

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) Logistic 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	Accur acy	Confusion Matrix																														
Decision tree classifier	<pre>print(classification_report(y_test, pred_dt))</pre> <pre>[159] ✓ 0.0s</pre> <table><thead><tr><th>...</th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>72</td></tr><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td><td>48</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>120</td></tr><tr><td>macro avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>120</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>120</td></tr></tbody></table>	...	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	1.00	<pre>print(accuracy)</pre> <pre>print(conmat)</pre> <pre>[144] ✓ 0.0s</pre> <pre>... 1.0</pre> <pre>[[72 0]</pre> <pre> [0 48]]</pre>
...	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																													
Logistic Regression	<pre>print(classification_report(y_test, pred))</pre> <pre>✓ 0.0s</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.95</td><td>0.96</td><td>0.95</td><td>72</td></tr><tr><td>1</td><td>0.94</td><td>0.92</td><td>0.93</td><td>48</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.94</td><td>120</td></tr><tr><td>macro avg</td><td>0.94</td><td>0.94</td><td>0.94</td><td>120</td></tr><tr><td>weighted avg</td><td>0.94</td><td>0.94</td><td>0.94</td><td>120</td></tr></tbody></table>		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	0.941	<pre>accuracy_score(y_test,pred)</pre> <pre>✓ 0.0s</pre> <pre>0.9416666666666667</pre> <pre>confusion_matrix(y_test,pred)</pre> <pre>✓ 0.0s</pre> <pre>array([[69, 3],</pre> <pre> [4, 44]], dtype=int64)</pre>
	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																													
KNN Classification	<pre>print(classification_report(y_test1,pred_knn))</pre> <pre>✓ 0.0s</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.86</td><td>0.60</td><td>0.70</td><td>52</td></tr><tr><td>1</td><td>0.52</td><td>0.82</td><td>0.64</td><td>28</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.68</td><td>80</td></tr><tr><td>macro avg</td><td>0.69</td><td>0.71</td><td>0.67</td><td>80</td></tr><tr><td>weighted avg</td><td>0.74</td><td>0.68</td><td>0.68</td><td>80</td></tr></tbody></table>		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	0.68	<pre>conmat1=confusion_matrix(y_test1,pred_knn)</pre> <pre>conmat1</pre> <pre>[167] ✓ 0.0s</pre> <pre>... array([[31, 21],</pre> <pre> [5, 23]], dtype=int64)</pre> <pre>accuracy_score(y_test1,pred_knn)</pre> <pre>✓ 0.0s</pre> <pre>0.675</pre>
	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																													