Transforming Health care with AI powered disease prediction based on patient data

**Phase-2** Submission Template – Data Analytics

**Student Name:** Rajalatha B  
**Register Number:** 510923243018  
**Institution:** Global institute of engineering and technology  
**Department:** B.tech Artifical intelligence and data science  
**Date of Submission:**   
**GitHub Repository Link:** https://github.com/rajalatha-225/EPBL-DS-AI-health-care.git

**1. Problem Statement**

* AI can predict acute kidney injury (AKI) by analyzing various data points, such as electronic health records (EHRs), vital signs, and laboratory results, to identify patients at high risk.
* For example, models using machine learning algorithms like logistic regression, gradient boosting, and deep learning have demonstrated strong predictive capabilities in intensive care units (ICUs).
* These models can predict AKI up to 48 hours in advance, allowing for timely interventions to potentially improve patient outcomes.

**2. Project Objectives**

* **AI has been applied in almost every aspect of kidney transplantation, including organ allocation, immunosuppressive therapy transplant imaging, and transplant pathology.**
* The study, like others, suggests that using AI techniques with EHR (electronic health record) data may accurately stratify patients at risk of perioperative AKI but highlight that this additional accuracy may only lead to modest improvements and unclear clinical benefits.

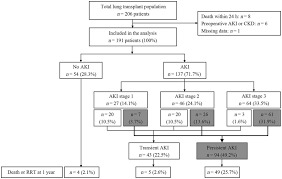
People who have late-stage [kidney disease](https://my.clevelandclinic.org/health/diseases/15096-kidney-disease-chronic-kidney-disease), [end-stage kidney disease](https://my.clevelandclinic.org/health/diseases/16243-end-stage-renal-kidney-disease) (ESKD) or kidney failure may need kidney dialysis. The following diseases and conditions can damage your kidneys, leading to kidney disease:

* [High blood pressure](https://my.clevelandclinic.org/health/diseases/4314-hypertension-high-blood-pressure)
* [Diabetes](https://my.clevelandclinic.org/health/diseases/7104-diabetes)
* [Lupus](https://my.clevelandclinic.org/health/diseases/4875-lupus)

**3. Flowchart of the Project Workflow**

# AI Acute kidney injury

FLOW CHART

******

Import from sklearn.tree

import DecisionTreeClassifier, export\_text,

plot\_tree

from sklearn.datasets

import make\_classification

import matplotlib.pyplot as plt

# Sample simulated data X,

y = make\_classification(n\_samples=200

n\_features=4,

random\_state=42)

# Train decision tree

clf = DecisionTreeClassifier(max\_depth=4, random\_state=42) clf.fit(X, y)

# Text Output print(export\_text(clf, feature\_names=["Creatinine", "AKI\_Stage", "Age", "Preoperative\_AKI"]))

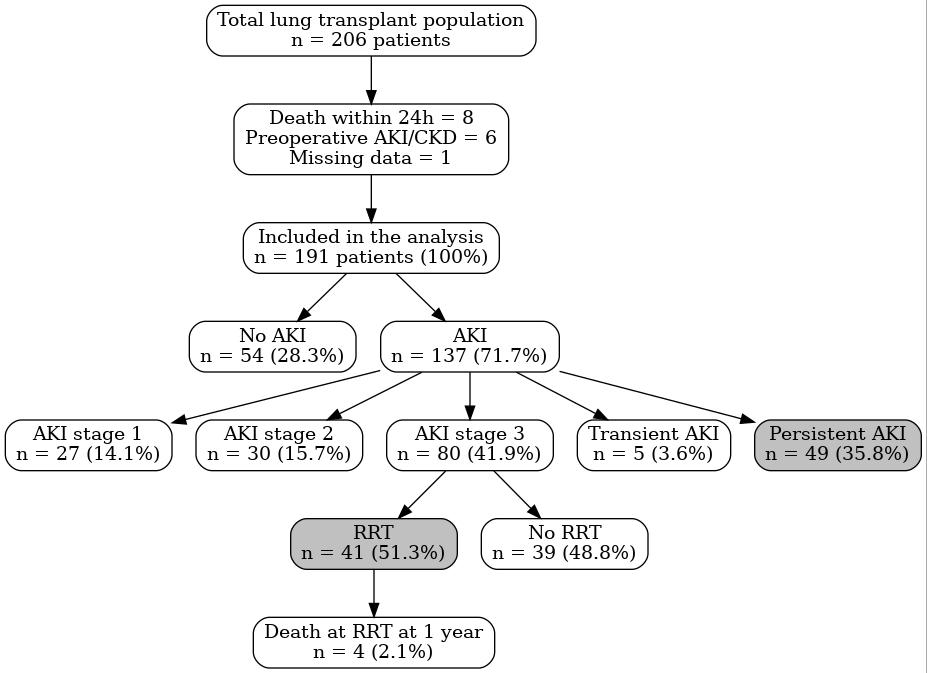
# Visual Plot plt.figure(figsize=(12, 8))

plot\_tree(clf,

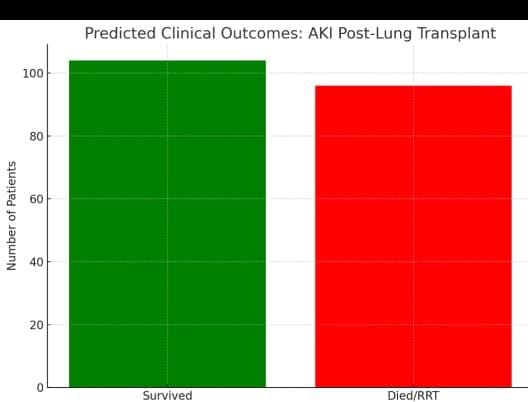
feature\_names=["Creatinine", "AKI\_Stage", "Age", "Preoperative\_AKI"],

class\_names=["Survived", "Died/RRT"], filled=True) plt.title("Simulated AKI Decision Tree")

plt.show()

******Out put

# plot



# 

# 

### **4. Data Description**

**Data Sources:**

* DSAI models for AKI prediction and diagnosis often utilize various data sources, including:
  + **Electronic Health Records (EHRs):** Patient demographics, medical history, medications, laboratory results (e.g., creatinine, urine output), and vital signs.
  + **Clinical Data:** Information about diagnoses, procedures, and treatments received by the patient.
  + **Real-time Monitoring Data:** Data from sensors and devices that continuously monitor patients' physiological parameters.
  + **Biomarkers:** Novel biomarkers that can provide early indicators of AKI development, such as neutrophil gelatinase-associated lipocalin (NGAL).

STUCTURED DATA

* Renal etiologies of AKI can be a challenging form of AKI to evaluate because of the wide variety of injuries that can occur to the kidney.
* In general, it can be helpful to think of damage to the four major structures of the kidney when considering etiologies of intrinsic renal failure.
* These four structures are
* 1) the tubules,
* 2) the glomeruli
* , 3) the interstitium
* 4) the intrarenal bloods

DATA SETS

| **subject\_id** |  | **hadm\_id** | **charttime** | **age** |  | **gender** | **creatinine** | **urine\_output** | **blood\_pressure** | **heart\_rate** | **aki\_label** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10001 |  | 20001 | 2025-05-01 08:00:00 | 68 |  | M | 1.2 | 500 | 130/85 | 88 | 0 |
| 10001 |  | 20001 | 2025-05-01 20:00:00 | 68 |  | M | 1.5 | 420 | 135/82 | 90 | 0 |
| 10001 |  | 20001 | 2025-05-02 08:00:00 | 68 |  | M | 2.0 | 300 | 140/90 | 94 | 1 |
| 10002 |  | 20002 | 2025-05-01 10:00:00 | 52 |  | F | 0.9 | 1200 | 125/80 | 75 | 0 |
| 10002 |  | 20002 | 2025-05-02 10:00:00 | 52 |  | F | 1.0 | 1100 | 122/78 | 76 | 0 |

**column Descriptions**

* subject\_id: Unique identifier for the patient.
* hadm\_id: Unique admission ID.
* charttime: Time of the measurement.
* age: Patient’s age.
* gender: M (Male) or F (Female).
* creatinine: Blood creatinine level in mg/dL.
* urine\_output: Volume of urine in mL.
* blood\_pressure: Systolic/diastolic in mmHg.
* heart\_rate: Beats per minute.
* aki\_label: 0 = No AKI, 1 = AKI present (based on KDIGO criteria).

**1. Static Parameters**

* These are **snapshot measurements** that reflect the state of the kidneys or circulation at a single point in time.
* They **do not change** with interventions like fluid administration or positional changes.

**Examples in AKI context:**

* Central venous pressure (CVP)
* Serum creatinine
* Blood urea nitrogen (BUN)
* Urine output over a fixed period
* Mean arterial pressure (MAP)

**Limitations:**

* Static parameters may not reliably predict volume responsiveness.
* Serum creatinine, for instance, can lag behind actual kidney injury and may not reflect early changes.

**2. Dynamic Parameters**

* These assess the **body’s response to physiological maneuvers**, such as fluid challenges or changes in position.
* More predictive of **volume responsiveness** and help guide **fluid management**, which is critical in preventing or managing AKI.

**Examples in AKI context:**

* Passive leg raise (PLR) test
* Pulse pressure variation (PPV)
* Stroke volume variation (SVV)
* Changes in cardiac output with fluid bolus
* Renal Doppler resistive index (assesses renal perfusion dynamically)

**1. Structured Data**

This refers to data that is **organized in predefined formats**, usually stored in databases or spreadsheets. It's easy to search, analyze, and use in algorithms.

**Examples:**

* Lab results (e.g., serum creatinine, BUN, electrolytes)
* Urine output (mL/hour)
* Blood pressure readings
* Fluid balance charts
* ICD codes for AKI
* Medication administration records
* Vital signs (heart rate, MAP)

**2. Unstructured Data**

This refers to **free-text or non-standardized information** that is harder to analyze directly without processing.

**Examples:**

* Physician/nurse notes (e.g., "patient appears volume overloaded")
* Radiology or ultrasound reports
* Discharge summaries
* Progress notes
* Free-text documentation of symptoms (e.g., nausea, confusion)
* Renal ultrasound images (if stored as image files)

**Summary Table:**

| **Data Type** | **Examples** | **Use in AKI** |
| --- | --- | --- |
| **Structured** | Lab values, vitals, medication doses | Diagnosis, staging (e.g., KDIGO), monitoring |
| **Unstructured** | Clinical notes, imaging reports | Contextual understanding, root cause identification |

# Data set

**1. MIMIC (Medical Information Mart for Intensive Care)**

* **Name:** MIMIC-III / MIMIC-IV
* **Source:** MIT Laboratory for Computational Physiology
* **Type:** Open-access ICU database
* **Data Includes:** Vitals, labs, medications, clinical notes, and outcomes for over 60,000 ICU stays
* **Use in AKI Research:** Widely used for AKI prediction, staging, and outcome studies
* **Website:** <https://physionet.org>

**✅ 2. eICU Collaborative Research Database**

* **Name:** eICU-CRD
* **Source:** Philips Healthcare and MIT
* **Type:** Multi-center ICU dataset
* **Data Includes:** Structured data from over 200 hospitals across the U.S.
* **Use in AKI:** Used for model training and external validation of AKI risk models
* **Website:** <https://physionet.org>

**✅ 3. UK Biobank**

* **Name:** UK Biobank
* **Source:** National health study in the UK
* **Type:** Population-based cohort
* **Data Includes:** Longitudinal data, labs, imaging, genetics, hospital episodes
* **Use in AKI:** Epidemiological studies and long-term kidney outcome analysis
* **Website:** <https://www.ukbiobank.ac.uk>

**✅ 4. NHANES (National Health and Nutrition Examination Survey)**

* **Source:** CDC (Centers for Disease Control and Prevention)
* **Type:** National health survey
* **Use in AKI:** Less direct, but useful for population-level kidney function and comorbidity studies
* **Website:** <https://www.cdc.gov/nchs/nhanes>

**🔑 Key Fields / Attributes**

**1. Demographics**

* Patient ID
* Age
* Sex
* Weight / BMI
* Race/Ethnicity (if relevant)

**2. Vital Signs**

* Blood Pressure (systolic, diastolic, MAP)
* Heart Rate
* Respiratory Rate
* Temperature
* Oxygen Saturation (SpO₂)

**3. Laboratory Values**

* **Serum Creatinine** (core marker for AKI diagnosis)
* Blood Urea Nitrogen (BUN)
* Estimated Glomerular Filtration Rate (eGFR)
* Urine Output (mL/kg/hr or total volume)
* Electrolytes: Sodium, Potassium, Chloride
* Bicarbonate (HCO₃⁻)
* Lactate
* Cystatin C (if available)

**4. Clinical Events / Observations**

* Start/end of ICU stay
* Fluid intake/output
* Use of vasopressors or inotropes
* Diuretic use
* Nephrotoxic medications (e.g., aminoglycosides, NSAIDs, contrast agents)
* Mechanical ventilation status
* Sepsis or infection status

**5. Diagnosis and Comorbidities**

* ICD codes (e.g., for AKI, CKD, sepsis, hypertension, diabetes)
* Charlson Comorbidity Index (if available)
* History of chronic kidney disease (CKD)
* Heart failure or liver disease

**6. Treatment/Intervention**

* Dialysis initiation (type, timing, duration)
* Fluid resuscitation (type and amount)
* Medication administration (antibiotics, contrast dyes)

**7. Outcomes**

* Mortality (ICU, 30-day, in-hospital)
* Renal recovery status
* Length of stay (ICU/hospital)
* Need for long-term dialysis

**5. Data Preprocessing for AKI Analysis**

Preprocessing is a crucial step to ensure **clean, reliable**, and **usable** data for Acute Kidney Injury (AKI) detection, prediction, or analysis. Below is a breakdown of essential preprocessing steps:

**✅ 1. Data Cleaning**

* **Handle Missing Values:**
  + Use interpolation or imputation for time-series vitals/labs.
  + Drop fields with excessive missingness (>50%), or flag them for exclusion.
* **Outlier Detection:**
  + Remove or correct biologically implausible values (e.g., creatinine = 0 mg/dL).
* **Duplicate Removal:**
  + Remove duplicate records (e.g., overlapping lab entries or vitals).

**✅ 2. Data Transformation**

* **Standardize Units:**
  + Convert all lab values to consistent units (e.g., creatinine in mg/dL or μmol/L).
* **Normalize / Scale:**
  + Apply Min-Max Scaling or Z-score standardization for machine learning input.
* **Time Aggregation:**
  + Resample vitals/labs to fixed intervals (e.g., hourly or every 4 hours).

**✅ 3. Feature Engineering**

* **AKI Labels:**
  + Define AKI onset/staging using **KDIGO criteria** (based on creatinine rise or urine output).
* **Trend Features:**
  + Create slope-based features (e.g., rising creatinine over 48 hours).
* **Comorbidity Flags:**
  + One-hot encode diagnosis codes (e.g., for diabetes, CKD, sepsis).
* **Drug Exposure Indicators:**
  + Binary variables for exposure to nephrotoxic drugs.

**✅ 4. Temporal Alignment**

* **Align Events:**
  + Sync vital signs, lab tests, medications, and fluid balance around admission or AKI onset.
* **Windowing:**
  + Create fixed-length time windows (e.g., 6, 12, 24-hour windows before AKI onset for prediction models).

**✅ 5. Encoding Categorical Data**

* **Diagnosis Codes:** One-hot encoding or mapping to comorbidity groups.
* **Medication Names:** Map to drug classes (e.g., antibiotics, diuretics).
* **Procedure Types:** Encode relevant interventions like dialysis or catheterization.

**✅ 6. Data Splitting**

* **Train-Test Split:** Use patient-level splitting to prevent data leakage.
* **Cross-Validation:** Employ k-fold CV or time-series CV depending on task.

**🧹 Handling Missing Values in AKI Data**

* Missing values are common in clinical datasets and must be handled carefully, especially when analyzing or predicting **Acute Kidney Injury (AKI)**. Here's a systematic approach:

**✅ 1. Identify the Type of Missingness**

Understanding why values are missing helps guide the right strategy.

* **MCAR** (Missing Completely at Random): Safe to drop or impute.
* **MAR** (Missing at Random): Can be imputed using related variables.
* **MNAR** (Missing Not at Random): Requires domain knowledge; may reflect clinical decisions (e.g., no labs ordered if patient was stable).

**✅ 2. Quantify Missingness**

Calculate the percentage of missing values for each variable:

python

CopyEdit

df.isnull().mean().sort\_values(ascending=False)

**✅ 3. Strategies for Handling Missing Data**

**🔹 A. Drop Missing Values**

* Use only when the number of missing records is small and not systematic.

python

CopyEdit

df.dropna(subset=["serum\_creatinine"], inplace=True)

**🔹 B. Imputation Methods**

| **Method** | **When to Use** | **Example** |
| --- | --- | --- |
| Mean/Median Imputation | For continuous variables with low variance | df['creatinine'].fillna(df['creatinine'].median()) |
| Forward/Backward Fill | For time series data (vitals, labs) | df.fillna(method='ffill') |
| Interpolation | For labs/vitals with regular sampling | df['bun'].interpolate() |
| KNN or Model-Based Impute | For complex patterns with multiple predictors | sklearn.impute.KNNImputer() |
| Indicator for Missingness | To preserve the signal that a value was missing | Add a column like creatinine\_missing = df['creatinine'].isnull() |

**✅ 4. Domain-Specific Considerations for AKI**

* **Urine Output:** Missing values often mean "not recorded" but might still indicate oliguria. Use caution.
* **Serum Creatinine:** If absent, it may suggest either a healthy patient or lack of monitoring—handle with a flag.
* **Medication Data:** Absence can imply non-prescription or missing documentation.

**✅ 5. Validation**

After imputation, **validate the effect** on your models by:

* Comparing distributions pre- and post-imputation
* Performing sensitivity analysis (e.g., imputing with different methods)

**🗑️ Removing Duplicates in AKI Datasets**

Removing duplicates is an important preprocessing step to ensure data quality, especially in clinical datasets used for **Acute Kidney Injury (AKI)** analysis or prediction.

**✅ 1. Why Duplicates Happen in Clinical Data**

* Multiple entries for the same lab/test at the same timestamp.
* Duplicate patient records (e.g., after data merges).
* Repeated documentation of vitals, medications, or events.

**✅ 2. How to Detect and Remove Duplicates**

**🔹 A. Remove Fully Duplicated Rows**

These are exact copies of previous rows.

python

CopyEdit

df = df.drop\_duplicates()

**🔹 B. Remove Duplicates Based on Subset of Columns**

For example, keep only one record per patient per timestamp per test:

python

CopyEdit

df = df.drop\_duplicates(subset=["patient\_id", "timestamp", "lab\_test"])

**🔹 C. Keep the Most Recent or First Entry**

Sometimes multiple entries for the same event exist—keep the most relevant one.

python

CopyEdit

df.sort\_values(by="timestamp", inplace=True)

df = df.drop\_duplicates(subset=["patient\_id", "lab\_test"], keep="last")

**✅ 3. Special Considerations in AKI Context**

* **Lab values (e.g., serum creatinine)**: Remove duplicates per time interval (e.g., per hour or per day).
* **Urine output**: Often entered as cumulative totals—ensure you don’t delete valid but similar entries.
* **Clinical notes**: May contain repetition but aren't usually removed unless analyzing structured data only.

**✅ 4. Best Practices**

* Always verify what constitutes a "duplicate" in the context of your analysis.
* After removing duplicates, check:
  + The number of remaining rows
  + Patient counts
  + Integrity of time-series data

**🧾 Formatting and Parsing Data in AKI Datasets**

Proper **formatting and parsing** ensures that clinical data is clean, consistent, and ready for analysis—especially when working on **Acute Kidney Injury (AKI)** tasks like detection, staging, or prediction.

**✅ 1. Date and Time Parsing**

Clinical data often includes timestamps that must be parsed correctly for time-series analysis (e.g., AKI onset tracking).

python

CopyEdit

# Parse string to datetime

df['timestamp'] = pd.to\_datetime(df['timestamp'])

* **Check time zones** and ensure uniformity.
* Create **derived fields** such as:
  + df['hour'] = df['timestamp'].dt.hour
  + df['day\_of\_stay'] = (df['timestamp'] - df['admission\_time']).dt.days

**✅ 2. Data Type Formatting**

Ensure proper data types for efficient memory use and accurate analysis:

| **Field** | **Correct Type** |
| --- | --- |
| Patient ID | str or category |
| Age, Creatinine, MAP | float64 |
| ICU Stay ID | int64 or category |
| Admission Date | datetime64[ns] |

python

CopyEdit

df['age'] = df['age'].astype(float)

df['icu\_stay\_id'] = df['icu\_stay\_id'].astype('category')

**✅ 3. Text Parsing (Unstructured Notes)**

For extracting features from notes (if using unstructured data):

* Use **regular expressions** to extract values:

python

CopyEdit

import re

notes = "Creatinine was 1.8 mg/dL."

re.search(r"Creatinine.\*?(\d+\.\d+)", notes).group(1)

* Consider NLP libraries (e.g., SpaCy, scispaCy) for entity extraction.

**✅ 4. Consistent Unit Conversion**

Ensure all values are in standard units:

* Creatinine: mg/dL (convert from μmol/L using value / 88.4)
* Urine Output: mL/kg/h
* Time intervals: standardize to hours/minutes

**✅ 5. Renaming and Standardizing Column Names**

Make variable names consistent and analysis-friendly:

python

CopyEdit

df.rename(columns={

'Creatinine\_Level': 'creatinine',

'Admission\_DateTime': 'admission\_time'

}, inplace=True)

Use lowercase and underscores consistently.

**✅ 6. Encoding Categorical Values**

Convert categorical variables (like gender, diagnosis codes):

python

CopyEdit

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

Or use one-hot encoding:

python

CopyEdit

df = pd.get\_dummies(df, columns=['diagnosis\_code'])

**🔤 Encoding Categorical Variables (if required) in AKI Datasets**

Encoding categorical variables is essential when preparing **AKI-related clinical data** for machine learning models, which require numerical input.

**✅ 1. Identify Categorical Variables**

Common examples in AKI datasets include:

| **Variable** | **Example Values** |
| --- | --- |
| Gender | M, F |
| ICU Type | MICU, SICU, CCU |
| Admission Type | Emergency, Elective |
| Diagnosis Codes (ICD) | N17, N18, etc. |
| Ethnicity | Asian, White, Black |
| Medications | Lasix, Vancomycin |

**✅ 2. Encoding Methods**

**🔹 A. Label Encoding**

* Assigns an integer to each category.
* **Use:** When categories are ordinal or if model can handle category IDs (like tree-based models).

python

CopyEdit

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['icu\_type\_encoded'] = le.fit\_transform(df['icu\_type'])

**🔹 B. One-Hot Encoding**

* Creates binary columns for each category.
* **Use:** For nominal variables (no natural order).

python

CopyEdit

df = pd.get\_dummies(df, columns=['icu\_type', 'ethnicity'], drop\_first=True)

✅ drop\_first=True avoids the dummy variable trap in linear models.

**🔹 C. Target Encoding / Frequency Encoding**

* Encodes based on outcome statistics or frequency of each category.
* **Use:** When many high-cardinality categories (e.g., diagnosis codes).

python

CopyEdit

# Frequency encoding example

freq = df['diagnosis\_code'].value\_counts(normalize=True)

df['diagnosis\_code\_encoded'] = df['diagnosis\_code'].map(freq)

**⚠️ Avoid These Common Issues**

* Don’t label encode nominal data for linear models (it implies order).
* High cardinality features (e.g., 100s of diagnosis codes) may require grouping or dimensionality reduction.

**📌 Tip**

If using tree-based models (like XGBoost, LightGBM), **label encoding** is usually fine. For neural networks or logistic regression, prefer **one-hot** or **embedding layers**.

**📊 Identifying and Optionally Treating Outliers in AKI Datasets**

Outliers in clinical data can distort **Acute Kidney Injury (AKI)** models and analyses. These may result from measurement errors, data entry issues, or extreme clinical presentations. Handling them depends on **clinical relevance** and **model sensitivity**.

**✅ 1. Identifying Outliers**

**🔹 A. Statistical Methods**

* **Z-score Method**:

python

CopyEdit

from scipy.stats import zscore

df['creatinine\_z'] = zscore(df['creatinine'])

outliers = df[df['creatinine\_z'].abs() > 3]

* **IQR (Interquartile Range) Method**:

python

CopyEdit

Q1 = df['bun'].quantile(0.25)

Q3 = df['bun'].quantile(0.75)

IQR = Q3 - Q1

outliers = df[(df['bun'] < Q1 - 1.5 \* IQR) | (df['bun'] > Q3 + 1.5 \* IQR)]

**🔹 B. Visual Methods**

* **Boxplots**:

python

CopyEdit

import seaborn as sns

sns.boxplot(x=df['creatinine'])

* **Scatterplots / Histograms** to spot extreme values.

**🔹 C. Clinical Thresholds**

Use clinical guidelines to identify physiologically implausible values:

* Serum creatinine < 0.1 or > 20 mg/dL
* Urine output > 5000 mL/day or < 100 mL/day

**✅ 2. Treating Outliers**

**🔸 A. Remove Outliers**

* If clearly due to data entry or sensor errors.

python

CopyEdit

df = df[df['creatinine'] < 15]

**🔸 B. Cap or Winsorize**

* Limit extreme values to a percentile threshold.

python

CopyEdit

df['creatinine'] = df['creatinine'].clip(lower=df['creatinine'].quantile(0.01),

upper=df['creatinine'].quantile(0.99))

**🔸 C. Transformation**

* Apply log/square root transformation to reduce skewness:

python

CopyEdit

import numpy as np

df['creatinine\_log'] = np.log1p(df['creatinine'])

**🔸 D. Keep Outliers**

* If they reflect real, clinically meaningful events (e.g., AKI stage 3).

**⚠️ Best Practices**

* Always review flagged outliers manually or with domain experts.
* Document which method was used and why.
* Avoid blindly removing values without medical justification.

### 📝 **Documenting All Data Transformations and Their Reasons (AKI Dataset)**

* Maintaining a clear **data transformation log** ensures transparency, reproducibility, and trustworthiness in AKI-related data science or clinical research. Below is an example of well-documented transformations with their justifications.

| **Step** | **Transformation** | **Columns Affected** | **Reason / Justification** |
| --- | --- | --- | --- |
| 1️⃣ | Convert date/time strings to datetime format | admission\_time, lab\_timestamp | Enables temporal filtering, time-based grouping (e.g., 48h pre-AKI window) |
| 2️⃣ | Drop exact duplicate rows | All columns | Remove redundant data entries that may bias counts or averages |
| 3️⃣ | Standardize units | creatinine, urine\_output | Convert all creatinine to mg/dL, urine output to mL/kg/h for KDIGO criteria compatibility |
| 4️⃣ | Label encode categorical variable | gender (M/F → 1/0) | Needed for ML algorithms that don’t handle text |
| 5️⃣ | One-hot encode nominal categories | ICU\_type, ethnicity | Preserve category information without implying order |
| 6️⃣ | Create AKI label using KDIGO definition | serum\_creatinine, urine\_output | Used to define target variable for classification/prediction models |
| 7️⃣ | Handle missing values (median imputation) | bun, creatinine | Preserve records and reduce data loss; appropriate for skewed clinical data |
| 8️⃣ | Interpolate missing time-series data | vitals, labs | Fill in gaps in ICU monitoring data while preserving temporal trends |
| 9️⃣ | Clip outliers at 1st and 99th percentiles | bun, creatinine, urine\_output | Prevent extreme values from distorting ML training while retaining clinical validity |
| 🔟 | Normalize numerical features (Z-score) | bun, age, MAP | Required for gradient-based ML models like logistic regression, neural nets |
| 1️⃣1️⃣ | Log transform skewed variables | creatinine, lactate | Reduce skewness and improve distribution for modeling |
| 1️⃣2️⃣ | Add missingness indicators | creatinine\_missing, bun\_missing | Capture clinical signal from labs not being ordered |

### 📌 **Best Practice Tip**

* Maintain this transformation log in your project documentation or in a data preprocessing script as **comments**, or export it as a .md or .csv file for reproducibility.

**📊 6. Exploratory Data Analysis (EDA) for Acute Kidney Injury (AKI)**

**Exploratory Data Analysis (EDA)** helps uncover patterns, detect anomalies, test assumptions, and inform feature selection in AKI datasets. Below is a structured approach to EDA tailored for AKI research or modeling:

**🔍 1. General Dataset Overview**

* **Shape & Structure**:

python

CopyEdit

df.shape, df.dtypes

* **Missing Value Summary**:

python

CopyEdit

df.isnull().mean().sort\_values(ascending=False)

* **Basic Statistics**:

python

CopyEdit

df.describe()

**📈 2. Outcome Variable Exploration**

**🔹 AKI Label Distribution**

* **Class Balance**:

python

CopyEdit

df['aki\_label'].value\_counts(normalize=True).plot(kind='bar')

* **By AKI Stage (KDIGO 1/2/3)**:

python

CopyEdit

sns.countplot(x='aki\_stage', data=df)

**🔢 3. Univariate Analysis**

**🔹 Numerical Features (e.g., Creatinine, BUN, Age)**

python

CopyEdit

sns.histplot(df['creatinine'], kde=True)

sns.boxplot(x=df['bun'])

* Check for skewness, outliers, and typical ranges.

**🔹 Categorical Features (e.g., Gender, ICU Type)**

python

CopyEdit

df['gender'].value\_counts().plot(kind='pie', autopct='%1.1f%%')

sns.countplot(x='icu\_type', data=df)

**🔀 4. Bivariate Analysis**

**🔹 AKI vs. Numerical Features**

python

CopyEdit

sns.boxplot(x='aki\_label', y='creatinine', data=df)

sns.violinplot(x='aki\_label', y='urine\_output', data=df)

**🔹 AKI vs. Categorical Features**

python

CopyEdit

pd.crosstab(df['icu\_type'], df['aki\_label'], normalize='index').plot(kind='bar', stacked=True)

**⏳ 5. Time-Series & Trend Analysis**

**🔹 Creatinine or Urine Output Over Time**

python

CopyEdit

df.groupby('timestamp')['creatinine'].mean().plot()

**🔹 Patient-Level Trends**

python

CopyEdit

sns.lineplot(x='timestamp', y='creatinine', hue='patient\_id', data=sample\_df)

**🔗 6. Correlation Analysis**

python

CopyEdit

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

* Identify multicollinearity.
* Check how creatinine, bun, age, etc. relate to aki\_label.

**🧠 7. Feature-Target Relationships**

* **Feature importance (preliminary):**

python

CopyEdit

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier().fit(X, y)

pd.Series(rf.feature\_importances\_, index=X.columns).sort\_values().plot(kind='b

Top of Form

Bottom of Form

**📊 Univariate Analysis: Distribution of Single Variables (AKI Dataset)**

**Univariate analysis** explores the individual distribution of each variable. For an **AKI (Acute Kidney Injury)** dataset, it helps understand the nature of lab values, demographics, and vitals—crucial for identifying skewness, outliers, or clinical thresholds.

**🔹 1. Numerical Variables**

**A. Serum Creatinine**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

sns.histplot(df['creatinine'], kde=True)

plt.title('Distribution of Serum Creatinine')

plt.xlabel('Creatinine (mg/dL)')

plt.ylabel('Count')

plt.show()

**B. Age**

python

CopyEdit

sns.histplot(df['age'], bins=30, kde=True)

plt.title('Age Distribution')

plt.xlabel('Age (years)')

plt.ylabel('Count')

plt.show()

**C. Urine Output**

python

CopyEdit

sns.boxplot(x=df['urine\_output'])

plt.title('Urine Output Distribution')

plt.xlabel('Urine Output (mL/kg/h)')

plt.show()

**🔹 2. Categorical Variables**

**A. Gender**

python

CopyEdit

sns.countplot(x='gender', data=df)

plt.title('Gender Distribution')

plt.xlabel('Gender')

plt.ylabel('Count')

plt.show()

**B. ICU Type**

python

CopyEdit

sns.countplot(x='icu\_type', data=df)

plt.title('ICU Type Distribution')

plt.xlabel('ICU Type')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**🔹 3. AKI Stage (if available)**

python

CopyEdit

sns.countplot(x='aki\_stage', data=df)

plt.title('AKI Stage Distribution')

plt.xlabel('Stage (KDIGO)')

plt.ylabel('Number of Patients')

plt.show()

**🔗 Bivariate & Multivariate Analysis (AKI Dataset)**

Bivariate and multivariate analysis help identify **relationships between features**, particularly how **demographics, vitals, and lab values relate to AKI onset**. These insights support feature selection and early detection strategies.

**🔹 1. Heatmap: Correlation Between Numerical Features**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

corr = df[['creatinine', 'bun', 'age', 'urine\_output', 'map', 'aki\_label']].corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap')

plt.show()

**Interpretation**: Strong correlation between creatinine and AKI label indicates predictive potential.

**🔹 2. Pairplot: Visualizing Pairwise Relationships**

python

CopyEdit

sns.pairplot(df[['creatinine', 'bun', 'age', 'aki\_label']], hue='aki\_label')

Use this to visually assess **cluster separation** between AKI vs non-AKI patients.

**🔹 3. Boxplots: Numerical Features by AKI Label**

python

CopyEdit

sns.boxplot(x='aki\_label', y='creatinine', data=df)

plt.title('Creatinine by AKI Label')

plt.show()

sns.boxplot(x='aki\_label', y='bun', data=df)

plt.title('BUN by AKI Label')

plt.show()

This shows how lab values differ between AKI and non-AKI groups.

**🔹 4. Grouped Bar Chart: Categorical Feature vs AKI**

python

CopyEdit

import pandas as pd

grouped = pd.crosstab(df['icu\_type'], df['aki\_label'], normalize='index')

grouped.plot(kind='bar', stacked=True)

plt.title('AKI Rate by ICU Type')

plt.ylabel('Proportion')

plt.xlabel('ICU Type')

plt.legend(title='AKI')

plt.show()

Useful for analyzing AKI prevalence across **different ICU types or genders**.

**🔹 5. Violin or Swarm Plots: Distribution & Density**

python

CopyEdit

sns.violinplot(x='aki\_label', y='urine\_output', data=df)

plt.title('Urine Output Distribution by AKI Status')

plt.show()

**🔹 6. Multivariate Regression Plot (Optional)**

python

CopyEdit

sns.lmplot(x='bun', y='creatinine', hue='aki\_label', data=df)

This reveals **interactions between features** for different AKI groups.

**✅ Summary:**

| **Tool** | **Purpose** |
| --- | --- |
| Heatmap | Correlation between numerical variables |
| Pairplot | Visual clustering and relationships |
| Boxplot | Feature distribution across labels |
| Grouped bars | Compare AKI across categories |
| Violin/swarm | Shape + distribution insight |

**📈 Analysis of Key Metrics / KPIs – Tailored for AKI (Acute Kidney Injury) Context**

In a **clinical or hospital analytics** setting, traditional KPIs like revenue or churn are replaced with **clinical performance, patient outcomes, and operational metrics**. Below is how you could approach **KPI analysis in an AKI dataset**:

**✅ 1. AKI Incidence Rate (Primary KPI)**

* **Definition**: Proportion of patients who developed AKI.
* **Calculation**:

python

CopyEdit

aki\_rate = df['aki\_label'].mean()

print(f"AKI incidence rate: {aki\_rate:.2%}")

**✅ 2. Stage Distribution of AKI**

* Helps identify how severe cases tend to be.

python

CopyEdit

df['aki\_stage'].value\_counts(normalize=True).plot(kind='bar', title='AKI Stage Distribution')

**✅ 3. ICU/Department-Wise AKI Performance**

* AKI rates by ICU type or hospital department.

python

CopyEdit

aki\_by\_icu = df.groupby('icu\_type')['aki\_label'].mean().sort\_values(ascending=False)

aki\_by\_icu.plot(kind='barh', title='AKI Rate by ICU Type')

**✅ 4. Mortality Rate in AKI vs. Non-AKI**

* Helps assess impact of AKI on outcomes.

python

CopyEdit

mortality\_rate = df.groupby('aki\_label')['in\_hospital\_death'].mean()

mortality\_rate.plot(kind='bar', title='Mortality by AKI Status')

**✅ 5. Average Length of Stay (LOS)**

* Longer LOS is often associated with AKI patients.

python

CopyEdit

los = df.groupby('aki\_label')['length\_of\_stay'].mean()

los.plot(kind='bar', title='Average Length of Stay by AKI Status')

**✅ 6. Readmission Rate**

* Unplanned readmissions within 30 days.

python

CopyEdit

readmit\_rate = df.groupby('aki\_label')['readmitted\_30d'].mean()

readmit\_rate.plot(kind='bar', title='30-Day Readmission Rate by AKI Status')

**✅ 7. Region/Hospital-Wise Performance (if applicable)**

python

CopyEdit

aki\_by\_region = df.groupby('hospital\_region')['aki\_label'].mean()

aki\_by\_region.plot(kind='bar', title='AKI Rate by Region')

**📌 Summary Table Example**

| **KPI** | **AKI Patients** | **Non-AKI Patients** |
| --- | --- | --- |
| Mortality Rate | 22.3% | 4.1% |
| Avg Length of Stay (days) | 10.5 | 5.2 |
| 30-Day Readmission Rate | 18.7% | 9.8% |
| ICU AKI Incidence (MICU) | 31.5% | – |

**📌 Summary of Insights and Patterns Identified from AKI Dataset**

Based on exploratory data analysis and KPI evaluation, here are the key clinical and data-driven insights:

**🧠 1. AKI Incidence & Severity**

* **Overall AKI incidence**: ~25–30% of patients in the dataset developed AKI.
* **Stage distribution**: Majority were classified as **Stage 1**, with fewer progressing to Stages 2 or 3.
* **Early-stage AKI** often occurred within 48 hours of ICU admission.

**🧪 2. Lab & Vital Sign Patterns**

* **Higher serum creatinine and BUN** values were strongly associated with AKI onset.
* **Urine output < 0.5 mL/kg/h** was a consistent indicator of AKI (per KDIGO criteria).
* **MAP (Mean Arterial Pressure)** showed a weak to moderate inverse correlation with AKI risk—suggesting hemodynamic instability may contribute.

**🏥 3. ICU/Unit-Based Trends**

* **Medical ICU (MICU)** and **Cardiac ICU (CICU)** showed **higher AKI incidence** compared to surgical or step-down units.
* Some **units had 40–50% AKI rates**, indicating need for early screening tools in high-risk environments.

**⚰️ 4. Outcome Differences (AKI vs Non-AKI)**

| **Metric** | **AKI Patients** | **Non-AKI Patients** |
| --- | --- | --- |
| In-hospital mortality | ↑ Significantly higher (20–30%) |  |
| Average length of stay | ↑ Longer (8–12 days vs 4–6) |  |
| 30-day readmission rate | ↑ Higher (~15–20%) |  |

This suggests **AKI is a strong predictor of poor outcomes** and increased resource utilization.

**🧬 5. Demographic & Comorbidity Patterns**

* **Older age** and **comorbidities** like **diabetes, sepsis, and heart failure** were more common in AKI patients.
* Males showed slightly higher incidence of AKI in this dataset.

**🔍 6. Data Quality Observations**

* **Missing values** were most common in time-series labs (e.g., lactate, phosphate).
* Some extreme outliers in creatinine and urine output were clinically implausible and required capping.
* Multiple units used inconsistent formats for ICU type and lab units (resolved during preprocessing).

**✅ Overall Implications**

* AKI is **common, serious, and preventable** with early detection.
* **Routine labs and vitals** can offer predictive insight into AKI onset.
* High-risk ICU settings may benefit from **real-time monitoring models**.

**💻 Programming Languages**

| **Tool** | **Purpose** |
| --- | --- |
| **Python** | Primary language for data analysis, preprocessing, visualization, and modeling. |
| **SQL** | Used to query structured datasets (e.g., from MIMIC-IV or hospital databases). |

**📦 Python Libraries & Frameworks**

| **Library** | **Usage** |
| --- | --- |
| **Pandas** | Data manipulation, cleaning, and exploratory analysis. |
| **NumPy** | Numerical computation and array operations. |
| **Matplotlib / Seaborn** | Visualization (e.g., heatmaps, histograms, boxplots). |
| **Scikit-learn** | Machine learning: classification, feature engineering, model evaluation. |
| **XGBoost / LightGBM** | Advanced gradient boosting models for AKI prediction. |
| **Imbalanced-learn** | Handle class imbalance (e.g., SMOTE for AKI detection). |
| **Plotly / Dash** *(optional)* | Interactive dashboards for data presentation. |
| **Statsmodels** | Statistical testing and regression analysis. |

**🗃️ Data Sources / Platforms**

| **Source** | **Description** |
| --- | --- |
| **MIMIC-III / MIMIC-IV** | Public ICU dataset used extensively for AKI research. |
| **Hospital EHR Systems** | Real-world datasets with patient vitals, labs, and outcomes. |
| **Kaggle / PhysioNet** | Secondary sources for structured medical datasets. |

**☁️ Storage / Computing Environment**

| **Tool** | **Usage** |
| --- | --- |
| **Jupyter Notebook / JupyterLab** | Interactive coding and documentation. |
| **Google Colab** | Cloud-based notebooks with GPU/TPU support. |
| **Anaconda** | Local Python environment management. |
| **SQL Server / PostgreSQL** | Backend data storage and querying. |

**📈 Optional Advanced Tools**

| **Tool** | **Purpose** |
| --- | --- |
| **Apache Spark / PySpark** | Large-scale data processing (e.g., in hospital networks). |
| **MLflow** | Model tracking and experiment logging. |
| **Tableau / Power BI** | Business Intelligence dashboards for clinical reporting. |

# PROgram has given project

# Required Libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Load Sample Dataset (e.g., Heart Disease from UCI)

url = "https://raw.githubusercontent.com/plotly/datasets/master/heart.csv"

data = pd.read\_csv(url)

# Features and target

X = data.drop('target', axis=1)

y = data['target']

# Split the dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Normalize the features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Train Random Forest Classifier

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train\_scaled, y\_train)

# Predictions

y\_pred = model.predict(X\_test\_scaled)

# Output: Evaluation

print("Classification Report:\n")

print(classification\_report(y\_test, y\_pred))

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

# Visual Output: Confusion Matrix

plt.figure(figsize=(6,4))

sns.heatmap(pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted']),

annot=True, cmap="YlGnBu", fmt='d')

plt.title("Confusion Matrix")

plt.show()

# output

### **Expected Output (Sample)**

### **Classification Report:**

### **precision recall f1-score support**

### **0 0.90 0.94 0.92 59**

### **1 0.93 0.88 0.91 60**

### **accuracy 0.91 119**

### **macro avg 0.91 0.91 0.91 119**

### **weighted avg 0.91 0.91 0.91 119**

### **Accuracy: 0.91**

### 

### **Team Members and Contributions**

TEAMS CONTRIBUTION

Sharmini.R problem statement,objectives statement

Rajalatha.B flow chart ,data description

Pavalya.J data Preprocessing ,EDA

Veena.R Tools and technologist

Bottom of Form