Data Collection and Preprocessing Phase

Date	11 June 2024
Team ID	SWTID1749620488
Project Title	Chronic Kidney Disease Detection Using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Section	**Description**
Data Overview	The dataset contains ~400 patient records with 25 columns, including demographic, clinical, and target variables. Initial checks show mixed data types and some inconsistent formatting (e.g., tab characters in labels).statistics, dimensions, and structure of the data.
Univariate Analysis	Descriptive stats like mean, median, and value counts were calculated for each feature. Categorical features like diabetesmellitus and appetite were explored with count plots.
Bivariate Analysis	Box plots and violin plots were used to compare distributions of numerical features like blood_urea, hemoglobin by class (ckd vs not-ckd). Correlation matrix and heatmap were plotted for continuous variables.
Multivariate Analysis	Pairplots and 3D scatter plots showed interaction between features like age, blood_glucose_random, and hemoglobin. PairGrid was used to explore multi-variable relationships by class..
Outliers and Anomalies	Visual inspection via box plots helped identify potential outliers in features like serum_creatinine and blood_glucose_random. However, no values were dropped — instead, missing values were imputed.
Data Preprocessing Code Screenshots	

Loading Data	pd.read_csv() was used to load the chronickidneydisease_dataset.csv dataset. Column names were renamed for clarity.
Handling Missing Data	Missing values were filled using: mean for continuous variables, mode for categorical ones. Used isnull().sum() to verify completeness.
Data Transformation	Label Encoding was applied to categorical variables like diabetesmellitus, appetite, pus_cell. Categorical cleanup (like replacing \tno) was performed.
Feature Engineering	The class column was cleaned and encoded to convert it into a binary target label. No new features were created, but multiple encodings and corrections were applied to existing ones.
Save Processed Data	Final models were saved using `pickle.dump()` for Logistic Regression, Gradient Boosting, Decision Tree, and Random Forest.