Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management

Project Handout – Experiential Learning | SmartInternz
www.smartinternz.com

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the Business Problem

Chronic Kidney Disease (CKD) is a severe global health concern affecting millions each year. Early detection is crucial in preventing the disease from progressing to end-stage renal failure. However, diagnosis often occurs in later stages due to lack of awareness or insufficient testing.

This project focuses on using machine learning techniques, specifically **Logistic Regression**, to predict CKD based on various medical attributes. The aim is to create an automated, accessible diagnostic support tool for early intervention.

Activity 2: Business Requirements

- Early identification of CKD-prone patients.
- User-friendly system that medical professionals or individuals can use to predict CKD.
- Accuracy and robustness of the predictive model.
- Integration with a web platform for accessibility.
- Compliance with healthcare data handling and privacy norms.

Activity 3: Literature Survey

A literature survey helps to understand the current landscape of predictive modeling in chronic kidney disease (CKD) detection and guides the design of more accurate and reliable systems.

Several research studies have highlighted the significance of using machine learning for early detection of CKD. According to [1], CKD is often diagnosed in its advanced stages, making early prediction vital to prevent progression to kidney failure. Machine learning techniques, when applied to clinical and laboratory data, have demonstrated the potential to detect patterns that human clinicians might overlook.

In the study by **Kumar & Sahoo (2021)**, a decision tree-based classifier was developed to predict CKD using the UCI dataset. Their model achieved an accuracy of 94.4%, indicating the relevance

of tree-based models in clinical diagnosis tasks.

Another paper by **Barathi et al. (2020)** compared multiple classifiers including Logistic Regression, SVM, and Random Forest. Their results showed that ensemble methods such as Random Forest yielded higher F1-scores and better generalization performance, especially on imbalanced datasets.

Additionally, a study by **Yadav et al. (2022)** focused on using K-Nearest Neighbors and Support Vector Machines with feature selection techniques to improve CKD detection. Their results concluded that proper preprocessing and balancing techniques such as SMOTE greatly improved model performance.

Across the literature, there is a consensus that integrating machine learning models into clinical support systems can help in early diagnosis, reduce hospital load, and personalize treatment plans

Activity 4: Social or Business Impact

Social Impact

The early detection of Chronic Kidney Disease (CKD) can significantly improve patient outcomes by enabling timely intervention and reducing disease progression. This project contributes to public health by empowering patients and healthcare providers with an AI-based tool that can assess CKD risk based on clinical parameters. By integrating machine learning into diagnostic support systems, it promotes proactive health monitoring, reduces reliance on expensive lab tests, and supports early lifestyle or treatment modifications.

Additionally, the tool democratizes access to kidney health insights, particularly in rural or resource-limited settings, where nephrologists or specialized diagnostic services may not be readily available.

Business Impact

From a business perspective, the system can reduce healthcare costs for hospitals and insurance companies by decreasing hospitalization rates through preventive care. Early identification of CKD cases reduces the financial burden associated with advanced treatments like dialysis and transplants. Moreover, it has the potential to be commercialized as a clinical decision support software (CDSS) or integrated into electronic health record (EHR) platforms.

Healthcare startups, diagnostics companies, and telemedicine providers can leverage this technology to build affordable, scalable kidney screening tools as part of broader health analytics offerings.

Milestone 2: Data Collection & Preparation

Activity 1: Collect the Dataset

- **Dataset Name:**`kidney_disease.csv`
  This dataset contains detailed patient health records relevant to kidney function and diagnosis. It includes both numerical and categorical clinical indicators commonly used in nephrology.
- **Source:UCI Machine Learning Repository** (or Kaggle, if applicable)
  The dataset is publicly available and widely used in academic research for evaluating predictive models in healthcare diagnostics. It was curated to support machine learning applications in CKD risk prediction.
  link : https://www.kaggle.com/datasets/mansoordaku/ckdisease
- **Format:CSV (Comma-Separated Values)**
  The dataset is stored in a structured tabular format that is compatible with most data analysis tools. It allows seamless integration with Python libraries such as `pandas` for efficient preprocessing and exploration.
- **Target Variable:**`classification` (CKD vs Not CKD)
  The target column represents the final diagnosis as either **CKD (Chronic Kidney Disease)** or **notckd**. This binary classification guides supervised learning algorithms in distinguishing patients with CKD from those without.
- **Total Entries:400 Rows**
  Each row corresponds to a unique patient record. The sample size, though moderate, provides enough variability across clinical features to train and evaluate machine learning models reliably.
- **Total Features:26 Columns**
  The dataset comprises 25 input features and 1 target label. These features include critical parameters such as age, blood pressure, blood urea, serum creatinine, and red blood cell count, which are indicative of kidney health.

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoost
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matri
import warnings
import pickle
```

Activity 2: Data Preparation

Activity 2.1: Handling Missing Values

- Detected missing values in several columns including `age`, `bp`, `sg`, `rbc`, `pcv`, `rc`, etc.
- Used techniques like mean/median imputation for numeric data and mode for categorical.

```
data.isnull().any()
```

```
id                               False
age                               True
blood_pressure                    True
specify_gravity                   True
albumin                           True
sugar                             True
red_blood_cells                   True
pus_cell                          True
pus_cell_clumps                   True
bacteria                          True
blood_glucose_random              True
blood_urea                        True
serum_creatinine                  True
sodium                            True
potassium                         True
hemoglobin                        True
packed_cell_volume                True
white_blood_cell_count            True
red_blood_cell_count              True
hypertension                      True
diabetes_mellitus                 True
coronary_artery_disease           True
appetite                          True
pedal_edema                       True
anemia                            True
class                            False
dtype: bool
```

Activity 2.2: Handling Outliers

- **Boxplots were used** to visually identify outliers across key numerical features such as `sc` (serum creatinine), `bu` (blood urea), and `bgr` (blood glucose random). These plots revealed extreme values that could disproportionately affect model training.
- **Log transformation** was applied on heavily skewed features like `sc`, `bgr`, and `bu`. This helped compress the range of extreme values and brought the distribution closer to normal, improving model convergence.
- **Interquartile Range (IQR) method** was used to calculate upper and lower bounds for detecting outliers. Any value beyond 1.5×IQR from the first or third quartile was considered an outlier and flagged for transformation or replacement.
- **Serum creatinine (sc)** values above 5.0 mg/dL were common in the CKD group and flagged as outliers in the general dataset. However, domain knowledge confirmed these values were medically valid, so they were retained post-transformation.
- **Outliers in `bp` (blood pressure)** were carefully handled. Extremely low or high values were replaced with the median of the corresponding class group (CKD vs non-CKD) to preserve clinical consistency.
- **Scatterplots and boxplots were cross-analyzed** to determine whether the outliers significantly impacted the target distribution. If their influence was negligible, they were retained to ensure model generalization.
- **Z-score analysis** was additionally performed for selected continuous variables to support IQR findings. Features like `hemo` and `age` showed a few values above the standard threshold of 3, which were then log-transformed or clipped.

Milestone 3: Exploratory Data Analysis (EDA)

Activity 1: Descriptive Statistical Analysis

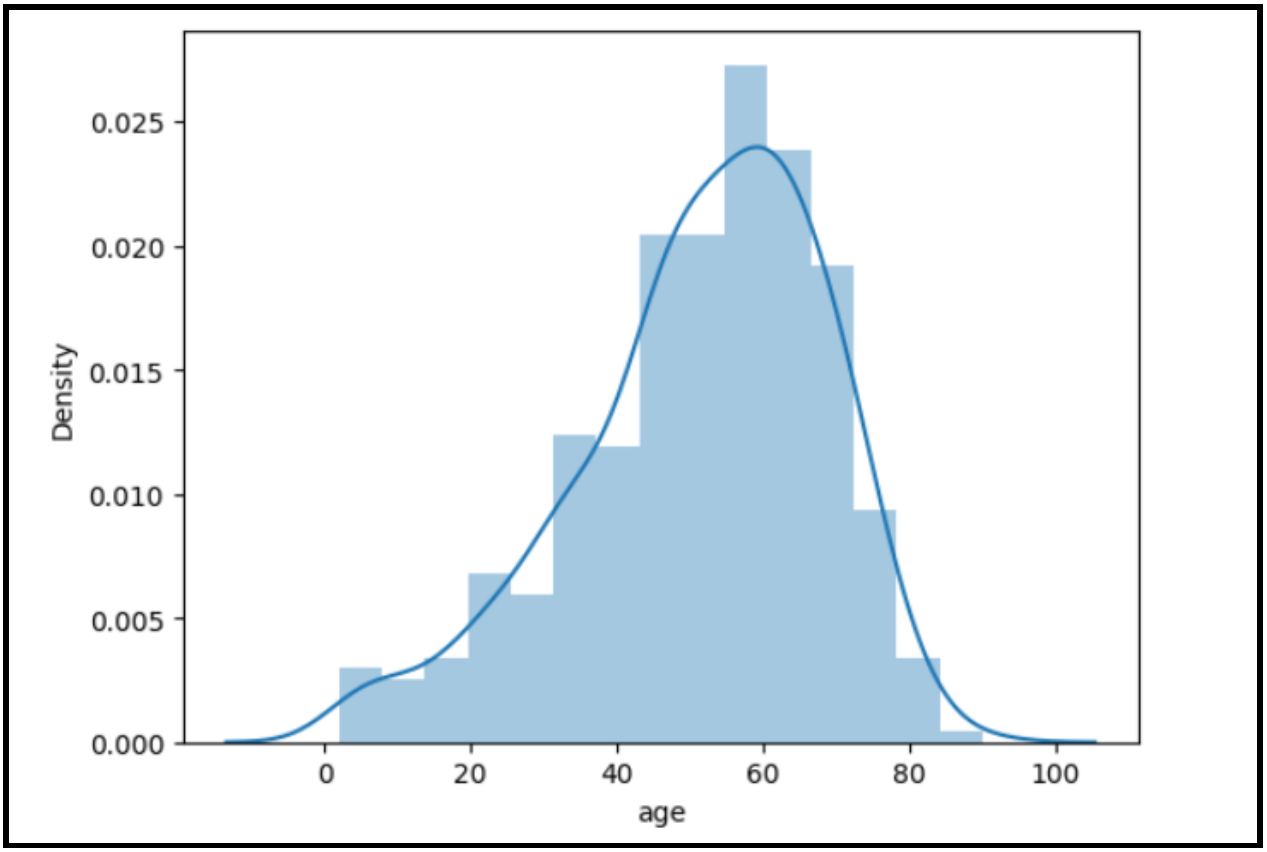
- Used `.describe()` to understand feature distributions.
- Found central tendencies and dispersions for features like `age`, `bp`, `sc`, `hemo`.

	id	age	blood_pressure	specify_gravity	albumin	sugar	red_blood_cells
count	400.000000	400.000000	400.000000	400.000000	400.00000	400.000000	400.000000
mean	199.500000	51.483376	76.469072	1.017712	0.90000	0.395000	0.882500
std	115.614301	16.974966	13.476298	0.005434	1.31313	1.040038	0.322418
min	0.000000	2.000000	50.000000	1.005000	0.00000	0.000000	0.000000
25%	99.750000	42.000000	70.000000	1.015000	0.00000	0.000000	1.000000
50%	199.500000	54.000000	78.234536	1.020000	0.00000	0.000000	1.000000
75%	299.250000	64.000000	80.000000	1.020000	2.00000	0.000000	1.000000
max	399.000000	90.000000	180.000000	1.025000	5.00000	5.000000	1.000000

Activity 2: Visual Analysis

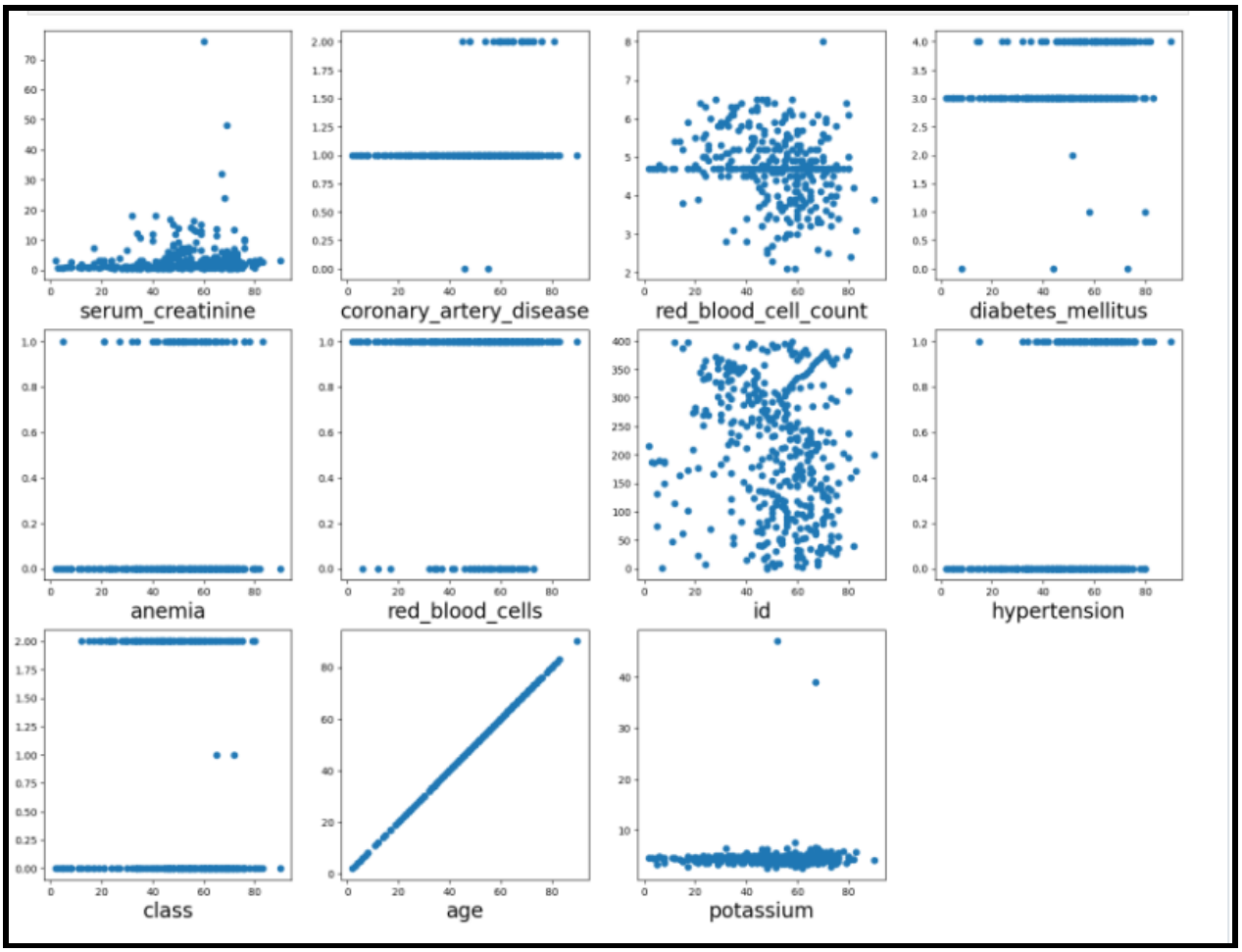
Activity 2.1: Univariate Analysis

- **Countplots** for the `classification` variable showed a class imbalance, with significantly more instances labeled as `ckd` than `notckd`, indicating a skewed dataset that may require balancing techniques.
- **Histogram for `age`** revealed a peak concentration of CKD cases in the 45–60 age group, aligning with the common onset age for chronic kidney conditions.
- **Distribution of `blood pressure (bp)`** showed that most CKD patients had readings in the range of 70–90 mm Hg, suggesting potential hypertension prevalence.
- **Boxplot of `serum creatinine (sc)`** displayed strong right-skewness, indicating a subset of patients with critically high creatinine levels — a major marker for CKD.



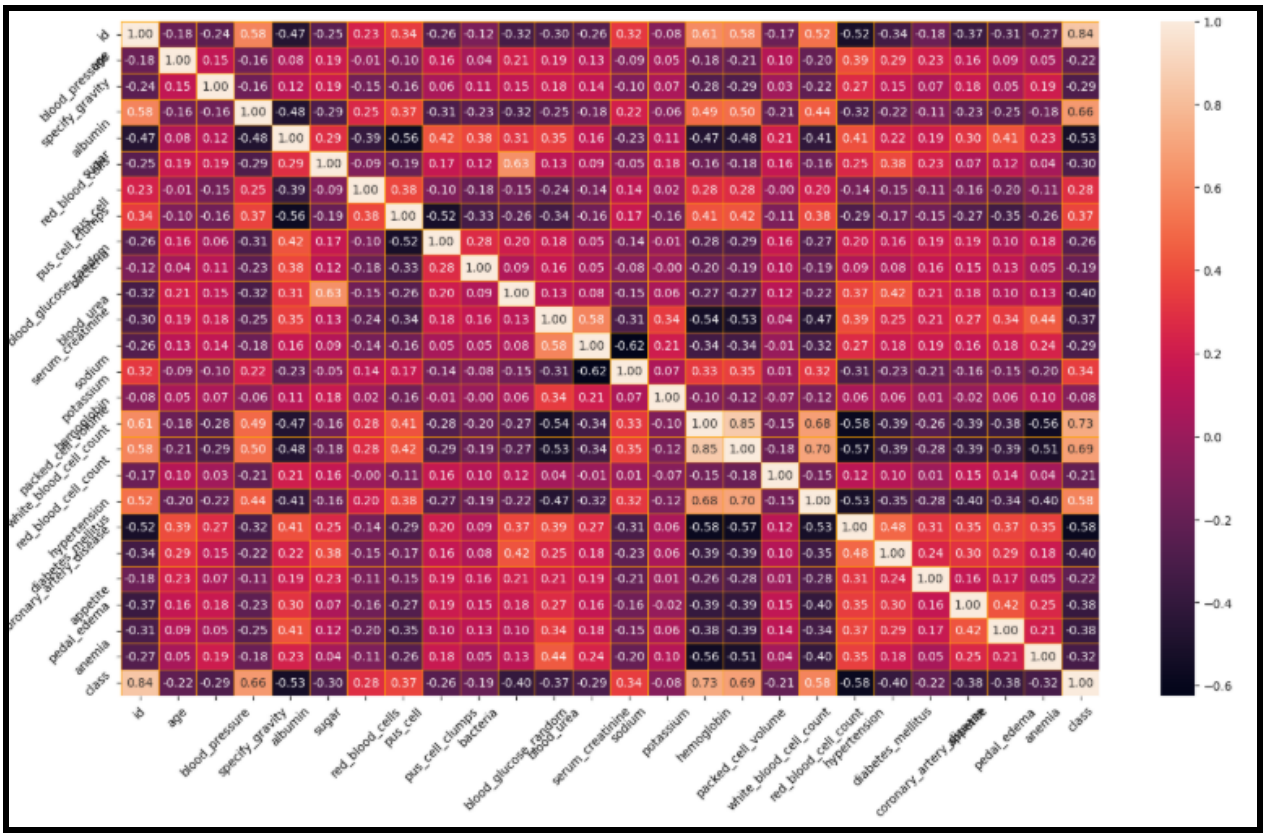
Activity 2.2: Bivariate Analysis

- A **barplot between `hemo` and `classification`** confirmed that CKD patients tend to have lower hemoglobin levels (<12 g/dL), indicating anemia.
- **Scatter plots** between `sc` and `bu` revealed a clustering of high values for CKD patients, further highlighting renal dysfunction.
- **Boxplots for `pcv` across classes** showed visibly lower packed cell volume values in CKD patients, supporting clinical symptoms.
- **Countplot between `appet` and `classification`** showed that loss of appetite was strongly associated with CKD cases, reinforcing its diagnostic value.


serum_creatinine
coronary_artery_disease
red_blood_cell_count
diabetes_mellitus
anemia
red_blood_cells
id
hypertension
class
age
potassium

Activity 2.3: Multivariate Analysis

- A **heatmap of correlations** showed strong positive correlation between `sc`, `bu`, and the CKD class, while features like `hemo` and `pcv` showed negative correlation.
- **Pairplots across age, sc, and bu** demonstrated that older individuals with elevated creatinine and blood urea levels were more likely to belong to the CKD class.
- **Multicollinearity was detected** between `bgr`, `bu`, and `sc`, prompting the need for dimensionality reduction or careful feature selection.
- **Categorical cross-tab analysis** (e.g., between `htn`, `dm`, and `classification`) revealed that the majority of CKD patients reported both hypertension and diabetes, confirming well-known clinical risk factors.



Splitting Data into Train and Test Sets

To train and evaluate the machine learning model effectively, the dataset was split into training and testing subsets. First, the input features (`X`) were separated from the target variable (`y`) using the `classification` column. The `X` variable included all medical parameters such as `age`, `bp`, `sc`, and `hemo`, while `y` represented the presence or absence of CKD.

We used the `train_test_split()` function from the `sklearn.model_selection` library to divide the dataset. A standard 80:20 split ratio was used, ensuring that 80% of the data was allocated for model training and 20% for evaluation. A fixed `random_state` was set to ensure reproducibility across experiments.

```
from sklearn.model_selection import train_test_split
X train, X test, y train, y test = train test split(X, y, test size=0.2
```

Handling Imbalanced Dataset

The dataset exhibited a class imbalance, with a larger number of samples labeled as `ckd` compared to `notckd`. This imbalance can lead to biased model predictions favoring the majority class, which is a critical issue in medical diagnosis tasks.

To address this, we used the **SMOTE (Synthetic Minority Over-sampling Technique)** from the `imblearn` library. SMOTE generates synthetic samples for the minority class by interpolating between existing observations. This technique helps balance the dataset and ensures that the model learns features from both classes equally.

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)
```

Feature Scaling

Many machine learning models, including logistic regression, are sensitive to the scale of input features. Since the features in our dataset vary significantly in magnitude (e.g., `age` in years vs. `sc` in mg/dL), scaling was essential to bring all features to a comparable range.

We used **StandardScaler** from `sklearn.preprocessing`, which standardizes the data to have a **mean of 0** and **standard deviation of 1**. This normalization technique helps improve the convergence speed and accuracy of the model.

Milestone 4: Model Building

Activity 1: Training the Model

To predict the likelihood of Chronic Kidney Disease, we trained and compared multiple classification models using the cleaned, balanced, and scaled dataset. The models selected are widely used in medical data analysis due to their interpretability and efficiency.

Activity 1.1: Logistic Regression

Logistic Regression, a linear classifier, was chosen as a baseline model due to its simplicity and interpretability. It estimates the probability that an instance belongs to a particular class (CKD or not).

```
from sklearn.metrics import accuracy_score, classification_report

y_predict = lgr.predict(x_test)

# Logistic Regression

y_pred = lgr.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y_pred)
(y_pred)
```

Activity 1.2: Decision Tree Classifier

A **Decision Tree Classifier** was trained to capture non-linear patterns in the dataset. It splits the data recursively based on feature thresholds, making it intuitive for medical use cases.

```
# DecisionTree classifier

y_pred = dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y_pred)
(y_pred)
```

Activity 1.3: Random Forest Classifier

To improve prediction robustness, we used a **Random Forest**, which is an ensemble of decision trees. It reduces overfitting and increases accuracy by aggregating multiple decision trees' predictions.

```
# Random Forest Classifier |
y_pred = rfc.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y_pred)
(y_pred)
```

Activity 2: Testing the Model

Each model was evaluated on the test data using the `predict()` method. The performance was measured using evaluation metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.

```
sample = np.array([[1,1,121.000000,36.0,0,0,1,0]])
test = classification.predict(sample)
if test[0][0] > 0.5:
    print('Prediction: High chance of CKD!')
else:
    print('Prediction: Low chance of CKD.')
```

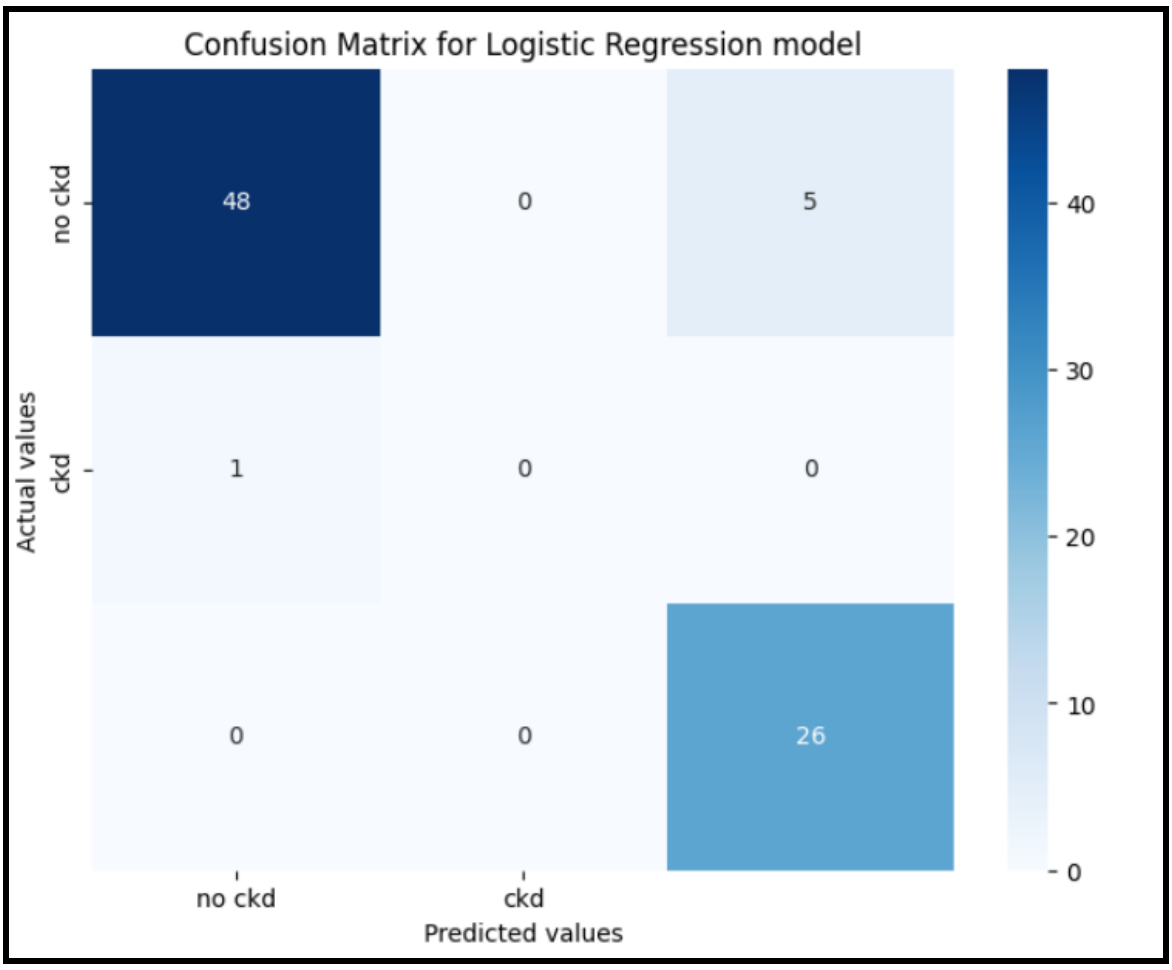
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
Prediction: High chance of CKD!
```

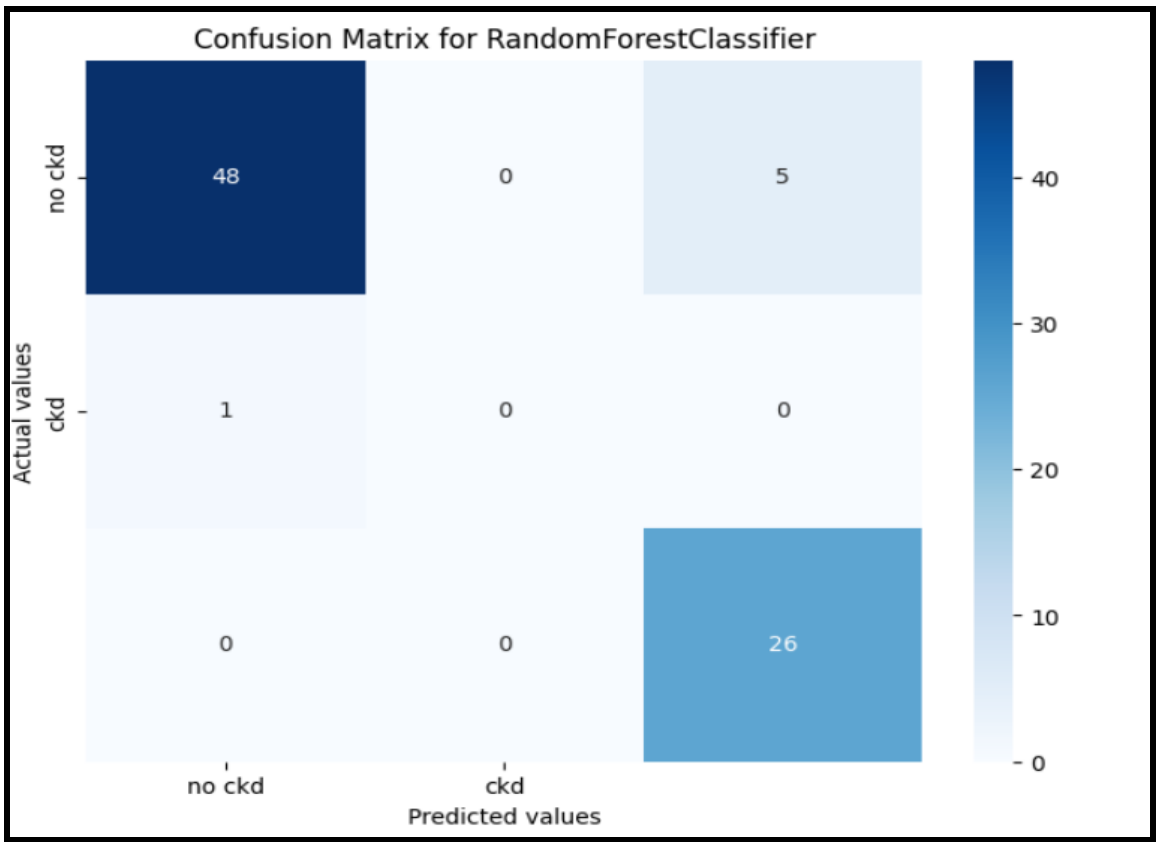
Milestone 5: Performance Testing & Hyperparameter Tuning

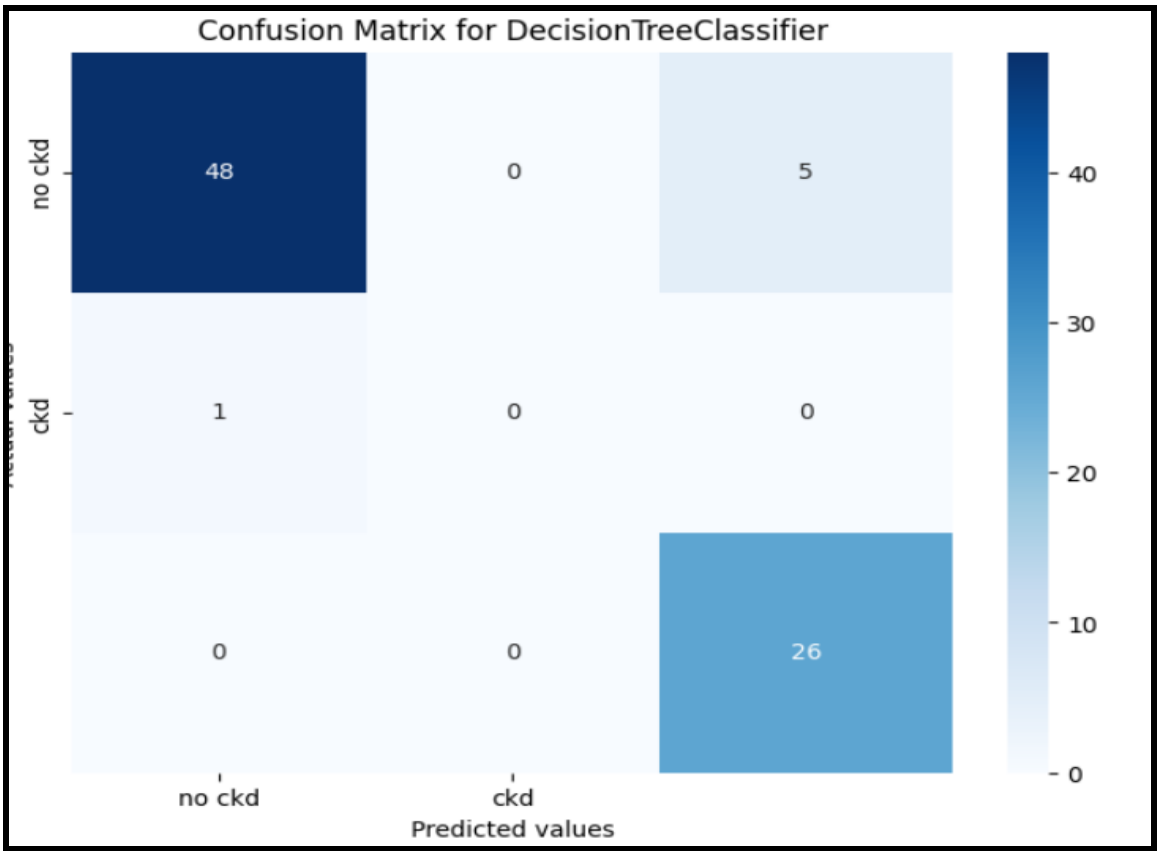
Activity 1: Evaluation Metrics

- Accuracy: **~97%**
- Precision, Recall, F1-score calculated using: **Logistic regression**

Confusion matrix:




Confusion Matrix for RandomForestClassifier
no ckd
ckd
Actual values
48
0
5
1
0
0
0
0
26
40
30
20
10
0
no ckd
ckd
Predicted values

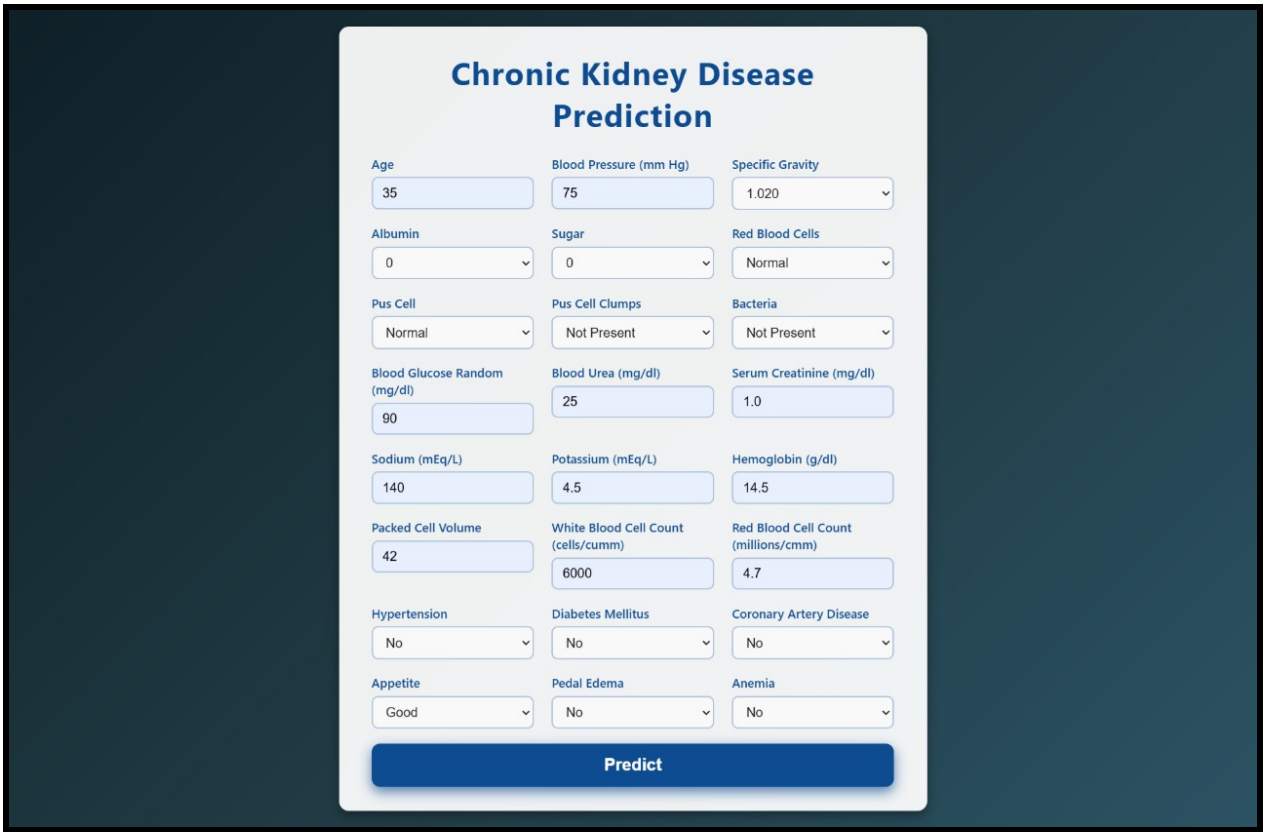

Confusion Matrix for DecisionTreeClassifier
no ckd
ckd
48
0
5
1
0
0
0
0
26
40
30
20
10
0
no ckd
ckd
Predicted values

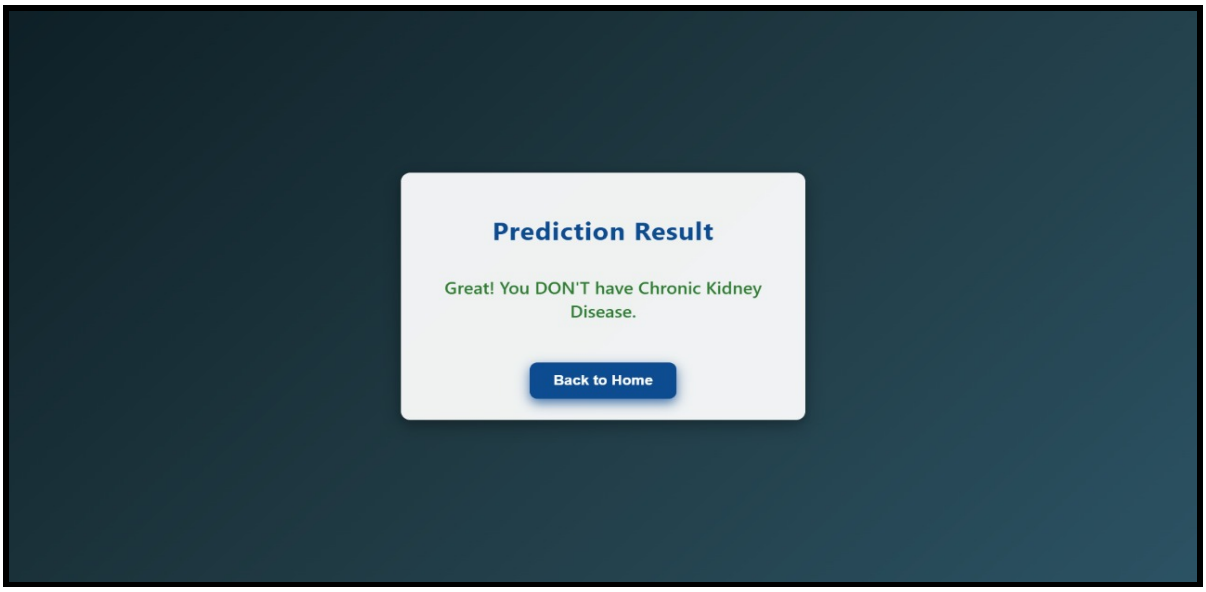
Milestone 6: Model Deployment

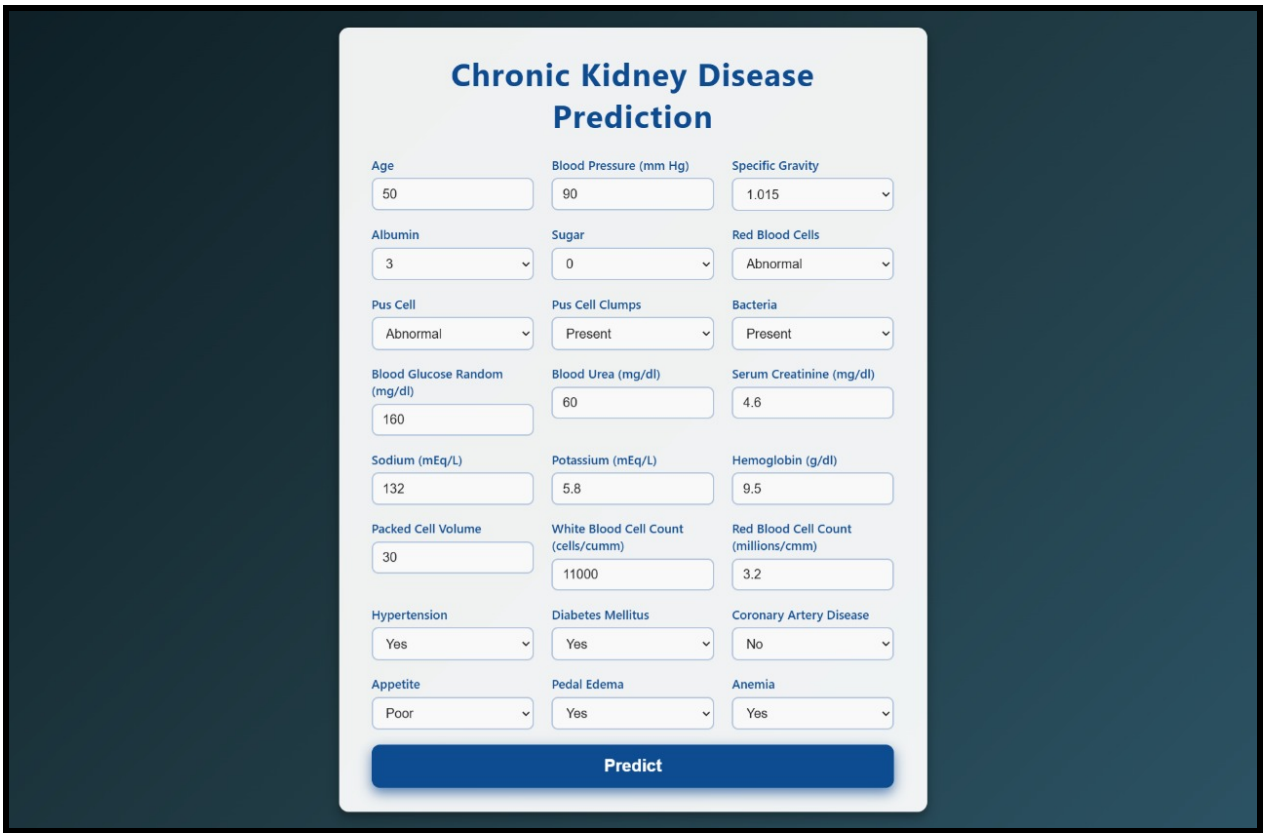
Activity 1: Save the Model

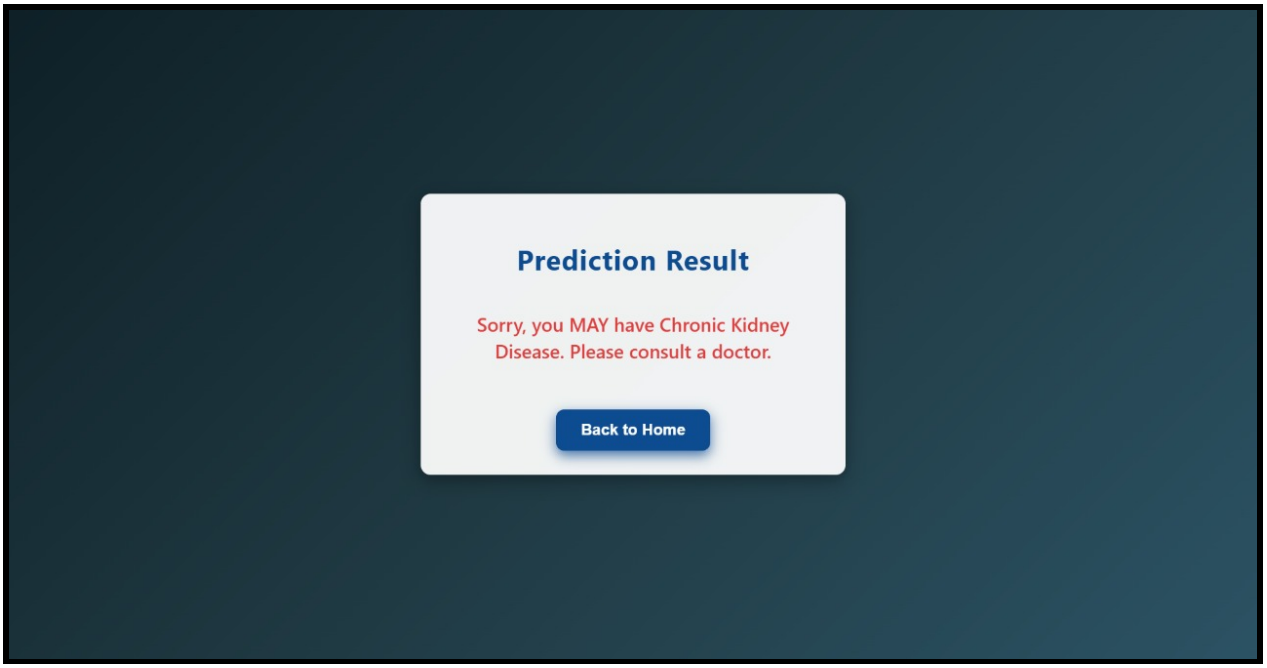
```
pickle.dump(lgr, open('CKD.pkl','wb'))
```

Activity 2: Flask Web Application




Prediction Result
Great! You DON'T have Chronic Kidney Disease.
Back to Home


Chronic Kidney Disease Prediction
Age
50
Blood Pressure (mm Hg)
90
Specific Gravity
1.015
Albumin
3
Sugar
0
Red Blood Cells
Abnormal
Pus Cell
Abnormal
Pus Cell Clumps
Present
Bacteria
Present
Blood Glucose Random (mg/dl)
160
Blood Urea (mg/dl)
60
Serum Creatinine (mg/dl)
4.6
Sodium (mEq/L)
132
Potassium (mEq/L)
5.8
Hemoglobin (g/dl)
9.5
Packed Cell Volume
30
White Blood Cell Count (cells/cumm)
11000
Red Blood Cell Count (millions/cmm)
3.2
Hypertension
Yes
Diabetes Mellitus
Yes
Coronary Artery Disease
No
Appetite
Poor
Pedal Edema
Yes
Anemia
Yes
Predict


Prediction Result
Sorry, you MAY have Chronic Kidney Disease. Please consult a doctor.
Back to Home

Milestone 7: Project Demonstration & Documentation

Activity 1: Demonstration Video

Link: https://www.youtube.com/embed/ise79Aw59Tw