**Final Project Report**

|  |  |
| --- | --- |
| Date | 12-JULY-2024 |
| Team ID | SWTID1720438678 |
| Project Name | Early prediction of chronic kidney disease |

**1) Introduction**

**1.1. Project Overview**

Chronic Kidney Disease (CKD) is a silent killer often diagnosed at advanced stages. This project aims to develop a machine learning model capable of accurately predicting CKD risk in its early stages using patient data. By analyzing factors like age, blood pressure, blood glucose, and kidney function indicators, the model will identify individuals at high risk. Early detection empowers patients and healthcare providers to take preventive measures, slowing disease progression and improving overall outcomes. This research contributes to better CKD management and potentially reduces the burden of kidney failure.

**1.2. Objectives**

**Project Objectives**

1. **Develop a predictive model:** Construct a machine learning model capable of accurately predicting the risk of developing CKD based on patient data.
2. **Identify key risk factors:** Determine the most significant factors contributing to CKD development through feature importance analysis.
3. **Improve early detection:** Enhance the early identification of individuals at high risk for CKD to enable timely interventions.
4. **Optimize healthcare resource allocation:** Provide a tool to prioritize screening and preventive measures for high-risk populations.
5. **Enhance patient outcomes:** Contribute to better CKD management and reduce the progression of the disease to end-stage renal disease.

**2) Project Initialization and Planning Phase**

**2.1 Define Problem Statement**

**Early detection of Chronic Kidney Disease (CKD) remains a significant challenge due to its asymptomatic nature and the lack of effective screening methods.** The delayed diagnosis of CKD often leads to irreversible kidney damage and increased morbidity and mortality. This project aims to address this critical issue by developing a machine learning model capable of accurately predicting CKD risk at an early stage, enabling timely interventions and improved patient outcomes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem**  **Statement (PS)** | **I am**  **(Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A doctor | Predict whether the person has ckd | I am unable to determine it accurately. | CKD often progresses silently without noticeable symptoms in its early stages | Frustrated |
| PS-2 | Patient | Treat my chronic kidney disease | I am unable to  get cured completely | Because of get it diagnosed in later stages. | Hopeless |

**2.2 Project Proposal (Proposed Solution)**

Chronic Kidney Disease (CKD) is a silent killer often diagnosed late. This project aims to develop a machine learning model to predict CKD risk early. By analyzing patient data, we seek to identify individuals at high risk for CKD. Early detection allows for timely interventions, preventing disease progression and improving patient outcomes. The model will use machine learning algorithms to analyze data, including demographics, medical history, and lab results. We expect to identify key risk factors and build a predictive model to be used by healthcare providers. Successful implementation can significantly impact CKD management and reduce the disease burden.

|  |  |
| --- | --- |
| **Project Overview** | |
| Objective | Develop models that can identify individuals at high risk of developing chronic diseases before they experience any symptoms. |
| Scope | This project aims to build a machine learning model that predicts the risk of chronic disease development. The model will focus on a specific disease (or related group) and use various data points to make predictions. The project scope excludes diagnosis, treatment implementation, and EHR integration, but may include an optional deployment tool for healthcare settings, adhering to data privacy regulations. |
| **Problem Statement** | |
| Description | |  | | --- | | The project will focus on building a model to assess an individual's risk of developing a specific chronic disease (or a set of related diseases). The model will use various data points to make predictions. | |
| Impact | |  | | --- | | Early prediction of chronic diseases can significantly improve patient outcomes by enabling early intervention and preventative measures. This can lead to better quality of life, reduced healthcare costs, and a potentially lighter burden on healthcare systems. | |
| **Proposed Solution** | |
| Approach | |  | | --- | | A machine learning model will be developed and trained on a dataset of relevant patient information. The project will likely involve data pre-processing, feature engineering, model selection, training, and evaluation | |
| Key Features | |  | | --- | | The key feature of this solution is the ability to predict chronic disease risk early, allowing for preventative action. The model may also be interpretable, providing insights into the factors that influence its predictions. | |

**Resource Requirements**

|  |  |  |
| --- | --- | --- |
| **Resource Type** | **Description** | **Specification/Allocation** |
| **Hardware** | | |
| Computing Resources | CPU/GPU specifications, number of cores | e.g., 2 x NVIDIA V100 GPUs |
| Memory | RAM specifications | e.g., 8 GB |
| Storage | Disk space for data, models, and logs | e.g., 1 TB SSD |
| **Software** | | |
| Frameworks | Python frameworks | e.g., Flask |
| Libraries | Additional libraries | e.g., scikit-learn, pandas, numpy |
| Development Environment | IDE, version control | e.g., Jupyter Notebook, Git |
| **Data** | | |
| Data | Source, size, format | e.g., Kaggle dataset |

**2.3 Initial Project Planning**

**Product Backlog, Sprint Schedule, and Estimation**

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** | **Sprint Start Date** | **Sprint End Date (Planned)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Project Initialization and Planning Phase | CKD-10 | Define Problem Statement | 1 | Low | Sanjay | 17/6/24 | 24/6/24 |
| Sprint-1 | Project Initialization and Planning Phase | CKD-11 | Project Proposal (Proposed Solution) | 1 | Low | Anin | 17/6/24 | 24/6/24 |
| Sprint-1 | Project Initialization and Planning Phase | CKD-12 | Initial Project Planning | 1 | Low | Allen | 17/6/24 | 24/6/24 |
| Sprint-2 | Data Collection and Preprocessing Phase | CKD-14 | Data Collection Plan and Raw Data Sources | 2 | Medium | kanishk | 24/6/24 | 1/7/24 |
| Sprint-2 | Data Collection and Preprocessing Phase | CKD-15 | Data Quality Report | 2 | Low | Anin | 24/6/24 | 1/7/24 |
| Sprint-2 | Data Collection and Preprocessing Phase | CKD-17 | Data Exploration and Preprocessing | 2 | Moderate | Kanishk | 24/6/24 | 1/7/24 |
| Sprint-3 | Model Development Phase | CKD-18 | Feature Selection Report | 1 | Low | Allen | 1/7/24 | 8/7/24 |
| Sprint-3 | Model Development Phase | CKD-19 | Initial Model Training Code, Model Validation and Evaluation Report | 2 | High | kanishk | 1/7/24 | 8/7/24 |
| Sprint-3 | Model Development Phase | CKD-20 | Model Selection Report | 2 | High | Anin | 1/7/24 | 8/7/24 |
| Sprint-3 | Model Optimization and Tuning Phase | CKD-22 | Model Optimization and Tuning Phase | 1 | High | Sanjay | 1/7/24 | 8/7/24 |

**3) Data Collection and Preprocessing Phase**

**3.1 Data Collection Plan and Raw Data Sources Identified**

**Data Collection Plan**

|  |  |
| --- | --- |
| **Section** | **Description** |
| Project Overview | |  |  |  | | --- | --- | --- | | |  | | --- | | This project aims to develop a machine learning model to predict the likelihood of chronic disease onset. The model will target a specific disease or a group of related diseases and use various data points to assess an individual's risk. | |  | | |  | |
| Data Collection Plan | |  | | --- | | The data for this project will be collected from multiple sources. The primary source will likely be Electronic Health Records (EHR) systems, containing valuable details like demographics, medical history, and lab results. Additionally, depending on the chosen disease, anonymized data from public health databases or wearable devices (with user consent) might also be incorporated. | |
| Raw Data Sources Identified | Csv file (47 kb) from skillwallet platform. |

**Raw Data Sources Template**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source Name** | **Description** | **Location/URL** | **Format** | **Size** | **Access Permissions** |
| Dataset 1 | A csv file that consist of data collected from about 400 people that may or may not have ckd. | https://drive.google.com/file/d/1mPl4yaTKuKZ3017YfYC19Ni7Y964eCNI/view | CSV | 47kb | Public |

**3.2 Data Quality Report**

The dataset exhibits significant quality issues impacting its reliability. Missing data is prevalent across multiple variables, hindering comprehensive analysis. Inconsistent formatting complicates data cleaning and preprocessing efforts. Data inconsistencies, such as variations in date formats and numerical representations, further reduce data integrity. These challenges necessitate robust data cleaning and imputation techniques to ensure data quality and validity for subsequent modeling. Addressing these issues is crucial to derive meaningful insights from the dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Source** | **Data Quality Issue** | **Severity** | **Resolution Plan** |
| Dataset | Missing values | Moderate | Imputation , Deletion |
| Dataset | Inconsistant formating | Low | Standardization , Data transformation |

**3.3 Data Exploration and Preprocessing**

|  |  |
| --- | --- |
| **Section** | **Description** |
| Data Overview | **Structure:** The data is organized in rows and columns, with column headers like "normal," "notpresent," "present," and seemingly unique patient identifiers in the first column.  **Dimensions:** It's difficult to determine the exact number of rows and columns from the limited view, but there are likely many rows representing individual patients and potentially dozens of columns containing various medical data points.  **Content:** The data includes a mix of numbers (potentially representing values for medical tests or measurements) and letters indicating presence or absence of certain conditions. |
| Univariate Analysis | **Univariate analysis** is the simplest form of data analysis where you examine each variable individually. It's about describing and summarizing a single variable without considering its relationship with other variables. |
| Bivariate Analysis | **Bivariate analysis** is a statistical method used to investigate the relationship between two variables. It helps to understand how changes in one variable are associated with changes in another. |
| Multivariate Analysis | **Multivariate analysis** involves the simultaneous analysis of multiple variables to understand their relationships and interdependencies. It’s a powerful technique for uncovering complex patterns and structures within data. |
| Outliers and Anomalies | **Outliers** and **anomalies** are data points that deviate significantly from the overall pattern of a dataset. While often used interchangeably, there are subtle differences between them. |
| **Data Preprocessing Code Screenshots** | |
| Loading Data |  |
| Handling Missing Data |  |
| Data Transformation |  |
| Feature Engineering |  |
| Save Processed Data |  |

**4) Model Development Phase**

**4.1 Feature Selection Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Description** | **Selected (Yes/No)** | **Reasoning** |
| Feature 1 | initial features in the dataset | Yes | Age, Blood Pressure, Specific Gravity, Albumin, Sugar, etc.  Derived or calculated features such as Packed Cell Volume, White Blood Cell Count, and Red Blood Cell Count. |
| Feature 2 | preprocessing steps | Yes | **Removing Columns**: You dropped the 'id' column as it was presumably irrelevant to the analysis.  **Renaming and Corrections**: You renamed columns for clarity and corrected mislabeled or inconsistent values in classification, diabetes\_mellitus, and coronary\_artery\_disease.  **Handling Missing Values**: You filled missing values in numerical columns with the mean and categorical columns with the mode, which is crucial for maintaining data integrity and ensuring the model functions correctly. |
| Feature 3 | Feature Transformation | Yes | **Encoding Categorical Features**: You converted categorical variables into a format suitable for modeling using label encoding. This includes transforming non-numeric features into numeric codes, which is essential for logistic regression.  **Numerical Corrections**: Converted strings in numerical columns like 'packed\_cell\_volume', 'white\_blood\_cell\_count', and 'red\_blood\_cell\_count' to numeric types, handling errors with coercion. |

**4.2 Model Selection Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Description** | **Hyperparameters** | **Performance Metric (e.g., Accuracy, F1 Score)** |
| **Logistic Regression** | Logistic Regression:  Strengths: Simple to interpret, computationally efficient, works well for linearly separable data.  Weaknesses: Might not capture complex relationships between features and the target variable. Can suffer from overfitting if not regularized properly. | random\_state=2  max\_iter=1000 | Accuracy: 0.9875  F1 Score: 0.98  Precision: 1  Recall: 0.98 |
| **Random forest** | Random Forest:  Strengths: Combines multiple decision trees to improve accuracy and reduce variance. Handles missing data and irrelevant features well.  Weaknesses: Can be a "black box" model, making interpretation more challenging. May require more computational resources for training compared to simpler models | n­\_estimators  max\_depth  criterion  random\_state | Accuracy: 0.9625  F1 Score: 0.97  Precision: 1  Recall: 0.94 |
| **Decision tree** | A decision tree is a supervised machine learning algorithm used for classification and regression. It resembles a flowchart with internal nodes representing decisions based on data attributes, branches representing possible outcomes, and leaf nodes representing final predictions. Decision trees are popular for their interpretability but can be prone to overfitting. Techniques like pruning help to mitigate this issue. | criterion='entropy'  random\_state=0 | Accuracy: 0.95  F1 Score: 0.93  Precision: 0.93  Recall: 0.93 |
| **SVM** | SVM is a supervised machine learning algorithm used for classification and regression. It finds the optimal hyperplane to separate data points into different classes. SVMs excel in high-dimensional spaces and are effective for complex patterns. Kernel trick allows SVMs to handle non-linear data by mapping it into higher dimensions. |  | Accuracy: 0.85  F1 Score: 0.83  Precision: 0.72  Recall: 0.97 |

**4.3 Initial Model Training Code, Model Validation and Evaluation Report**

**1. Logistic Regression**

**A screenshot of a computer code

Description automatically generated**

**2. Decision Tree**

**A screen shot of a computer code

Description automatically generated**

**3. Random Forest**

**A close-up of a computer code

Description automatically generated**

**4.SVM**

**A screenshot of a computer

Description automatically generated**

**Model Validation and Evaluation Report:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Classification Report** | **Accuracy** | **Confusion Matrix** |
| Logistic regression | A screenshot of a number of numbers  Description automatically generated | 0.9875 | A black and white text  Description automatically generated |
| Random forest | A screenshot of a number of numbers  Description automatically generated | 0.96 | A black and orange text  Description automatically generated |
| Decision tree | A screenshot of a computer screen  Description automatically generated | 0.95 | A number on a white background  Description automatically generated |
| SVM | A number of numbers in a row  Description automatically generated with medium confidence | 0.85 | A black and white text  Description automatically generated |

**5) Model Optimization and Tuning Phase**

**5.1 Hyperparameter Tuning Documentation**

**Given the model's exceptional accuracy of 99%, a decision was made to forego hyperparameter tuning.** This decision was based on the assumption that further optimization is unlikely to yield significant improvements. The model's performance on the validation set strongly suggests that it has effectively captured the underlying patterns in the data. However, it is acknowledged that in certain critical applications, even marginal gains in performance can be valuable, and hyperparameter tuning might be revisited in the future if the model's performance deteriorates in real-world conditions.

### **6) Results**

#### **6.1. Output Screenshots**



A screenshot of a medical form

Description automatically generated

### **7) Advantages and Disadvantages**

#### **7.1. Advantages**

**Advantages of early prediction of chronic kidney disease system**

Early prediction of CKD using machine learning offers several significant advantages:

* **Improved patient outcomes:** Early detection allows for timely interventions, slowing disease progression and preventing irreversible kidney damage.
* **Enhanced quality of life:** Patients can manage lifestyle factors and adhere to treatment plans to maintain kidney health.
* **Reduced healthcare costs:** Early intervention can prevent costly dialysis or transplantation.
* **Optimized resource allocation:** Identifying high-risk individuals enables targeted screening and preventive measures.
* **Advanced disease understanding:** Analyzing large datasets can uncover new risk factors and patterns in CKD development.
* **Improved patient compliance:** Personalized risk assessments can motivate patients to adopt healthier habits.
* **Potential for earlier biomarker discovery:** Machine learning can aid in identifying novel biomarkers for CKD.
* **Greater accessibility:** Machine learning models can be integrated into primary care settings for widespread screening.
* **Data-driven decision making:** Provides evidence-based insights for healthcare providers.

#### **7.2. Disadvantages**

#### **Challenges and Limitations**

Despite the promise of machine learning in early CKD prediction, several challenges and limitations exist:

* **Data Quality and Availability:** Access to high-quality, comprehensive, and representative datasets is crucial. Missing data, inconsistencies, and biases can significantly impact model performance.
* **Complex Disease Progression:** CKD is a multifaceted condition influenced by various factors, making it challenging to capture the complete disease spectrum in a model.
* **Generalizability:** Models developed on specific populations might not generalize well to different ethnicities, age groups, or geographical regions.
* **Ethical Considerations:** Ensuring data privacy and security, as well as addressing potential biases in the data, is essential.
* **Clinical Implementation:** Integrating the model into healthcare workflows and ensuring clinician acceptance requires careful planning and implementation strategies.
* **Model Interpretability:** While accuracy is important, understanding the reasons behind predictions is crucial for clinical trust and decision-making.
* **Dynamic Nature of Disease:** CKD progression can vary significantly among individuals, making it difficult to predict with high accuracy.
* **Cost and Resource Constraints:** Developing and maintaining a machine learning model requires substantial computational resources and expertise.

**8) Conclusion**

Early prediction of Chronic Kidney Disease (CKD) using machine learning holds immense potential to revolutionize patient care. By harnessing the power of data and advanced algorithms, we can identify individuals at high risk of developing CKD and implement timely interventions. This approach can significantly improve patient outcomes, reduce healthcare costs, and optimize resource allocation.

While challenges such as data quality, model interpretability, and clinical implementation exist, addressing these issues is crucial for the successful deployment of these models in real-world settings. Continued research and collaboration between clinicians, data scientists, and policymakers are essential to unlock the full potential of machine learning in preventing and managing CKD. Ultimately, this technology can contribute to a healthier population and a more efficient healthcare system.

**9) Future Scope**

The successful implementation of machine learning models for early CKD prediction is just the beginning. Future research should focus on:

* **Incorporating real-time data:** Integrating electronic health records and wearable devices to provide continuous monitoring and dynamic risk assessment.
* **Developing personalized prediction models:** Tailoring models to specific patient populations based on demographics, genetic factors, and lifestyle.
* **Exploring novel biomarkers:** Identifying and incorporating new biomarkers to enhance predictive accuracy.
* **Integrating explainable AI:** Developing models that can provide clear and understandable explanations for their predictions to build trust among clinicians.
* **Longitudinal studies:** Conducting long-term follow-up studies to evaluate the model's performance and impact on patient outcomes.
* **Collaborations:** Fostering partnerships between researchers, clinicians, policymakers, and industry to accelerate progress and real-world implementation.

### **10) Appendix**

#### **10.1. Source Code**

#### **a) Model Training and Testing Code**

import pandas as pd #used for data manipulation

import numpy as np #used for numerical analysis

from collections import Counter as c # return counts of number of classess

import matplotlib.pyplot as plt #used for data Visualization

import seaborn as sns #data visualization library

import missingno as msno #finding missing values

from sklearn.metrics import accuracy\_score, confusion\_matrix #model performance

from sklearn.model\_selection import train\_test\_split #splits data in random train and test array

from sklearn.preprocessing import LabelEncoder #encoding the levels of categorical features

from sklearn. linear\_model import LogisticRegression #Classification ML algorithm

import pickle #Python object hierarchy is converted into a byte stream

data = pd.read\_csv('chronickidneydisease.csv')

data.head()

data.tail()

data.head(10)

# Dropping the 'id' column

data = data.drop(columns=['id'])

# Renaming columns

data.columns = [

'age', 'blood\_pressure', 'specific\_gravity', 'albumin', 'sugar',

'red\_blood\_cells', 'pus\_cell', 'pus\_cell\_clumps', 'bacteria',

'blood\_glucose\_random', 'blood\_urea', 'serum\_creatinine', 'sodium',

'potassium', 'hemoglobin', 'packed\_cell\_volume', 'white\_blood\_cell\_count',

'red\_blood\_cell\_count', 'hypertension', 'diabetes\_mellitus',

'coronary\_artery\_disease', 'appetite', 'pedal\_edema', 'anemia', 'classification'

]

# Display the updated dataframe

data.head()

# Understanding data types of each column

data\_types = data.dtypes

print("Data Types:\n", data\_types)

# Summary statistics for numeric features

numeric\_summary = data.describe()

print("\nSummary Statistics for Numeric Features:\n", numeric\_summary)

# Summary statistics for categorical features

categorical\_summary = data.describe(include=['object'])

print("\nSummary Statistics for Categorical Features:\n", categorical\_summary)

# Checking unique values in the classification column

print("Unique values in 'classification' column:", data['classification'].unique())

# Mapping the 'classification' column to numerical values

# Assuming 'ckd' as 1 and 'notckd' as 0, and handling any other variations

data['classification'] = data['classification'].replace({

'ckd': 1,

'notckd': 0,

'ckd\t': 1, # handling any additional variations

'\tckd': 1

})

# Verifying the mapping

print("Value counts for 'classification' column after mapping:")

print(data['classification'].value\_counts())

# Display the dataframe to check the changes

data.head()

# Identify all object type columns and store them in a set

catcols = set(data.select\_dtypes(include=['object']).columns)

# Define the columns to remove from catcols (non-categorical columns)

non\_categorical\_columns = ['packed\_cell\_volume', 'white\_blood\_cell\_count', 'red\_blood\_cell\_count']

# Remove the non-categorical columns from catcols

for col in non\_categorical\_columns:

if col in catcols:

catcols.remove(col)

# Display the corrected categorical columns

print("Corrected Categorical Columns:", catcols)

# Convert columns to appropriate types

for col in catcols:

data[col] = data[col].astype('category')

# Display the updated dataframe and data types to confirm changes

print("\nUpdated Data Types:\n", data.dtypes)

print("\nUpdated Data:\n", data.head())

print(catcols)

# Identify all numerical columns

numcols = set(data.select\_dtypes(include=['number']).columns)

# Define the columns to remove from numcols (non-numerical columns)

non\_numerical\_columns = ['specific\_gravity', 'albumin', 'sugar']

# Remove the non-numerical columns from numcols

for col in non\_numerical\_columns:

if col in numcols:

numcols.remove(col)

# Print the continuous columns

print("Continuous (Numerical) Columns:", numcols)

additional\_continuous\_columns = ['red\_blood\_cell\_count', 'packed\_cell\_volume', 'white\_blood\_cell\_count']

for col in additional\_continuous\_columns:

numcols.add(col)

# Print the continuous columns

print("Continuous (Numerical) Columns:", numcols)

additional\_categorical\_columns = ['specific\_gravity', 'albumin', 'sugar']

for col in additional\_categorical\_columns:

catcols.add(col)

# Print the corrected lists of columns

print("Continuous (Numerical) Columns:", numcols)

print("Categorical Columns:", catcols)

data['diabetes\_mellitus'] = data['diabetes\_mellitus'].replace({' yes': 'yes', '\tyes': 'yes', '\tno': 'no'})

data['coronary\_artery\_disease'] = data['coronary\_artery\_disease'].replace({'\tno': 'no'})

c(data['coronary\_artery\_disease'])

c(data['diabetes\_mellitus'])

data.isnull().any()

data.isnull().sum()

data['packed\_cell\_volume'] = pd.to\_numeric(data['packed\_cell\_volume'], errors='coerce')

data['white\_blood\_cell\_count'] = pd.to\_numeric(data['white\_blood\_cell\_count'], errors='coerce')

data['red\_blood\_cell\_count'] = pd.to\_numeric(data['red\_blood\_cell\_count'], errors='coerce')

# Fill null values with column means for numerical columns

numerical\_columns = [

'blood\_glucose\_random', 'blood\_pressure', 'blood\_urea', 'hemoglobin',

'packed\_cell\_volume', 'potassium', 'red\_blood\_cell\_count', 'serum\_creatinine',

'sodium', 'white\_blood\_cell\_count'

]

for column in numerical\_columns:

data[column].fillna(data[column].mean(), inplace=True)

# Fill null values with the mode for categorical columns

categorical\_columns = [

'age', 'hypertension', 'pus\_cell\_clumps', 'appetite', 'albumin',

'pus\_cell', 'red\_blood\_cells', 'coronary\_artery\_disease', 'bacteria',

'anemia', 'sugar', 'diabetes\_mellitus', 'pedal\_edema', 'specific\_gravity'

]

for column in categorical\_columns:

data[column].fillna(data[column].mode()[0], inplace=True)

data.isnull().sum()

from sklearn.preprocessing import LabelEncoder

# Apply label encoding on all the categorical columns

label\_encoders = {}

for column in catcols:

le = LabelEncoder()

data[column] = le.fit\_transform(data[column])

label\_encoders[column] = le

# Display the first few rows of the updated dataframe

print(data.head())

import pandas as pd

# Define the columns for the independent variables

selcols = [

'red\_blood\_cells', 'pus\_cell', 'blood\_glucose\_random', 'blood\_urea',

'pedal\_edema', 'anemia', 'diabetes\_mellitus', 'coronary\_artery\_disease'

]

# Define the target column

target\_column = 'classification'

# Create the DataFrame for independent variables

X = pd.DataFrame(data, columns=selcols)

# Create the DataFrame for the dependent variable

y = pd.DataFrame(data, columns=[target\_column])

# Display the shapes of the resulting datasets to confirm the split

print(X.shape)

print(y.shape)

# Split the data into training and test sets

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

# Display the shapes of the resulting datasets to confirm the split

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("y\_train shape:", y\_train.shape)

print("y\_test shape:", y\_test.shape)

from sklearn.linear\_model import LogisticRegression

# Initialize the model

logistic\_model = LogisticRegression(random\_state=2, max\_iter=1000)

# Fit the model with the training data

logistic\_model.fit(X\_train, y\_train.values.ravel())

# Predict the target variable on the test set

y\_pred = logistic\_model.predict(X\_test)

# Print the predictions

print(y\_pred)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix, classification\_report

# Predict the target variable on the test set

y\_pred = logistic\_model.predict(X\_test)

# Evaluate the model's performance

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

print("Confusion Matrix:\n", conf\_matrix)

print("Classification Report:\n", class\_report)

import pickle

# Save the model to a file

with open('logistic\_model.pkl', 'wb') as file:

pickle.dump(logistic\_model, file)

# Save the modified dataset to a CSV file

data.to\_csv('modified\_chronic\_kidney\_disease.csv', index=False)

import numpy as np

import pickle

# Load the model

model\_path = 'logistic\_model.pkl' # Ensure the path is correct

model = pickle.load(open(model\_path, 'rb'))

# Define the features based on the provided values

features = np.array([5.2, 1, 121, 44, 0, 0, 1, 0]).reshape(1, -1)

# Predict the outcome

prediction = model.predict(features)

output = 'CKD' if prediction[0] == 1 else 'Not CKD'

print(output)

#### **b) Flask Code**

from flask import Flask, render\_template, request

import pickle

import numpy as np

app = Flask(\_\_name\_\_)

# Load the model

model = pickle.load(open('logistic\_model.pkl', 'rb'))

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

# Get form data

features = [

float(request.form['red\_blood\_cells']),

float(request.form['pus\_cell']),

float(request.form['blood\_glucose\_random']),

float(request.form['blood\_urea']),

float(request.form['pedal\_edema']),

float(request.form['anemia']),

float(request.form['diabetes\_mellitus']),

float(request.form['coronary\_artery\_disease'])

]

# Convert to numpy array and reshape for prediction

final\_features = np.array(features).reshape(1, -1)

prediction = model.predict(final\_features)

output = 'CKD' if prediction[0] == 1 else 'Not CKD'

return render\_template('index.html', prediction=output)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

#### **c) CSS Code**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.container {

width: 50%;

margin: 0 auto;

text-align: center;

padding: 20px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

margin-top: 50px;

border-radius: 10px;

}

h2 {

color: #333;

}

form div {

margin-bottom: 10px;

}

label {

display: block;

margin-bottom: 5px;

}

input, select {

padding: 10px;

width: calc(100% - 22px); /\* Adjusting for padding and border \*/

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #45a049;

}

**d)** HTML Code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CKD Prediction</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

</head>

<body>

<div class="container">

<h2>Chronic Kidney Disease Prediction</h2>

<form action="/predict" method="post">

<div>

<label for="red\_blood\_cells">Red Blood Cells</label>

<input type="number" id="red\_blood\_cells" name="red\_blood\_cells" step="0.1" required>

</div>

<div>

<label for="pus\_cell">Pus Cell</label>

<select id="pus\_cell" name="pus\_cell" required>

<option value="0">Normal</option>

<option value="1">Abnormal</option>

</select>

</div>

<div>

<label for="blood\_glucose\_random">Blood Glucose Random</label>

<input type="number" id="blood\_glucose\_random" name="blood\_glucose\_random" required>

</div>

<div>

<label for="blood\_urea">Blood Urea</label>

<input type="number" id="blood\_urea" name="blood\_urea" required>

</div>

<div>

<label for="pedal\_edema">Pedal Edema</label>

<select id="pedal\_edema" name="pedal\_edema" required>

<option value="0">No</option>

<option value="1">Yes</option>

</select>

</div>

<div>

<label for="anemia">Anemia</label>

<select id="anemia" name="anemia" required>

<option value="0">No</option>

<option value="1">Yes</option>

</select>

</div>

<div>

<label for="diabetes\_mellitus">Diabetes Mellitus</label>

<select id="diabetes\_mellitus" name="diabetes\_mellitus" required>

<option value="0">No</option>

<option value="1">Yes</option>

</select>

</div>

<div>

<label for="coronary\_artery\_disease">Coronary Artery Disease</label>

<select id="coronary\_artery\_disease" name="coronary\_artery\_disease" required>

<option value="0">No</option>

<option value="1">Yes</option>

</select>

</div>

<button type="submit">Predict</button>

</form>

<div>

{% if prediction %}

<h3>Prediction: {{ prediction }}</h3>

{% endif %}

</div>

</div>

</body>

</html>

#### **10.2. GitHub & Project Demo Link**

Project Demo Link : <https://drive.google.com/file/d/1adVAYjdlEuYSTMjNcnFUKrZi-9SOQriU/view?usp=sharing>

Github Link : <https://github.com/koopatroopa787/smartinternz_assignement.git>