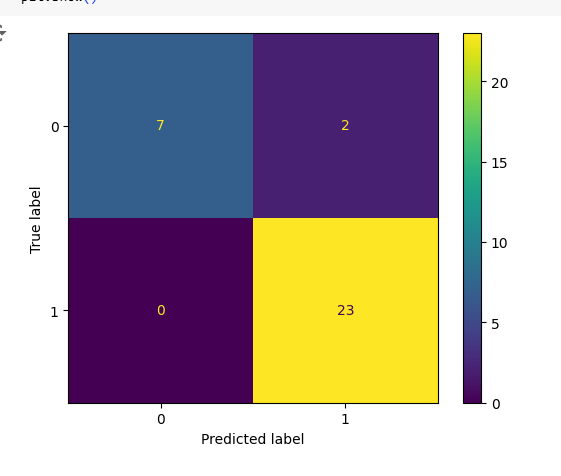


Table [3] Best Hyperparameters For Conventional Models

|  |  |
| --- | --- |
| Models | Best Hyperparameters |
| Logistic Regression |  |
| SVM |  |
| ADA Boost | Nestimators:200  Learning\_rate:1.0  Algorithm:samme |

Table 3: DNN results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Set | Accuracy | F1 Score | Precision | Recall | Sensitivity | Specificity |
| Train | X.XX | X.XX | X.XX | X.XX | X.XX | X.XX |
| Validation | X.XX | X.XX | X.XX | X.XX | X.XX | X.XX |
| Test | X.XX | X.XX | X.XX | X.XX | X.XX | X.XX |