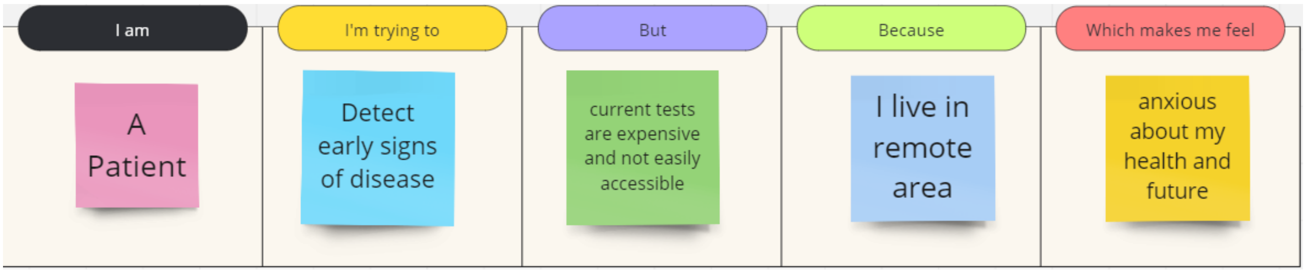
Project Initialization and Planning Phase

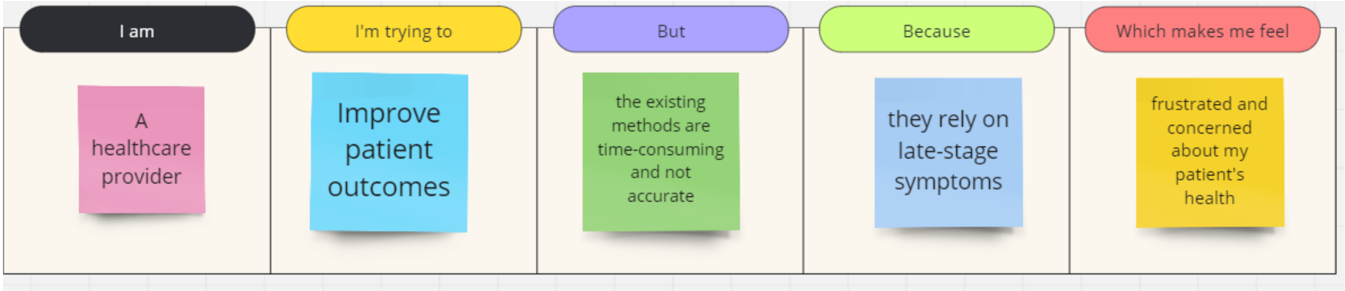
Date	10 July 2024
Team ID	SWTID1721205662
Project Name	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	3 Marks

Define Problem Statements (Customer Problem Statement Template):

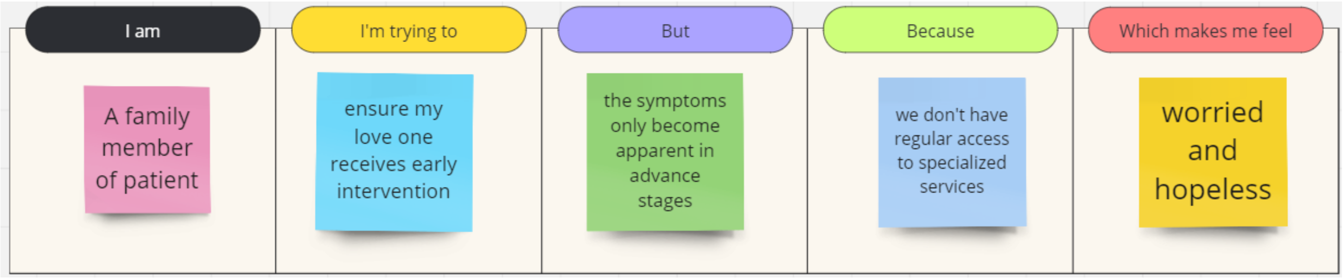
To address the challenges of early prediction and diagnosis of chronic kidney disease (CKD), it is essential to understand the specific issues faced by patients, healthcare providers, and researchers. Developing a problem statement that reflects these challenges helps in creating targeted machine learning solutions that can improve the early detection and management of CKD. This approach aims to enhance patient outcomes, reduce healthcare costs, and provide better tools for healthcare professionals.



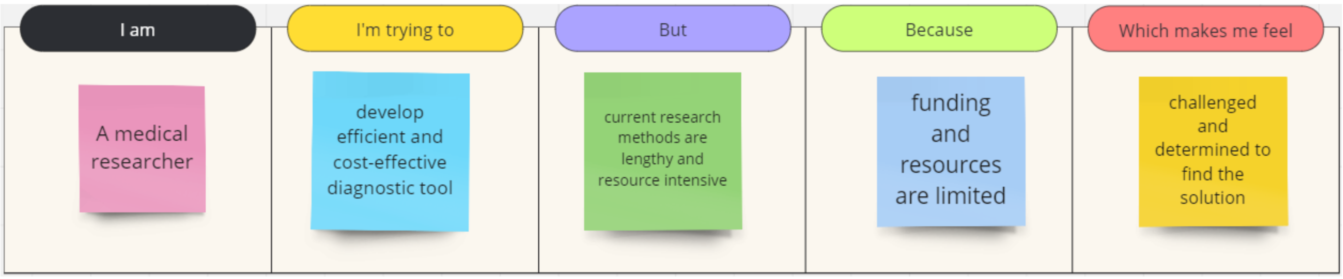
I am	I'm trying to	But	Because	Which makes me feel
A Patient	Detect early signs of disease	current tests are expensive and not easily accessible	I live in remote area	anxious about my health and future



I am	I'm trying to	But	Because	Which makes me feel
A healthcare provider	Improve patient outcomes	the existing methods are time-consuming and not accurate	they rely on late-stage symptoms	frustrated and concerned about my patient's health



I am	I'm trying to	But	Because	Which makes me feel
A family member of patient	ensure my love one receives early intervention	the symptoms only become apparent in advance stages	we don't have regular access to specialized services	worried and hopeless



I am	I'm trying to	But	Because	Which makes me feel
A medical researcher	develop efficient and cost-effective diagnostic tool	current research methods are lengthy and resource intensive	funding and resources are limited	challenged and determined to find the solution

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a patient	detect early signs of chronic kidney disease	the current tests are expensive and not easily accessible	I live in a remote area with limited healthcare facilities	anxious about my health and future
PS-2	a healthcare provider	improve patient outcomes by predicting chronic kidney disease earlier	the existing methods are time-consuming and not very accurate	they rely on late-stage symptoms and extensive lab tests	frustrated and concerned about my patients' well-being

PS-3	a family member of a patient	ensure my loved one receives early intervention for chronic kidney disease	the symptoms are not apparent until the disease has progressed significantly	we don't have regular access to specialized healthcare services	worried and helpless
PS-4	a medical researcher	develop efficient and cost-effective diagnostic tools for chronic kidney disease	current research methods are lengthy and resource-intensive	funding and resources are limited	challenged and determined to find a solution

Project Initialization and Planning Phase

Date	10th July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

The primary objective of this project is to develop a robust machine learning model that can accurately detect chronic kidney disease (CKD) using patient data.

Project Overview	
Objective	Develop a robust machine learning model to accurately detect chronic kidney disease (CKD) using patient data.
Scope	- Data preprocessing and cleaning. - Feature selection and engineering. - Training and evaluating multiple machine learning models. - Selection of the best-performing model. - Deployment in a user-friendly interface.
Problem Statement	
Description	Chronic kidney disease is a significant health issue requiring early detection to prevent severe complications. Current diagnostic methods are time-consuming and require extensive medical expertise.
Impact	- Enable early detection and treatment of CKD. - Reduce the burden on healthcare professionals. - Improve patient outcomes with timely interventions.
Proposed Solution	
Approach	- **Data Preprocessing:** Handle missing values, normalize data, encode categorical variables. - **Feature Selection and Engineering:** Identify relevant features, create new features if necessary. - **Model Training and Evaluation:** Train multiple models, evaluate using metrics like accuracy, precision, recall, F1-score.

	- **Model Selection:** Select and fine-tune the best-performing model. - **Deployment:** Develop and deploy a user-friendly interface
Key Features	- **Accuracy:** High accuracy in detecting CKD. - **Efficiency:** Quick and automated detection. - **User-Friendly Interface:** Easy for healthcare providers to use. - **Scalability:** Can handle large data volumes.

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	44KB	e.g., Kaggle dataset

Initial Project Planning

Date	10 July 2024
Team ID	SWTID1721205662
Project Name	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create a product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story / Task	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	Understanding and Loading Data	Low	Karan S	10/07/2024	12/07/2024
Sprint-1	Data Collection and Preprocessing	Data Cleaning	High	Parth	10/07/2024	12/07/2024
Sprint-1	Data Collection and Preprocessing	EDA	Medium	Karan V	10/07/2024	12/07/2024
Sprint-1	Project Report	Report	Medium	Karan V	10/07/2024	12/07/2024
Sprint-2	Model Development	Training the Model	Medium	Parth	14/07/2024	16/07/2024
Sprint-2	Model Development	Evaluating The Model	Medium	Karan V	14/07/2024	16/07/2024

Sprint	Functional Requirement (Epic)	User Story / Task	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-2	Model Tuning and Testing	Model Tuning	High	Karan S	14/07/2024	16/07/2024
Sprint-2	Model Tuning and Testing	Model Testing	Medium	Karan S	14/07/2024	16/07/2024
Sprint-3	Web Integration and Deployment	Building HTML Templates	Low	Parth	17/07/2024	19/07/2024
Sprint-3	Web Integration and Deployment	Local Deployment	Medium	Karan V	17/07/2024	19/07/2024

Data Collection and Preprocessing Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification Template

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan Template

Section	**Description**
Project Overview	The machine learning project aims to predict the presence of chronic kidney disease based on patient information. Using a dataset with features such as age, blood pressure, specific gravity, albumin, sugar, and various blood test results, the objective is to build a model that accurately classifies the disease status, facilitating early diagnosis and treatment.
Data Collection Plan	• The data is sourced from medical records containing information relevant to chronic kidney disease. • The dataset includes various patient attributes and test results necessary for the prediction model. • Prioritize datasets with diverse demographic and clinical information to ensure model robustness.

Raw Data Sources Identified	The raw data sources for this project include datasets obtained from Kaggle, the popular platform for data science competitions. The provided sample data represents a subset of the collected information, encompassing variables such as Age, BP, Urea and other Kidney Health Related factors.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Kaggle Dataset	The dataset comprises patient details (Age, BP, Urea, Sugar Level etc.) and Chronic Kidney Disease Outcomes.	https://drive.google.com/file/d/1mPl4yaTKuKZ3017YfYC19Ni7Y964eCNI/view?usp=sharing	CSV	42.5 KB	Public

Data Collection and Preprocessing Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	2 Marks

Data Quality Report Template

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

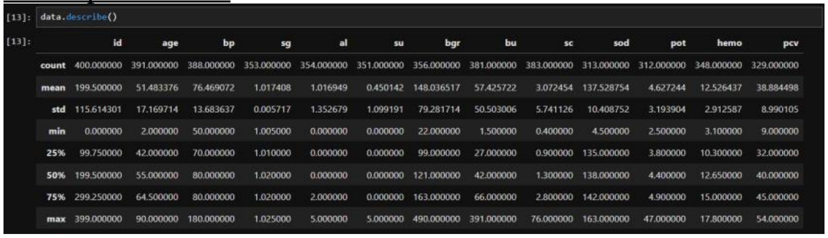
Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Missing values in the 'Age', 'Blood Pressure', 'Specific Gravity', 'Albumin', 'Sugar', 'Red Blood Cells', 'Pus Cell', 'Pus Cell Clumps', 'Bacteria', 'Blood Glucose Random', 'Blood Urea', 'Serum Creatinine', 'Sodium', 'Potassium', 'Hemoglobin', 'Packed Cell Volume', 'White Blood Cell Count', 'Red Blood Cell Count', 'Hypertension', 'Diabetes Mellitus', 'Coronary Artery Disease', 'Appetite', 'Pedal Edema', 'Anemia', 'Classification' columns.	High	Use Mean/Mode Imputation
Kaggle Dataset	Categorical Data in Dataset	Moderate	Label Encoding has to be done in the data.

Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing

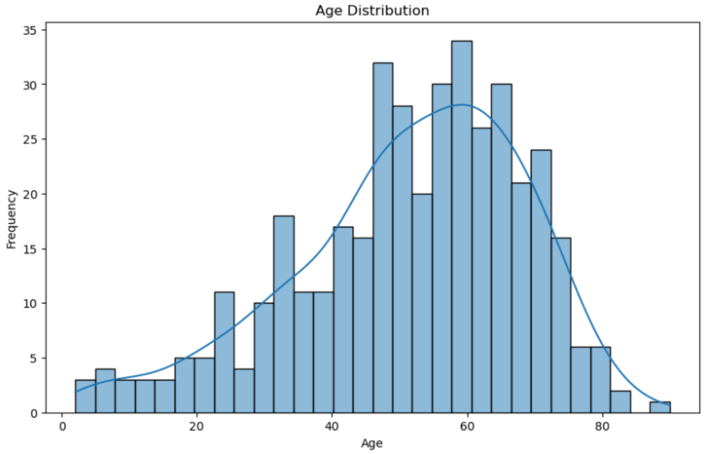
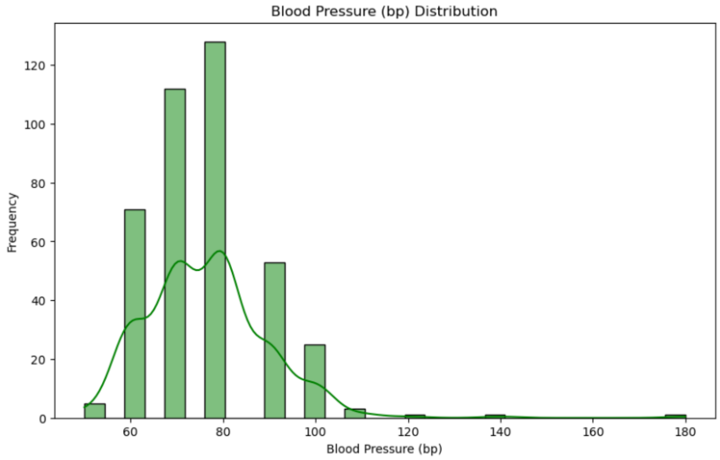
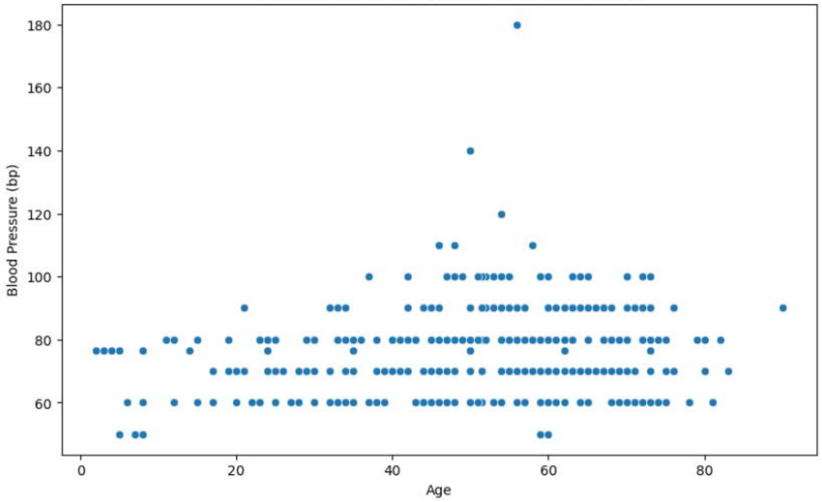
Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

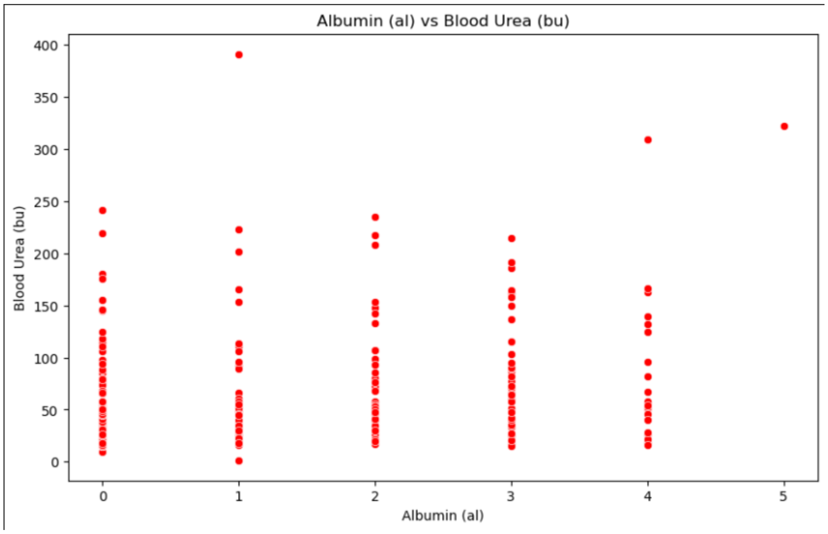
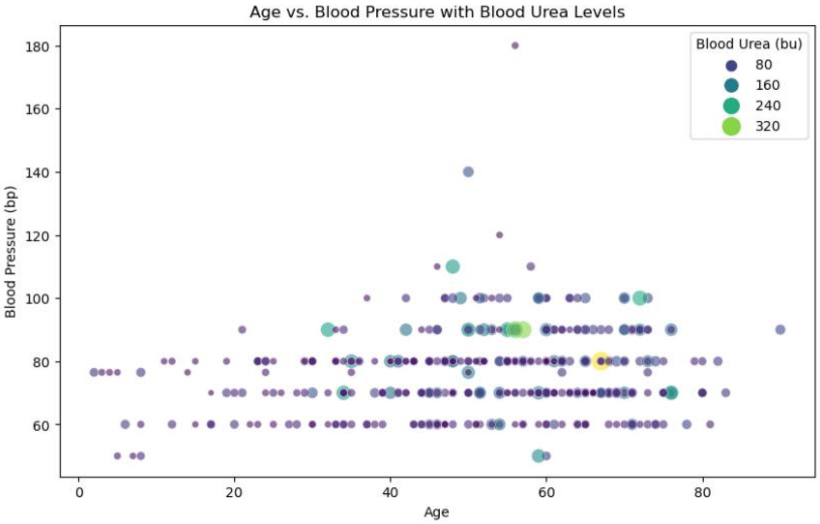
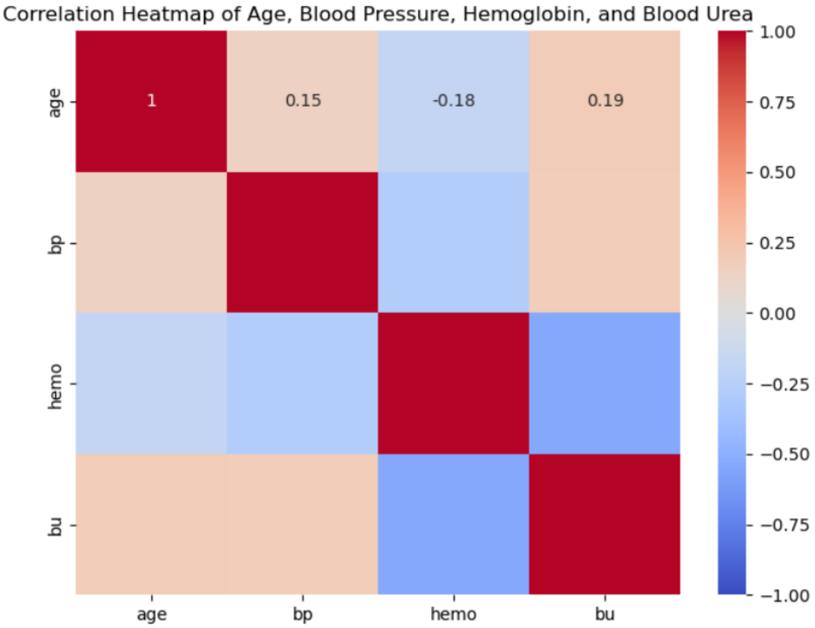
Section	**Description**
Data Overview	Dimension: 400 rows x 26 columns Descriptive Stats: 

```
[13]: data.describe()
```

```
[13]:
              id         age          bp          sg          al          su         bgr         bu          sc         sod         pot        hemo         pcv
count 400.000000  391.000000  388.000000  353.000000  354.000000  351.000000  356.000000  381.000000  383.000000  313.000000  312.000000  348.000000  329.000000
mean  199.500000   51.483376   76.469072    1.017408    1.016949    0.450142  148.036517   57.425722    3.072454  137.528754    4.627244   12.526437   38.884498
std   115.614301   17.169714   13.683637    0.005717    1.352679    1.099191   79.281714   50.503006    5.741126   10.408752    3.193904    2.912587    8.990105
min     0.000000    2.000000   50.000000    1.005000    0.000000    0.000000   22.000000    1.500000    0.400000    4.500000    2.500000    3.100000    9.000000
25%    99.750000   42.000000   70.000000    1.010000    0.000000    0.000000   99.000000   27.000000    0.900000  135.000000    3.800000   10.300000   32.000000
50%   199.500000   55.000000   80.000000    1.020000    0.000000    0.000000  121.000000   42.000000    1.300000  138.000000    4.400000   12.650000   40.000000
75%   299.250000   64.500000   80.000000    1.020000    2.000000    0.000000  163.000000   66.000000    2.800000  142.000000    4.900000   15.000000   45.000000
max   399.000000   90.000000  180.000000    1.025000    5.000000    5.000000  490.000000  391.000000   76.000000  163.000000   47.000000   17.800000   54.000000
```

Univariate Analysis	

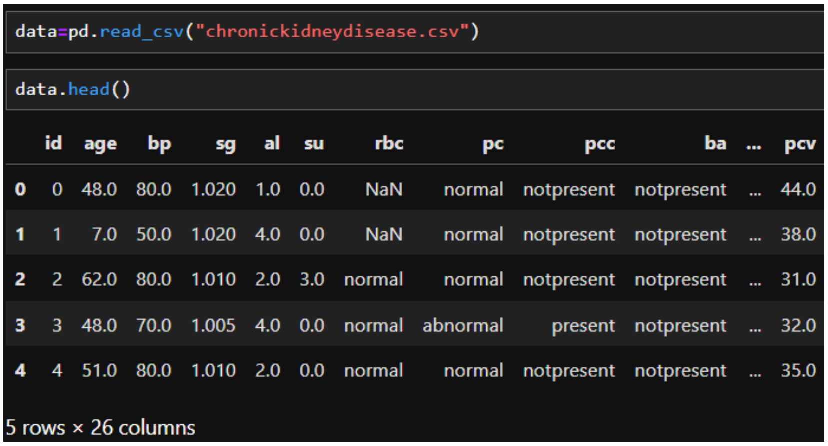
	Age Distribution  Blood Pressure (bp) Distribution 
Bivariate Analysis	Relationship Between Age and Blood Pressure (bp) 

	
Multivariate Analysis	 

Outliers and Anomalies	-

Data Preprocessing Code Screenshots

Loading Data

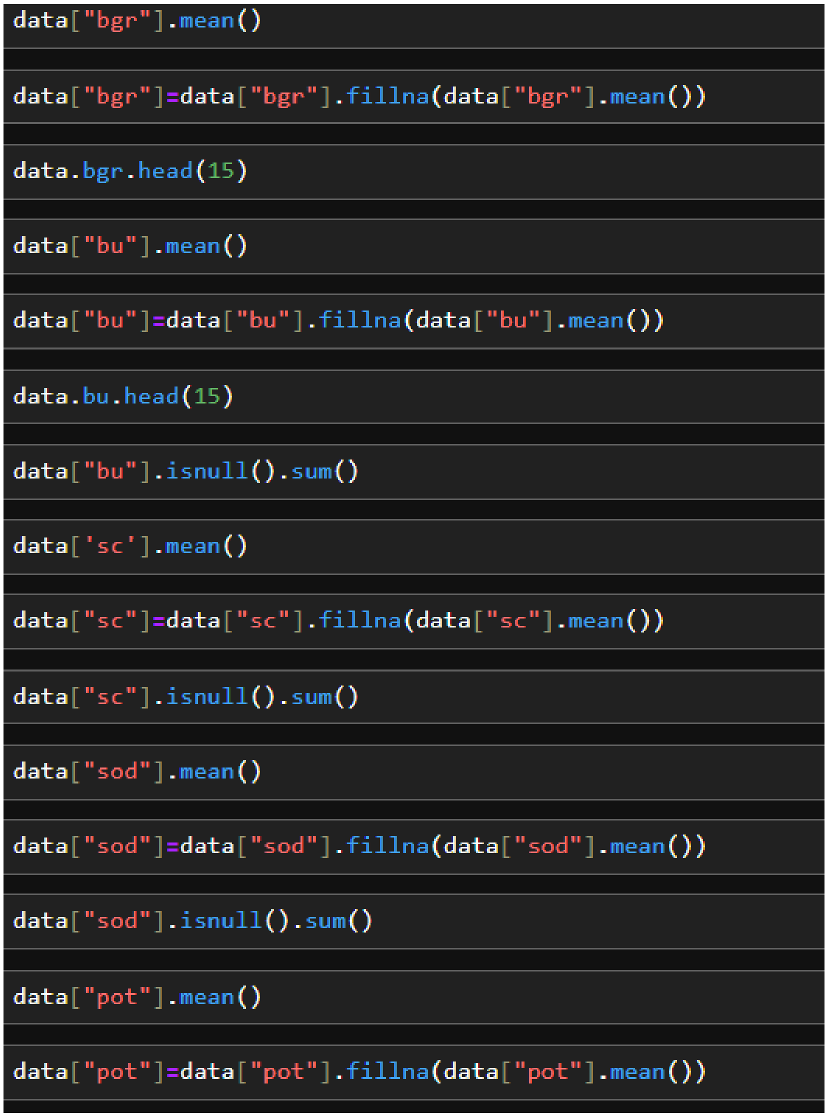


```
data=pd.read_csv("chronickidneydisease.csv")
```

```
data.head()
```

```
   id   age    bp     sg   al   su     rbc        pc         pcc          ba  ...   pcv
0   0  48.0  80.0  1.020  1.0  0.0     NaN    normal  notpresent  notpresent  ...  44.0
1   1   7.0  50.0  1.020  4.0  0.0     NaN    normal  notpresent  notpresent  ...  38.0
2   2  62.0  80.0  1.010  2.0  3.0  normal    normal  notpresent  notpresent  ...  31.0
3   3  48.0  70.0  1.005  4.0  0.0  normal  abnormal     present  notpresent  ...  32.0
4   4  51.0  80.0  1.010  2.0  0.0  normal    normal  notpresent  notpresent  ...  35.0

5 rows × 26 columns
```

Handling Missing Data



```
data["bgr"].mean()
```

```
data["bgr"]=data["bgr"].fillna(data["bgr"].mean())
```

```
data.bgr.head(15)
```

```
data["bu"].mean()
```

```
data["bu"]=data["bu"].fillna(data["bu"].mean())
```

```
data.bu.head(15)
```

```
data["bu"].isnull().sum()
```

```
data['sc'].mean()
```

```
data["sc"]=data["sc"].fillna(data["sc"].mean())
```

```
data["sc"].isnull().sum()
```

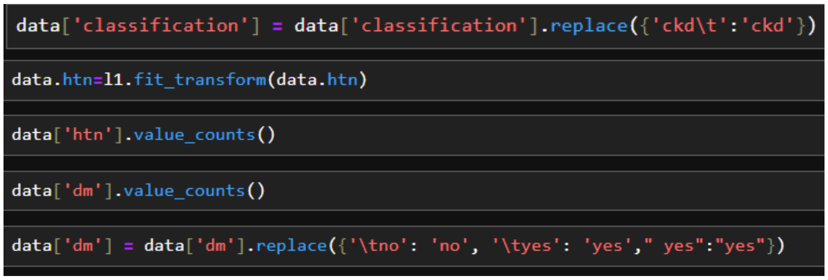
```
data["sod"].mean()
```

```
data["sod"]=data["sod"].fillna(data["sod"].mean())
```

```
data["sod"].isnull().sum()
```

```
data["pot"].mean()
```

```
data["pot"]=data["pot"].fillna(data["pot"].mean())
```

Data Transformation	
	`data['classification'] = data['classification'].replace({'ckd\t':'ckd'})`
	`data.htn=l1.fit_transform(data.htn)`
	`data['htn'].value_counts()`
	`data['dm'].value_counts()`
	`data['dm'] = data['dm'].replace({'\tno': 'no', '\tyes': 'yes'," yes":"yes"})`

Feature Engineering	Attached the codes in final submission.
Save Processed Data	-

Model Development Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	5 Marks

Feature Selection Report Template

In the forthcoming update, each feature will come with a brief description. Users will specify whether they have selected the feature and provide their reasoning. This process will streamline decision-making and improve transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
id	**id**: Identifier for each entry.	**No**	For predicting a chronic disease, id of patient is not relevant.
age	**age**: Age of the patient.	**Yes**	Older age is a risk factor for CKD.
bp	**bp**: Blood pressure	**Yes**	High blood pressure can damage kidneys.

sg	**sg**: Specific gravity of urine.	**Yes**	Measures urine concentration, which can be affected by kidney function.
al	**al**: Albumin level in urine.	**Yes**	High levels in urine can indicate kidney damage
su	**su**: Sugar level in urine.	**Yes**	High sugar levels in urine (glycosuria) can indicate diabetes mellitus, a leading cause of CKD.
ba	**ba**: Bacteria.	**No**	Bacterial infection is not relevant to CKD in any form.
pc	**pc**: Pus cell.	**Yes**	Indicate infection or inflammation in the urinary tract or kidneys
sc	**sc**: Serum creatinine.	**Yes**	Elevated levels are indicators of reduced kidney function.
bgr	**bgr**: Blood glucose random.	**Yes**	High levels can indicate diabetes, a major cause of CKD

bu	**bu**: Blood urea	**Yes**	Elevated levels are indicators of reduced kidney function.

sod	**sod**: Sodium level.	**Yes**	Electrolyte levels are regulated by the kidneys
Pot	**pot**: Potassium level.	**Yes**	Electrolyte levels are regulated by the kidneys
hemo	**hemo**: Hemoglobin level.	**Yes**	Low levels (anemia) are common in CKD.
pcv	**pcv**: Packed cell volume	**Yes**	Can be reduced in CKD.
wc	**wc**: White blood cell count.	**Yes**	Abnormal levels can be a sign of kidney issues.
rc	**rc**: Red blood cell count.	**Yes**	Low count can indicate anemia

htn	**htn**: Hypertension	**Yes**	Both a cause and a complication of CKD.
cad	**cad**: Coronary artery disease	**Yes**	Associated with CKD due to common risk factors.

appet	**appet**: Appetite	**Yes**	Symptoms related to advanced CKD.
pe	**pe**: Pedal edema	**Yes**	Symptoms related to advanced CKD.
ane	**ane**: Anemia	**Yes**	Common in CKD due to reduced erythropoietin production.
classification	**classification**: Classification of the disease	**Yes**	**The target variable which classifies whether the patient suffers from CKD or not.**

Model Development Phase

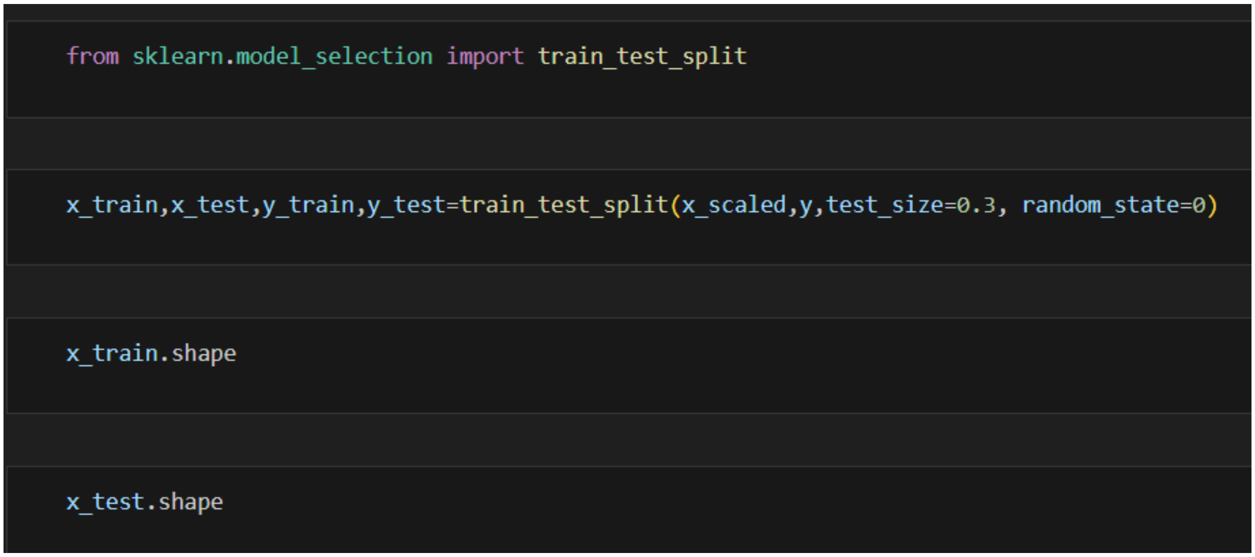
Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

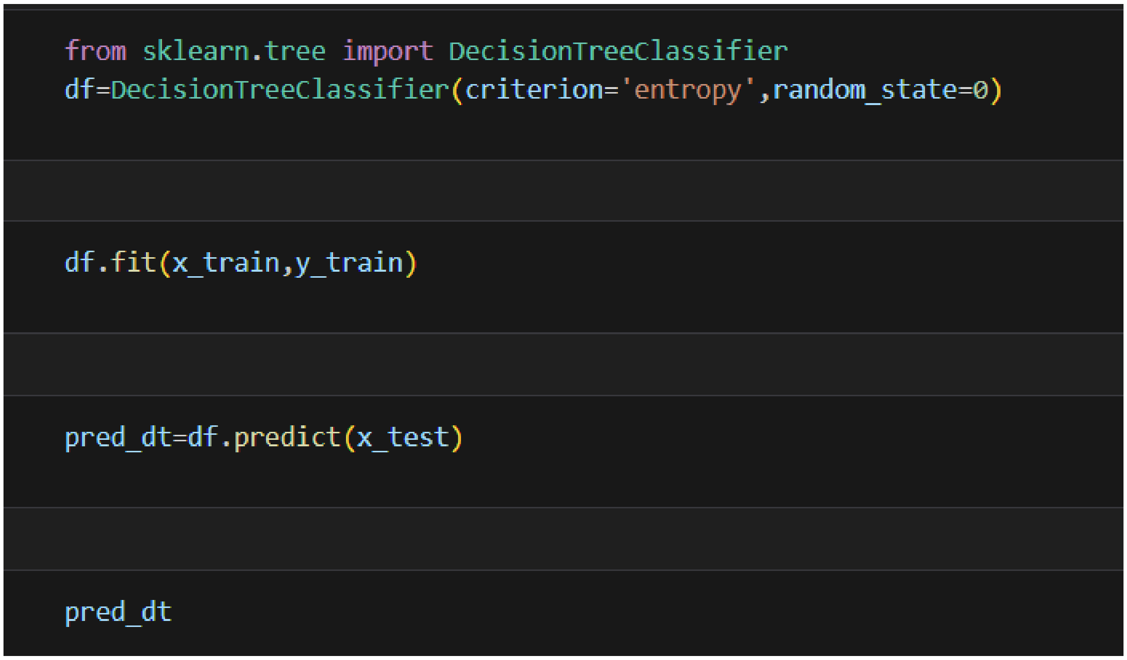
Initial Model Training Code:

1) **Splitting the data into test and train split.**



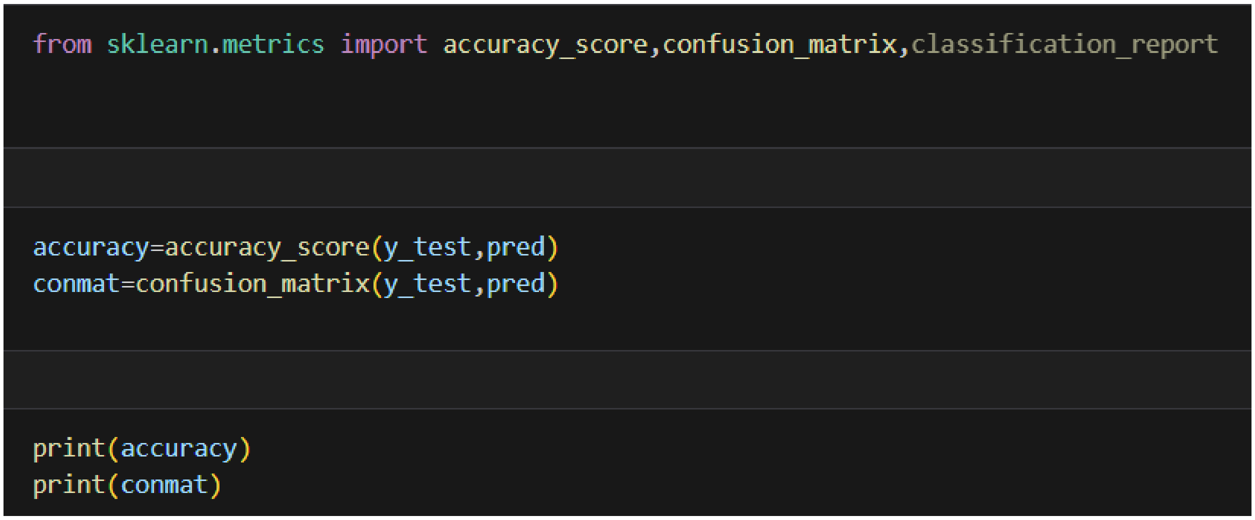
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.3, random_state=0)

x_train.shape

x_test.shape
```

2)Decision tree classifier



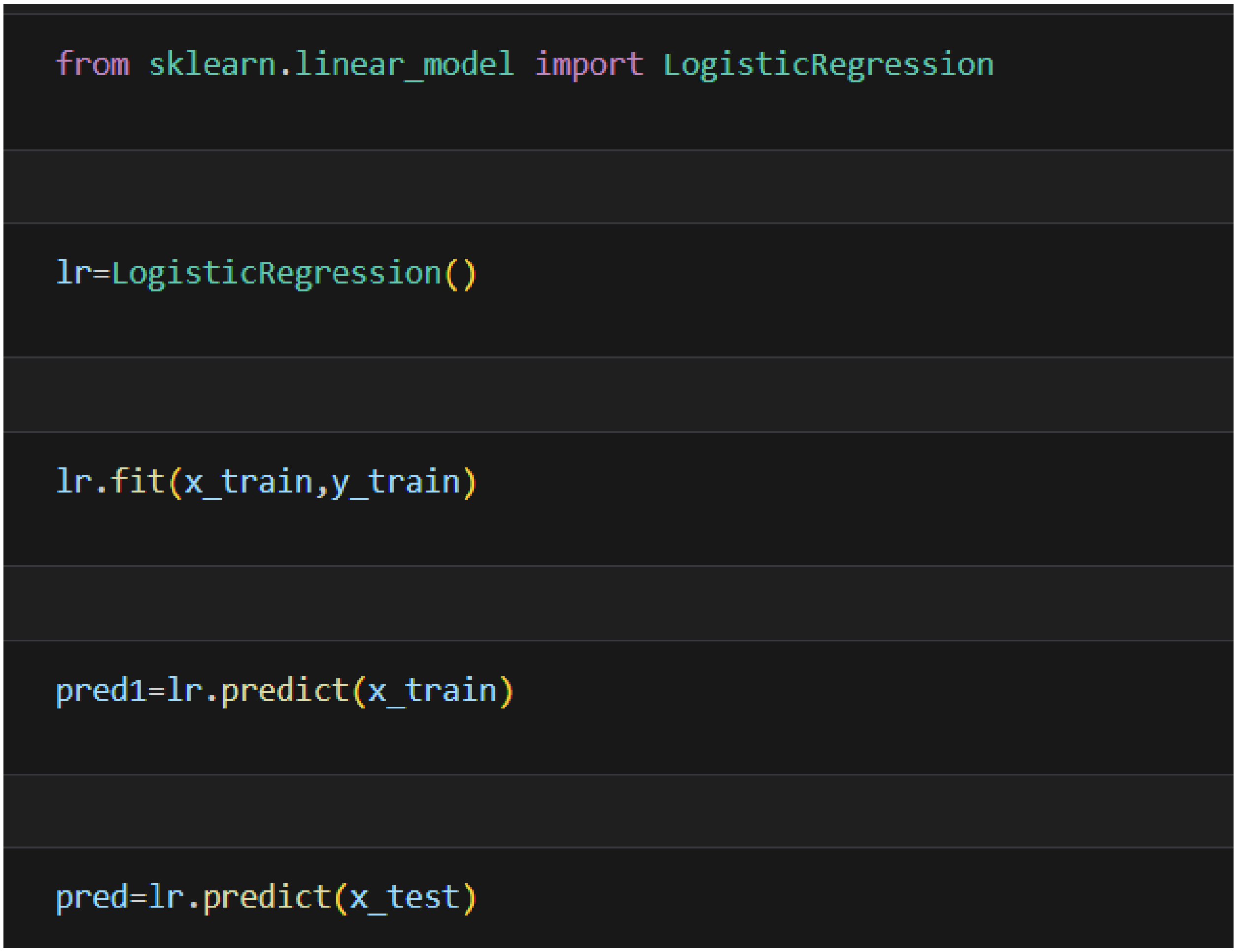
```
from sklearn.tree import DecisionTreeClassifier
df=DecisionTreeClassifier(criterion='entropy',random_state=0)

df.fit(x_train,y_train)

pred_dt=df.predict(x_test)

pred_dt
```

3)Evaluation metrics



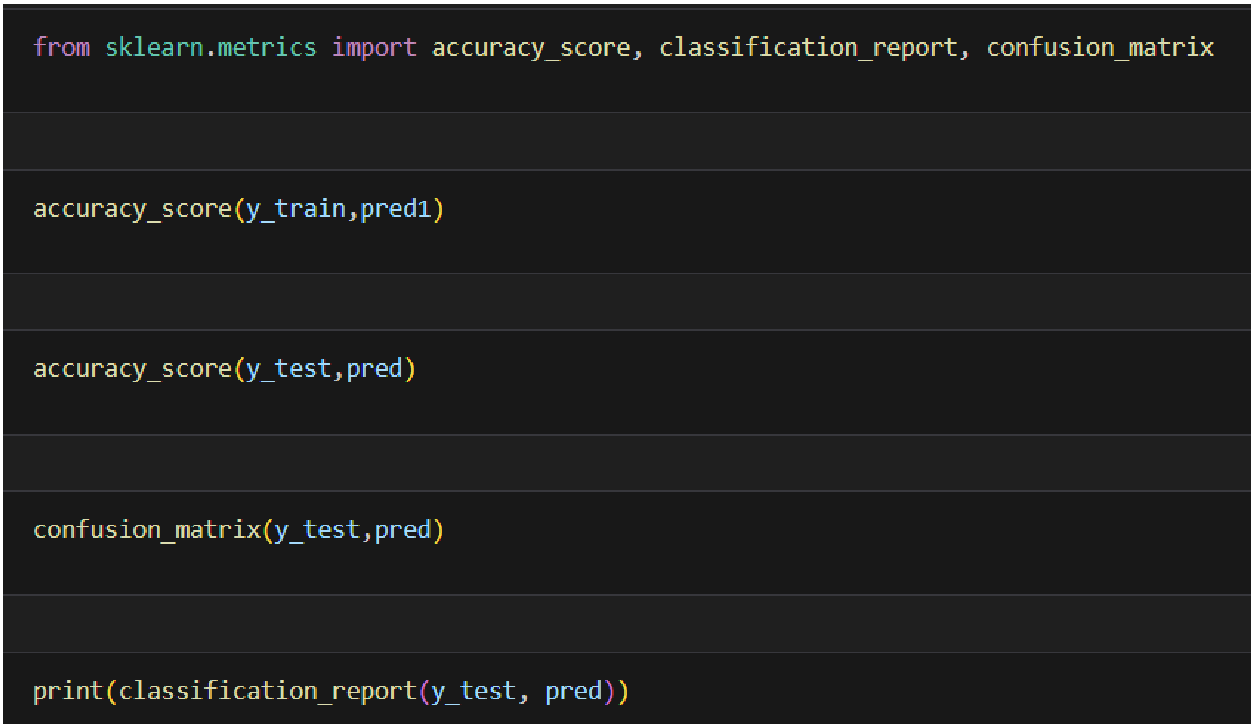
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

accuracy=accuracy_score(y_test,pred)
conmat=confusion_matrix(y_test,pred)

print(accuracy)
print(conmat)
```

4) Logisitic regression



```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()

lr.fit(x_train,y_train)

pred1=lr.predict(x_train)

pred=lr.predict(x_test)
```

5) Evaluation metrics



```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
accuracy_score(y_train,pred1)
```

```
accuracy_score(y_test,pred)
```

```
confusion_matrix(y_test,pred)
```

```
print(classification_report(y_test, pred))
```

6) KNN Classifier

```
x_train1,x_test1,y_train1,y_test1=train_test_split(x,y,test_size=0.2, random_state=0)
```

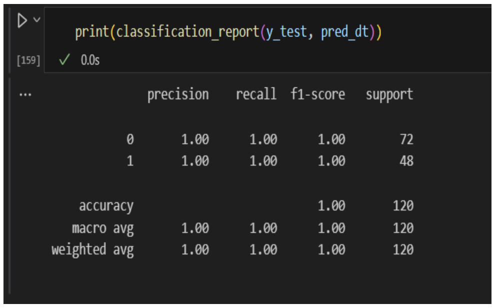
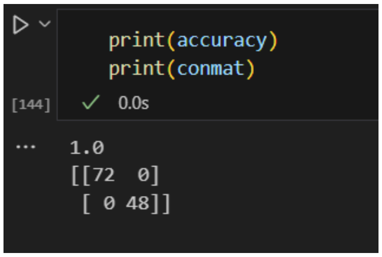
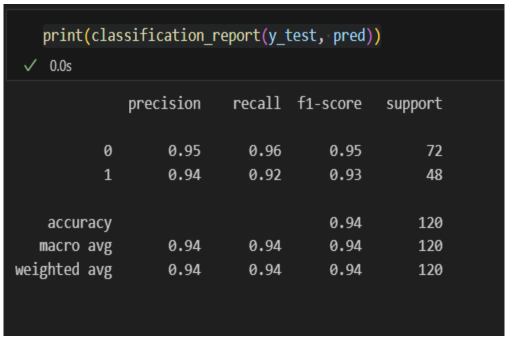
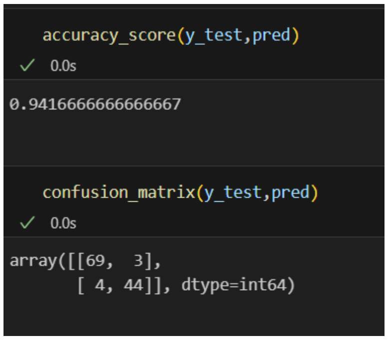
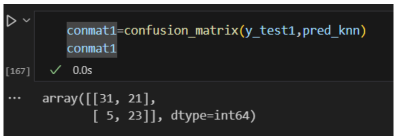
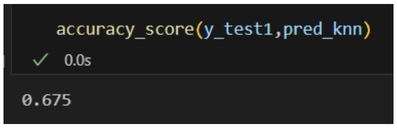
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

```
knn.fit(x_train1,y_train1)
```

```
pred_knn=knn.predict(x_test1)
```

```
pred_knn
```

Model Validation and Evaluation Report:

Model	**Classification Report**	**Accuracy**	**Confusion Matrix**
Decision tree classifier		1.00	
Logistic Regression		0.941	
KNN Classification		0.68	 

Decision tree classifier

```
print(classification_report(y_test, pred_dt))
[159]  ✓ 0.0s
...              precision    recall  f1-score   support

           0       1.00      1.00      1.00        72
           1       1.00      1.00      1.00        48

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

```
print(accuracy)
print(conmat)
[144]  ✓ 0.0s
...  1.0
     [[72  0]
      [ 0 48]]
```

Logistic Regression

```
print(classification_report(y_test, pred))
✓ 0.0s
              precision    recall  f1-score   support

           0       0.95      0.96      0.95        72
           1       0.94      0.92      0.93        48

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

```
accuracy_score(y_test,pred)
✓ 0.0s
0.9416666666666667
```

```
confusion_matrix(y_test,pred)
✓ 0.0s
array([[69,  3],
       [ 4, 44]], dtype=int64)
```

KNN Classification

```
print(classification_report(y_test1,pred_knn))
✓ 0.0s
              precision    recall  f1-score   support

           0       0.86      0.60      0.70        52
           1       0.52      0.82      0.64        28

    accuracy                           0.68        80
   macro avg       0.69      0.71      0.67        80
weighted avg       0.74      0.68      0.68        80
```

```
conmat1=confusion_matrix(y_test1,pred_knn)
conmat1
[167]  ✓ 0.0s
...  array([[31, 21],
            [ 5, 23]], dtype=int64)
```

```
accuracy_score(y_test1,pred_knn)
✓ 0.0s
0.675
```

Model Development Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	6 Marks

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Decision tree Classifier	easily interpretable tree-like model for classification, highlights important features and provides clear, visual insights into the decision-making process for diagnosing chronic kidney disease.	--	Accuracy score=100%

Logistic Regression	statistical model used for binary classification, predicts the probability of a binary outcome based on input features. Simple and effective in predicting chronic kidney diseases.	--	Accuracy =94.1%
KNN Classifier	Classifies based on nearest neighbors; adapts well to data patterns, effective for local variations in chronic disease classification criteria.	--	Accuracy=67.5%

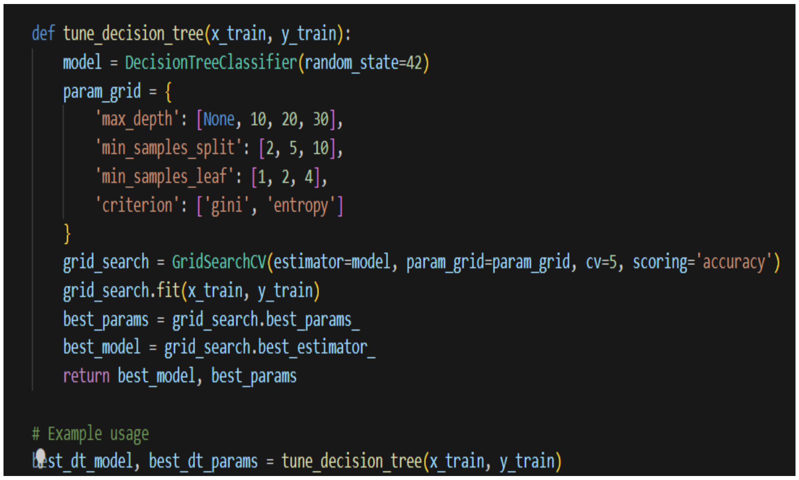
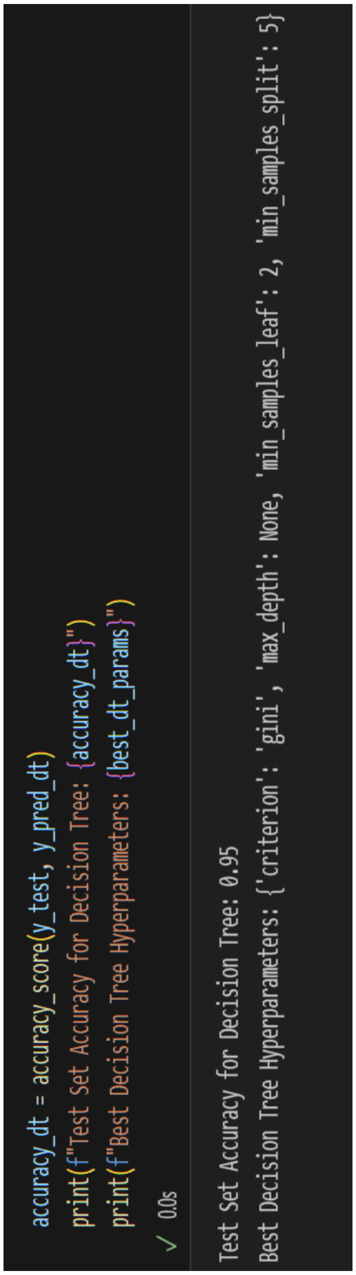
Model Optimization and Tuning Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

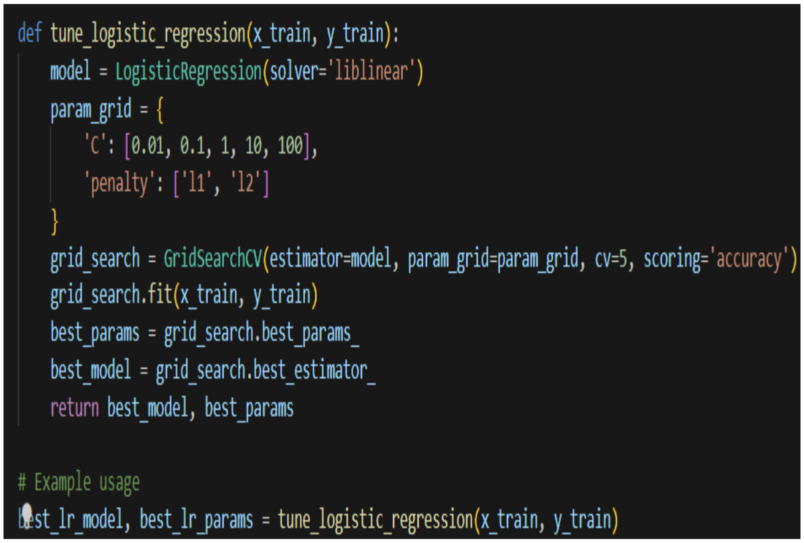
Model	Tuned Hyperparameters	Optimal Values
Decision Trees Classifier		

```python
def tune_decision_tree(x_train, y_train):
    model = DecisionTreeClassifier(random_state=42)
    param_grid = {
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'criterion': ['gini', 'entropy']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train, y_train)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params

# Example usage
best_dt_model, best_dt_params = tune_decision_tree(x_train, y_train)
```

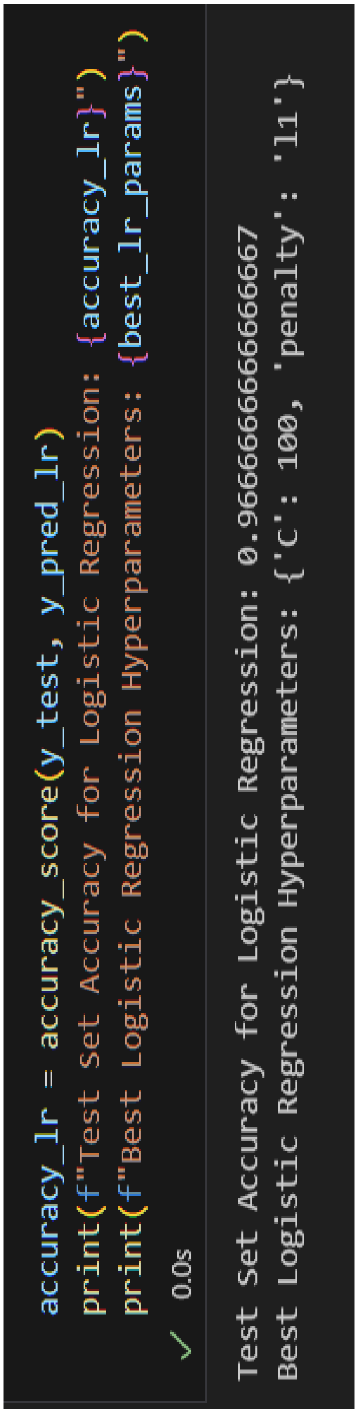
```python
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f"Test Set Accuracy for Decision Tree: {accuracy_dt}")
print(f"Best Decision Tree Hyperparameters: {best_dt_params}")
```

```
✓ 0.0s

Test Set Accuracy for Decision Tree: 0.95
Best Decision Tree Hyperparameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5}
```

Logistic regression



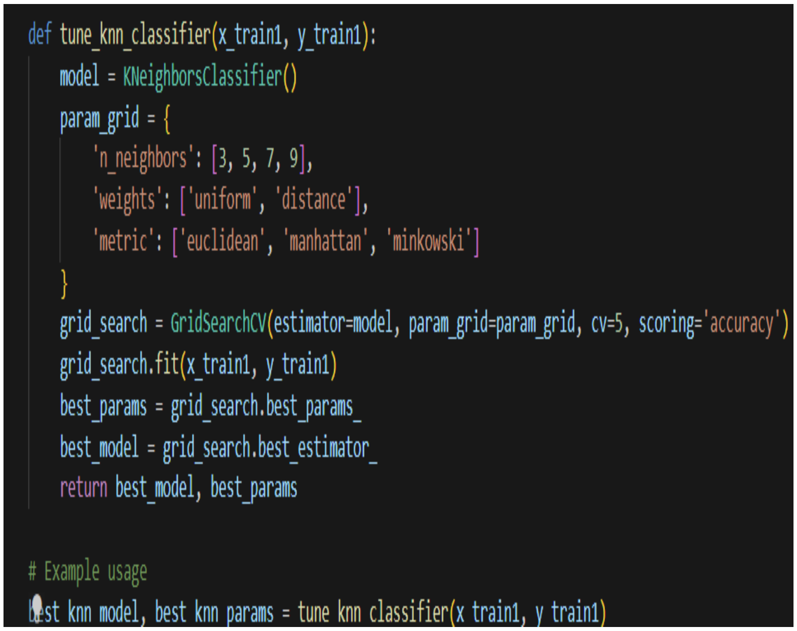
```
def tune_logistic_regression(x_train, y_train):
    model = LogisticRegression(solver='liblinear')
    param_grid = {
        'C': [0.01, 0.1, 1, 10, 100],
        'penalty': ['l1', 'l2']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train, y_train)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params


# Example usage
best_lr_model, best_lr_params = tune_logistic_regression(x_train, y_train)
```



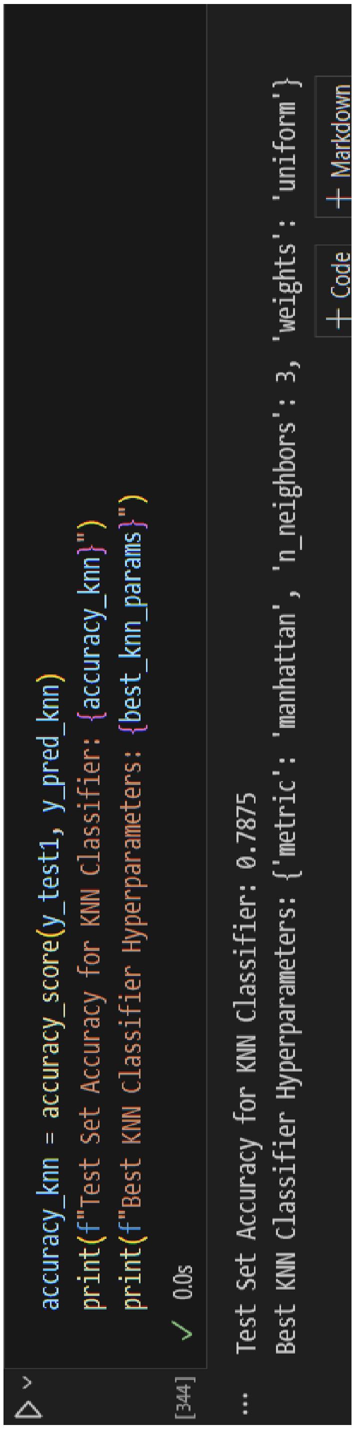
```
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print(f"Test Set Accuracy for Logistic Regression: {accuracy_lr}")
print(f"Best Logistic Regression Hyperparameters: {best_lr_params}")
✓ 0.0s

Test Set Accuracy for Logistic Regression: 0.9666666666666667
Best Logistic Regression Hyperparameters: {'C': 100, 'penalty': 'l1'}
```

KNN

Classifier



```
def tune_knn_classifier(x_train1, y_train1):
    model = KNeighborsClassifier()
    param_grid = {
        'n_neighbors': [3, 5, 7, 9],
        'weights': ['uniform', 'distance'],
        'metric': ['euclidean', 'manhattan', 'minkowski']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train1, y_train1)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params

# Example usage
best_knn_model, best_knn_params = tune_knn_classifier(x_train1, y_train1)
```

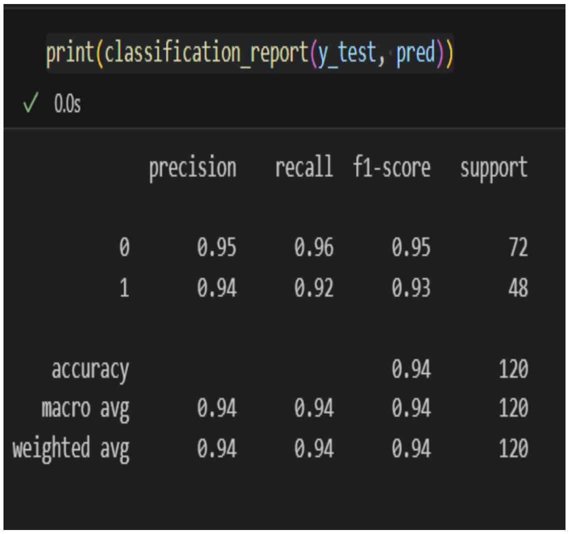
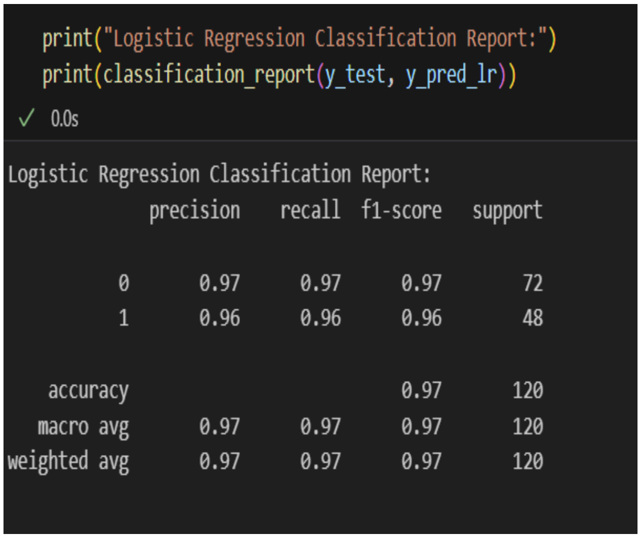
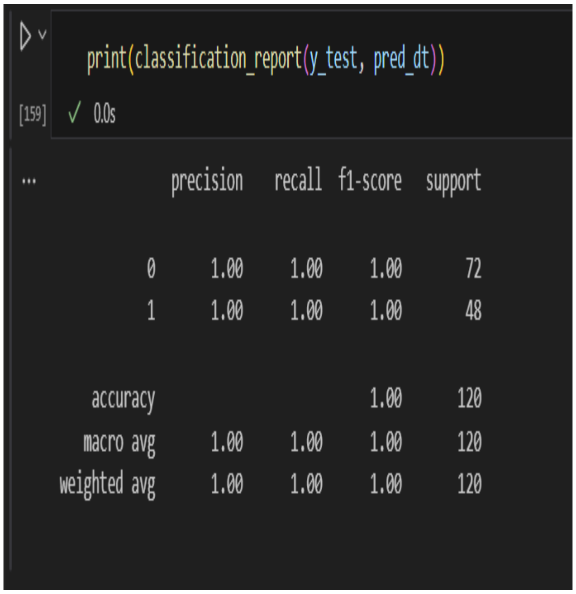
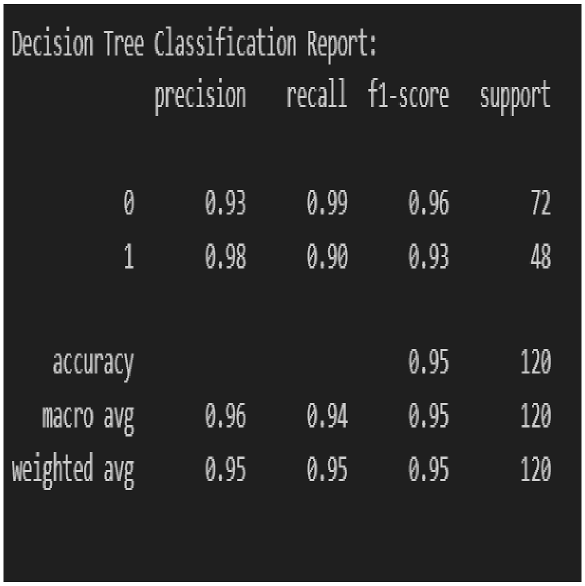


```
accuracy_knn = accuracy_score(y_test1, y_pred_knn)
print(f"Test Set Accuracy for KNN Classifier: {accuracy_knn}")
print(f"Best KNN Classifier Hyperparameters: {best_knn_params}")
```

[344] ✓ 0.0s

```
Test Set Accuracy for KNN Classifier: 0.7875
Best KNN Classifier Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'uniform'}
```

Performance Metrics Comparison Report (2 Marks):

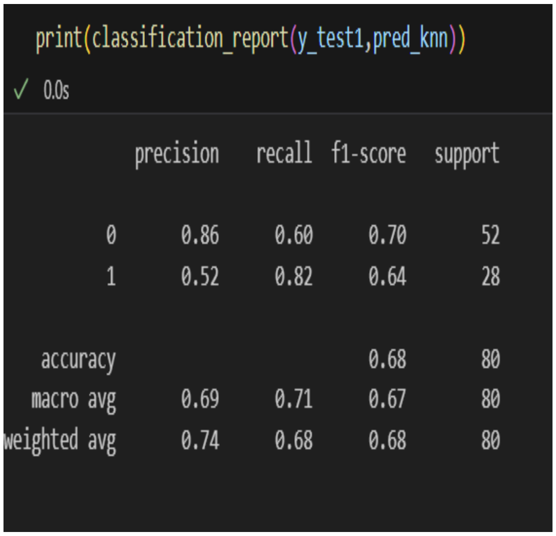
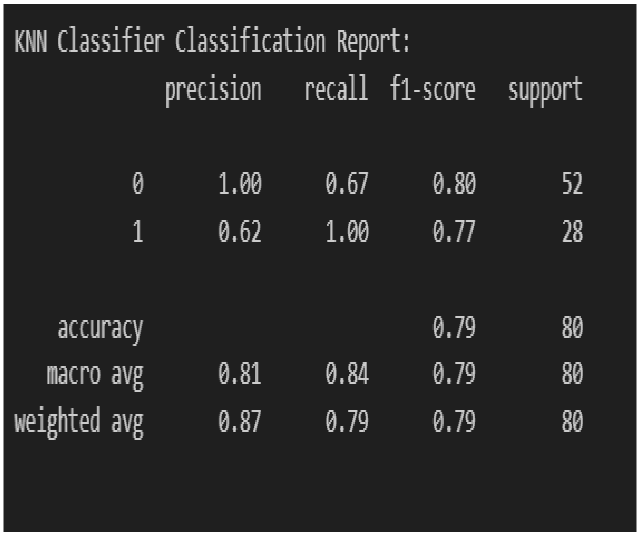
Model	Baseline Metric	Optimized Metric
Logistic regression		
Decision trees classifier		

```
print(classification_report(y_test, pred))
✓ 0.0s

              precision    recall  f1-score   support

           0       0.95      0.96      0.95        72
           1       0.94      0.92      0.93        48

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

```
print("Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred_lr))
✓ 0.0s

Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        72
           1       0.96      0.96      0.96        48

    accuracy                           0.97       120
   macro avg       0.97      0.97      0.97       120
weighted avg       0.97      0.97      0.97       120
```

```
print(classification_report(y_test, pred_dt))
[159] ✓ 0.0s

...           precision    recall  f1-score   support

           0       1.00      1.00      1.00        72
           1       1.00      1.00      1.00        48

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96        72
           1       0.98      0.90      0.93        48

    accuracy                           0.95       120
   macro avg       0.96      0.94      0.95       120
weighted avg       0.95      0.95      0.95       120
```

KNN classifier		

```
print(classification_report(y_test1,pred_knn))
✓ 0.0s

              precision    recall  f1-score   support

           0       0.86      0.60      0.70        52
           1       0.52      0.82      0.64        28

    accuracy                           0.68        80
   macro avg       0.69      0.71      0.67        80
weighted avg       0.74      0.68      0.68        80
```

```
KNN Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        52
           1       0.62      1.00      0.77        28

    accuracy                           0.79        80
   macro avg       0.81      0.84      0.79        80
weighted avg       0.87      0.79      0.79        80
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Logistic Regression	Logistic Regression is the better model for CKD prediction because of its simplicity, interpretability, efficiency, and ability to provide probabilistic outputs, which are crucial for clinical decision-making. Its performance, along with its ability to highlight important features, makes it an excellent choice for medical applications like CKD diagnosis.