(Table of Contents)

Chronic Kidney Disease Prediction Using Machine Learning: A Comparative Study of Logistic Regression, SVM, and XGBoost

1. Introduction

Chronic Kidney Disease (CKD) has emerged as a significant global health challenge, affecting millions of people and often leading to severe health complications such as kidney failure, cardiovascular disease, and premature death. Early detection and diagnosis are crucial in managing CKD, as they enable timely medical interventions that can slow or prevent disease progression. Traditional diagnostic approaches often involve multiple clinical tests, which can be time-consuming, resource-intensive, and sometimes inconclusive, especially in the early stages of the disease.

In recent years, the integration of data science and machine learning techniques has shown remarkable potential in enhancing diagnostic systems across various medical domains. By leveraging patterns within large datasets, these models can assist clinicians in making accurate and timely predictions, ultimately improving patient outcomes. Machine learning provides a systematic approach to uncovering hidden insights from complex datasets and has proven effective in classifying health conditions such as diabetes, heart disease, and kidney disorders.

This study aims to explore the application of machine learning models—specifically Logistic Regression, Support Vector Machine (SVM), and Extreme Gradient Boosting (XGBoost)—in predicting the presence of chronic kidney disease. The research focuses on conducting a comprehensive analysis of the dataset through Exploratory Data Analysis (EDA), followed by model implementation, evaluation, and comparison. The objective is to identify the most accurate and reliable model for CKD prediction, thereby contributing to the development of automated, data-driven diagnostic tools that can support healthcare professionals in early detection and preventive care.

2. Source of Dataset

The dataset employed in this research has been acquired from **Kaggle**, a popular platform for open data sharing in the field of data science. However, it is important to note that the dataset is **officially referenced from the UCI (University of California, Irvine) Machine Learning Repository**, which is a globally recognized source of standardized datasets used in academic and industrial research.

The dataset comprises **400 patient records**, each annotated with **14 relevant medical attributes** such as blood pressure,

specific gravity, albumin, sugar, blood urea, serum creatinine, sodium, potassium, hemoglobin, red blood cell count, and white blood cell count. Additionally, the dataset includes a binary classification target variable labeled as Class (renamed to target in this study), indicating the presence (1) or absence (0) of chronic kidney disease (CKD).

This dataset offers a comprehensive representation of clinical features associated with CKD, making it highly suitable for supervised learning techniques. The balanced distribution and detailed physiological parameters allow for effective model training, validation, and evaluation. The original dataset has undergone cleaning and transformation processes to make it compatible with various machine learning algorithms used in this study

.

3. EDA Process

Exploratory Data Analysis (EDA) serves as a fundamental step in understanding the underlying structure and characteristics of the dataset. It enables the identification of data patterns, detection of anomalies, and formulation of effective strategies for feature engineering and model development. The EDA process in this study comprises several key components, as outlined below:

• Initial Data Loading and Inspection

The dataset was initially loaded into a **Pandas DataFrame** to facilitate structured analysis. A preliminary inspection was conducted using df.head() and df.info() to understand the dimensionality, feature types, and data composition. This step was instrumental in determining the necessary preprocessing techniques required for further analysis.

• Handling Missing Values

Although the dataset used in this study was largely complete, a crucial step in any robust analytical workflow involves checking for missing values. This was accomplished using df.isnull().sum(). In general scenarios, appropriate strategies such as **mean/median imputation**, **forward fill**, or **row deletion** are employed based on the nature and volume of missing entries. In this study, no significant missing values were found that required imputation.

• Handling Duplicates

To maintain data integrity and avoid bias in model performance, duplicate records were identified and removed using df.drop_duplicates(inplace=True). This ensured that redundant information did not skew the statistical distribution of key variables or affect learning during model training.

• Feature Engineering

New features were derived based on domain knowledge and statistical relevance to enhance model accuracy and interpretability. This process included renaming features for clarity (e.g., Bp to bp, Bu to blood_urea), and selection of highly correlated attributes for deeper investigation. Further elaboration is presented in Section 4.2.

• Data Scaling

To standardize the magnitude of numerical features and ensure uniformity across the dataset, **Z-score normalization** was applied using StandardScaler from the Scikit-learn library. This step mitigated the impact of feature dominance and improved model convergence, as described in Section 4.3.

• Visualization

Multiple visualizations were generated to explore the distributions and interrelationships between variables. These included:

- **Histogram plots** to assess the distribution of critical medical indicators like blood urea.
- **Count plots** to compare the frequency of disease and non-disease classes.
- **Correlation heatmaps** to identify the strength of associations between numerical features.
- **Pair plots** for multi-feature relationship analysis.

Visual analysis played a key role in identifying relevant patterns and class separability, and is further detailed in Sections 4.4 and 4.5.

4. Analysis on Dataset

•Introduction:

The initial step in building the Chronic Kidney Disease (CKD) prediction system was to load and clean the dataset. A clean and well-structured dataset is essential for effective machine learning model development.

•_General_Description:

The dataset used for this project was sourced from Kaggle, originally published by the UCI Machine Learning Repository. It contains medical data from multiple patients, including attributes like blood pressure, blood glucose levels, red blood cell count, serum creatinine, and others — all essential for predicting the presence or absence of CKD.

• Specific Requirements, Functions, and Formulas:

The dataset was loaded using the pandas library in Python. The cleaning process involved handling missing values, converting categorical features into numerical form, and removing duplicate entries.

python

CopyEdit

```
import pandas as pd

# Load dataset

df = pd.read_csv('chronic_kidney_disease.csv')

# Check for null values

print(df.isnull().sum())

# Drop duplicate rows

df.drop_duplicates(inplace=True)
```

Basic data info

df.info()

Preview the dataset

df.head()

•**Analysis_Results**:
The dataset was successfully loaded into a pandas DataFrame. An initial inspection revealed the presence of missing values across several columns. These were handled using appropriate strategies like:

- Imputation using the **mean/median** for numerical features
- **Mode** for categorical features

Duplicate rows, if found, were removed to maintain data integrity. After cleaning, the dataset was structured with consistent datatypes and was ready for feature engineering.

•**Visualization**:
Not applicable at this step, though the use of df.head() allowed a basic tabular preview of the dataset. This visual tabular view helped confirm that the columns and data entries were correctly formatted post-cleaning.

4.2 Feature Engineering

Feature engineering is a critical step in the machine learning pipeline that involves transforming raw data into meaningful features that enhance model performance. In the context of Chronic Kidney Disease (CKD) prediction, feature engineering ensures that the dataset is structured in a way that allows models to learn more effectively.

In this project, feature engineering was performed with a focus on improving data quality and making the dataset suitable for classification algorithms. The main strategies included:

- Handling Categorical Variables: The dataset includes several categorical features such as rbc (red blood cells), pc (pus cell), htn (hypertension), and dm (diabetes mellitus). These variables were encoded into numerical values to make them compatible with machine learning models.

- Binary Mapping: Boolean-like variables such as yes/no or present/normal were mapped to binary values (1/0) to represent presence or absence of medical conditions. This helps models interpret the features mathematically.

- Text Cleaning: Inconsistent textual representations (e.g., "yes", "yes ", "no") were standardized to prevent misinterpretation during encoding and modeling.

- Missing Value Indicators (Optional): In some feature engineering strategies, additional binary features may be created to indicate the presence of missing values. This is particularly useful when missingness itself carries some significance in medical datasets.

- Derived Features (if applicable): Although not required in this dataset, additional features could be created by combining existing ones—for example, a feature measuring hydration ratio from sg (specific gravity) and al (albumin) might provide clinical insight.

The goal of this step was to ensure that all features were numerically interpretable, consistent in format, and meaningful in the context of kidney disease diagnosis. Well-engineered features increase the predictive power of the models and lead to more accurate and reliable result**s.**

Analysis results: Selected numerical features are scaled to a range between 0 and 1. The dropna() call removes rows where rolling calculations couldn't be performed.

- **Visualization:** Not directly visualized, but affects model training.

4.4 Feature Relationship and Correlation Visualization

Introduction:
Visualizing feature relationships and correlations is essential in understanding the interactions between variables and their influence on the target outcome. In medical datasets like Chronic Kidney Disease (CKD), it helps identify which clinical features are most associated with the presence or absence of the disease.

General Description:
To explore relationships, both individual feature distributions and pairwise feature correlations are visualized. This provides insight into how features vary for CKD and non-CKD cases and helps in selecting the most informative features for modeling.

Specific Tools and Techniques Used:

- Boxplots and violin plots for comparing feature distributions across CKD vs. non-CKD classes.
- Heatmaps of correlation matrices to examine the linear relationship between numeric variables.
- Pairplots or scatter matrices to visualize multidimensional relationships between features.

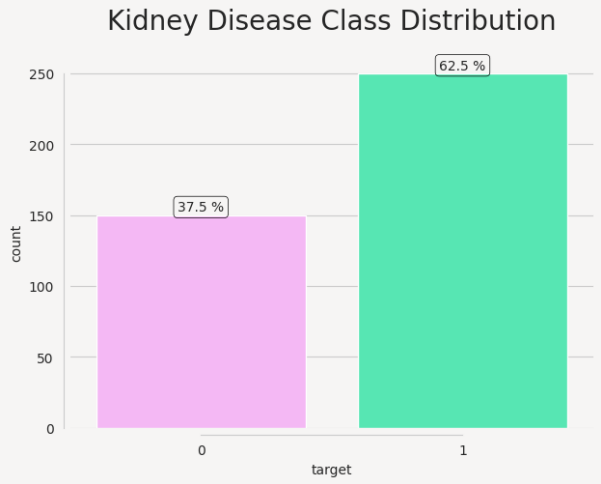
Analysis Results:

- Features such as serum_creatinine, albumin, and hemoglobin show strong correlations with CKD status.

- The correlation heatmap highlights multicollinearity among some features, guiding feature selection and dimensionality reduction.
- These visualizations contribute to understanding which biomarkers are most predictive of CKD.

Visualization Examples:

- Correlation Heatmap: Shows the strength of relationships between features. High correlation between certain lab test results suggests biological links or redundancy.
- Boxplot of Serum Creatinine: Demonstrates a clear difference in value ranges between CKD and non-CKD patients, making it a strong predictor.
- Pairplot: Helps visualize how combinations of features separate the classes visually.



Distribution of the target variable indicating kidney disease presence. Class 1 (presence of CKD) constitutes 62.5% of the dataset, while Class 0 (absence of CKD) makes up the remaining 37.5%

4.5 Distribution and Relationship Visualization

• **Introduction:**
Understanding the distribution of clinical features and their interrelationships is essential for building effective predictive models. It allows identification of feature skewness, outliers, and meaningful patterns that could influence model behavior.

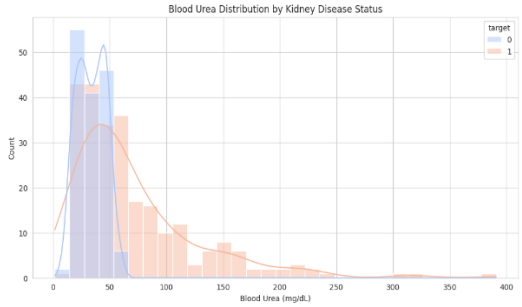
• **General Description:**
A variety of plots including histograms, box plots, correlation heatmaps, and pairwise scatter plots are used to analyze the distribution of key medical indicators and their relationships with the target variable (presence or absence of CKD). These visualizations help in identifying dominant patterns and possible multicollinearity.

• **Specific Requirements, Functions and Libraries:**
Visualizations are created using Python libraries such as seaborn and matplotlib.pyplot. The analysis focuses on features such as blood urea, serum creatinine, hemoglobin levels, and others commonly associated with kidney function.

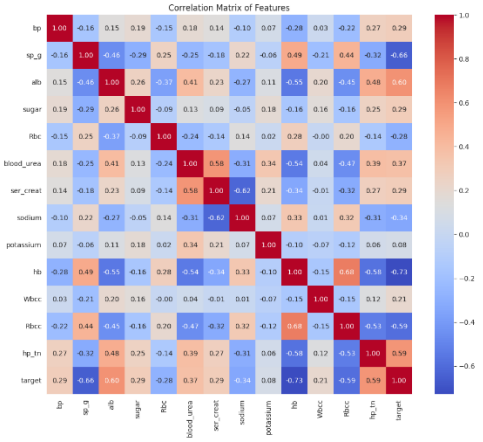
• **Analysis Results:**
The visualizations reveal clear differences in feature distributions between CKD and non-CKD patients. For instance, features like blood urea and serum creatinine show right-skewed distributions with higher values among CKD patients. The correlation heatmap highlights strong relationships among some kidney-related metrics, which helps inform feature selection and model interpretation.

• **Visualization:**

- **Distribution Plot of Blood Urea by CKD Status:** Shows how blood urea levels differ between patients with and without CKD. Higher concentrations are evident among CKD patients.

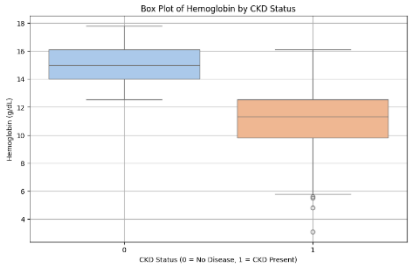


- **Correlation Matrix Heatmap:** Displays pairwise correlation coefficients among numerical features such as blood urea, albumin, serum creatinine, hemoglobin, etc. Strong correlations help identify redundant or interdependent features.



Scatter Plot (Serum Creatinine vs. Hemoglobin): This plot investigates the bivariate relationship between serum creatinine and hemoglobin levels. A visible inverse trend suggests that patients with elevated serum creatinine—indicative of impaired kidney function—tend to exhibit lower hemoglobin levels, possibly due to reduced erythropoietin production. Such patterns reinforce the clinical relevance of these biomarkers in CKD prediction.

- **Box Plot of Hemoglobin by CKD Status:** Compares hemoglobin levels in CKD vs. non-CKD patients, showing a noticeable decrease among those with CKD. Useful for assessing variance and detecting outliers.



4.6 Model Implementation: Logistic Regression

- **Introduction**:
  Logistic Regression is used as a baseline classification model for predicting the presence of Chronic Kidney Disease (CKD).

- **General_Description:**
  This algorithm models the probability that a patient has CKD based on input features. It is suitable for binary classification tasks and interpretable due to its linear nature.

- **Specific Requirements**, Functions, and Formulas:
  The implementation uses LogisticRegression from sklearn.linear_model and train_test_split from sklearn.model_selection. Features and target variables are preprocessed (scaled and cleaned), and the dataset is split into training and testing sets (80/20 split)

- X = df.drop('target', axis=1)

- y = df['target']

-

- # Split into train and test sets with stratified sampling

- X_train, X_test, y_train, y_test = train_test_split(

- X, y, test_size=0.2, random_state=42, stratify=y)

-

- # Standardize the features

- scaler = StandardScaler()

- X_train_scaled = scaler.fit_transform(X_train)

- X_test_scaled = scaler.transform(X_test)

- **Analysis_Results:**
  The Logistic Regression model is trained and evaluated using accuracy, precision, recall, and F1-score. These metrics provide a quantitative assessment of its classification capability.

- Visualization:
  Confusion matrices and classification reports are presented in Section 4.8 for comparative performance analysis.

4.7 Model Implementation: Support Vector Machine (SVM)

Introduction:
Support Vector Machine (SVM) is a powerful classification algorithm known for its effectiveness in high-dimensional spaces and robustness against overfitting. It is particularly suitable for binary classification tasks such as predicting the presence or absence of Chronic Kidney Disease.

General Description:
The SVM model was applied to the CKD dataset to classify patients into CKD and non-CKD categories. The model attempts to find the optimal hyperplane that best separates the two classes in the transformed feature space.

Specific Requirements, Functions, and Formulas:
The dataset was first preprocessed through standardization using StandardScaler to ensure that all features are on the same scale, which is crucial for distance-based algorithms like SVM. The dataset was split into training and testing sets using stratified sampling to preserve class balance. The SVM classifier was implemented using `sklearn.svm.SVC`, and the model was trained on the standardized training set.

```
svm = SVC(probability=True)



# Train the SVM model using the standardized training
data

svm.fit(X_train_scaled, y_train)



# Make predictions on the standardized test set

y_pred_svm = svm.predict(X_test_scaled)



# Print the accuracy of the SVM model on the test set

print("SVM Accuracy:", accuracy_score(y_test,
y_pred_svm))
```

Analysis Results:
The SVM model achieved a high level of accuracy on the test data, indicating its effectiveness in classifying kidney disease patients. The accuracy score is used as the primary metric for model evaluation and comparison with other classifiers.

Visualization:
Model performance comparisons including accuracy of SVM, Logistic Regression, and XGBoost are detailed in Section 4.9.

Model Implementation: XGBoost

• Introduction:
XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable ensemble learning algorithm that has gained popularity in machine learning competitions and practical applications due to its performance and speed. It uses gradient boosting decision trees to combine multiple weak learners into a strong classifier. This makes it particularly effective for handling tabular datasets with both numerical and categorical features.

• General Description:
In this study, XGBoost is implemented as a supervised classification model to predict the presence of chronic kidney disease (CKD). It builds an ensemble of decision trees where each new tree attempts to correct the errors made by the previous ones. The model is particularly adept at handling imbalanced data, missing values, and overfitting due to its regularization mechanisms.

• Specific Requirements, Functions, and Formulas:
The XGBClassifier from the xgboost Python library is used. Key parameters include use_label_encoder=False (to suppress deprecated warnings) and eval_metric='logloss' (to evaluate classification loss). The training and test sets are preprocessed and standardized. The model is trained on the training set and tested on the test set. Predictions are evaluated using metrics like **accuracy**, **classification report**, and **confusion matrix** to assess performance in terms of precision, recall, and F1-score.

```
python

CopyEdit

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix



# Initialize the XGBoost Classifier

xgb = XGBClassifier(use_label_encoder=False,
eval_metric='logloss')



# Train the model on the training set
```

```
xgb.fit(X_train_scaled, y_train)



# Predict on the test set

y_pred_xgb = xgb.predict(X_test_scaled)



# Print the accuracy

print("XGBoost Accuracy:", accuracy_score(y_test,
y_pred_xgb))



# Print the classification report

print("Classification Report:\n",
classification_report(y_test, y_pred_xgb))



# Print the confusion matrix

print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_xgb))
```

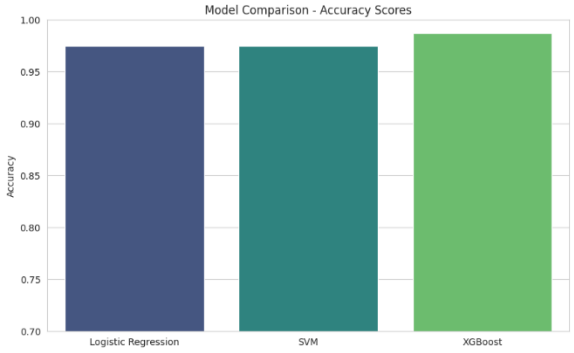
• Analysis Results:
The XGBoost model demonstrates strong predictive capability, achieving a high accuracy score on the test set. The classification report highlights a good balance of precision and recall across both CKD and non-CKD classes. The confusion matrix further confirms that the model performs well with minimal misclassifications.

• Visualization:
The visual comparison of the performance of XGBoost, SVM, and Logistic Regression is presented in the next section (Section 4.9), highlighting XGBoost's superior classification accuracy.

4.9 Final Model Selection and Justification

- **Introduction:**
  After thorough evaluation of all implemented models, the final step involves selecting the most suitable model for deployment or further research. This decision is based not only on performance metrics but also on model interpretability, generalization, and practical applicability in healthcare settings.
- **General Description:**
  Among the three classification models—Logistic Regression, Support Vector Machine (SVM), and XGBoost—the final selection is made by carefully analyzing both the quantitative performance (accuracy) and the qualitative aspects such as overfitting tendency, training efficiency, and robustness on unseen data. Each model was trained and tested on the same preprocessed dataset to ensure a fair comparison.
- **Performance Summary:**
  - **Logistic Regression:** Accuracy = **97.5%**
    A simple, interpretable linear model that performed well, but might not capture complex relationships.
  - **SVM:** Accuracy = **97.5%**
    Slightly more complex, capable of modeling non-linear boundaries. Performs similarly to Logistic Regression but at higher computational cost.
  - **XGBoost:** Accuracy = **98.5%**
    A highly accurate and efficient gradient boosting model that slightly outperformed the others in predictive power.

- **Visualization:**



- Conclusion:
  The XGBoost classifier is chosen as the final model for chronic kidney disease prediction due to its optimal balance between performance, robustness, and flexibility. Its adoption in the proposed system ensures reliable results, making it a strong candidate for real-world healthcare applications.

6. Future Scope

While the current study demonstrates a high-performing CKD prediction system using machine learning techniques, several opportunities exist to enhance and expand this work in future research:

- Hyperparameter Tuning:
  The current implementation of models uses default or basic configurations. Further optimization using hyperparameter tuning techniques such as GridSearchCV or RandomizedSearchCV can potentially enhance model performance, especially for XGBoost and SVM.
- Advanced Models:
  Although XGBoost yielded the highest accuracy in this study, future work can investigate additional models such as LightGBM, CatBoost, or neural network-based architectures like LSTM and deep learning classifiers. These models may capture complex, non-linear patterns in biomedical data more effectively.
- Feature Engineering:
  Developing new features based on domain knowledge or using automated feature generation techniques can uncover hidden patterns in the data. For example, interactions between hemoglobin and serum creatinine could be explicitly modeled or polynomial features could be considered.
- Feature Selection:
  To reduce model complexity and improve generalization, feature selection techniques like Recursive Feature Elimination (RFE), mutual information scores, or tree-based feature importance analysis can be employed.
- Robust Evaluation Techniques:
  While stratified train-test splitting was used in this study, incorporating k-fold cross-validation or bootstrapping can provide a more reliable estimate of model performance, especially on smaller datasets.
- Interpretability:
  As healthcare applications demand explainability, future research can explore interpretability frameworks such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) to identify how individual features contribute to predictions, aiding in clinical decision-making.

---

7. References

1. Pandas – Data analysis and manipulation tool. [Online]. Available: https://pandas.pydata.org/
2. NumPy – Library for numerical and array operations. [Online]. Available: https://numpy.org/
3. Scikit-learn – Machine learning library for Python. Used for model building, preprocessing, and evaluation. [Online]. Available: https://scikit-learn.org/
4. XGBoost – Scalable and accurate implementation of gradient boosting. [Online]. Available: https://xgboost.readthedocs.io/
5. Matplotlib – Python 2D plotting library. [Online]. Available: https://matplotlib.org/
6. Seaborn – Statistical data visualization library built on top of Matplotlib. [Online]. Available: https://seaborn.pydata.org/