Model Development Phase Template

Date	12th July 2024
Team ID	SWTID1720090815
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 model validation and evaluation report include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Random Forest Classifier

Initial Model

```python
rfc = RandomForestClassifier()
```
✓ 0.0s Python

```python
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,train_size=0.8,random_state=42)
rfc.fit(X_train,y_train)
y_pred= rfc.predict(X_test)
print(f'Accuracy:{accuracy_score(y_test,y_pred)*100:.4f}%')
```
✓ 0.2s Python

Outputs are collapsed ...

```python
y_pred = rfc.predict(Xtest)
print(f'Unseen Data Accuracy:{accuracy_score(ytest,y_pred)*100:.4f}%')
print(f'Unseen Data recall:{recall_score(ytest,y_pred)*100:.4f}%')
```
✓ 0.0s Python

Outputs are collapsed ...

```python
confusion_mat = confusion_matrix(ytest,y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_mat, display_labels = ['False', 'True'])
cm_display.plot()
plt.title("RFC Confusion Matrix")
plt.show()
```
✓ 0.1s Python

Outputs are collapsed ...

```python
print(classification_report(ytest,y_pred))
```
✓ 0.0s Python

Logistic Regression

Initial Model

```python
lr = LogisticRegression()
```
✓ 0.0s Python

```python
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,test_size=0.2,random_state=42)
lr.fit(X_train,y_train)
y_pred= lr.predict(X_test)
print(f'Accuracy:{accuracy_score(y_test,y_pred)*100:.4f}%')
```
✓ 0.0s Python

Outputs are collapsed ...

```python
y_pred = lr.predict(Xtest)
print(f'Unseen Data Accuracy:{accuracy_score(ytest,y_pred)*100:.4f}%')
print(f'Unseen Data Recall:{recall_score(ytest,y_pred)*100:.4f}%')
```
✓ 0.0s Python

Outputs are collapsed ...

```python
confusion_mat = confusion_matrix(ytest,y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_mat, display_labels = ['False', 'True'])
cm_display.plot()
plt.title("LR Confusion Matrix")
plt.show()
```
✓ 0.1s Python

Outputs are collapsed ...

```python
print(classification_report(ytest,y_pred))
```
✓ 0.0s Python

Decision Tree Classifier

Initial Model

```python
dt= DecisionTreeClassifier()
```
✓ 0.0s Python

```python
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,test_size=0.2,random_state=42)
dt.fit(X_train,y_train)
y_pred= dt.predict(X_test)
print(f'Accuracy:{accuracy_score(y_test,y_pred)*100:.4f}%')
```
✓ 0.0s Python

Outputs are collapsed ...

```python
y_pred = dt.predict(Xtest)
print(f'Unseen Data Accuracy:{accuracy_score(ytest,y_pred)*100:.4f}%')
print(f'Unseen Data Recall:{recall_score(ytest,y_pred)*100:.4f}%')
```
✓ 0.0s Python

Outputs are collapsed ...

```python
confusion_mat = confusion_matrix(ytest,y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_mat, display_labels = ['False', 'True'])
cm_display.plot()
plt.title("DT Confusion Matrix")
plt.show()
```
✓ 0.1s Python

Outputs are collapsed ...

```python
print(classification_report(ytest,y_pred))
```
✓ 0.0s Python

XGBoost Classifier

Initial Model

```python
xgb_model = XGBClassifier()
```
✓ 0.0s Python

```python
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,test_size=0.2,random_state=42)
xgb_model.fit(X_train,y_train)
y_pred= xgb_model.predict(X_test)
print(f'Accuracy:{accuracy_score(y_test,y_pred)*100:.4f}%')
```
✓ 1.0s Python

Outputs are collapsed ...

```python
y_pred = xgb_model.predict(Xtest)
print(f'Unseen Data Accuracy:{accuracy_score(ytest,y_pred)*100:.4f}%')
print(f'Unseen Data Recall:{recall_score(ytest,y_pred)*100:.4f}%')
```
✓ 0.1s Python

Outputs are collapsed ...

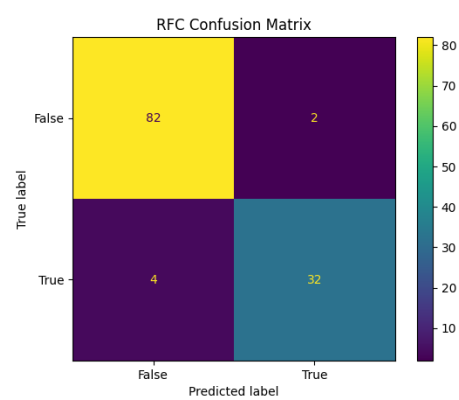
```python
confusion_mat = confusion_matrix(ytest,y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_mat, display_labels = ['False', 'True'])
cm_display.plot()
plt.title("XGBoost Confusion Matrix")
plt.show()
```
✓ 0.2s Python

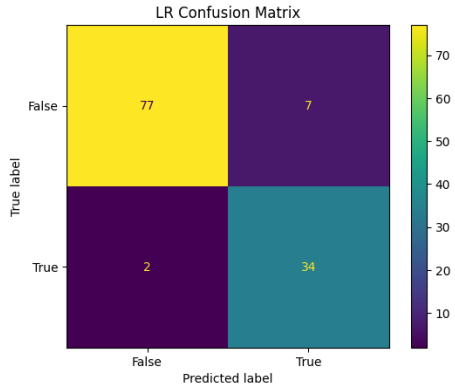
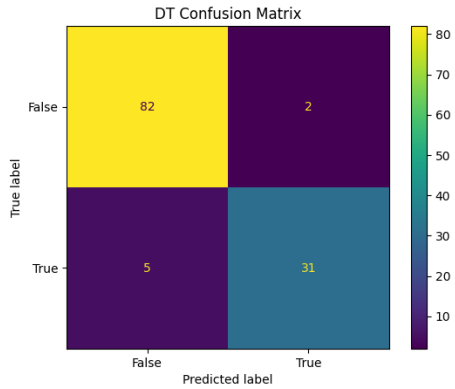
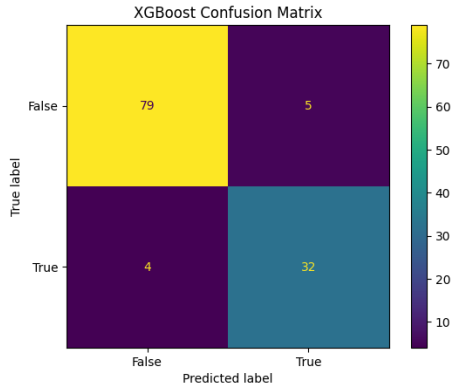
Outputs are collapsed ...

```python
print(classification_report(ytest,y_pred))
```
✓ 0.0s Python

Model Validation and Evaluation Report:

Note: In classification report, 0: CKD, 1: No CKD

Model	Classification Report	Training Accuracy	Unseen data Accuracy	Confusion Matrix
Random Forest Classifier	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.95</td><td>0.98</td><td>0.96</td><td>84</td></tr><tr><td>1</td><td>0.94</td><td>0.89</td><td>0.91</td><td>36</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.95</td><td>120</td></tr><tr><td>macro avg</td><td>0.95</td><td>0.93</td><td>0.94</td><td>120</td></tr><tr><td>weighted avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>120</td></tr></table>	98.39%	95.00%	

<table>
<tr>
<td>Logistic Regression</td>
<td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.97</td><td>0.92</td><td>0.94</td><td>84</td></tr>
<tr><td>1</td><td>0.83</td><td>0.94</td><td>0.88</td><td>36</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.93</td><td>120</td></tr>
<tr><td>macro avg</td><td>0.90</td><td>0.93</td><td>0.91</td><td>120</td></tr>
<tr><td>weighted avg</td><td>0.93</td><td>0.93</td><td>0.93</td><td>120</td></tr>
</table>
</td>
<td>92.50%</td>
<td>94.44%</td>
<td> 
</td>
</tr>
<tr>
<td>Decision Tree Classifier</td>
<td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.94</td><td>0.98</td><td>0.96</td><td>84</td></tr>
<tr><td>1</td><td>0.94</td><td>0.86</td><td>0.90</td><td>36</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>0.92</td><td>0.93</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>
</table>
</td>
<td>94.16%</td>
<td>86.11%</td>
<td> 
</td>
</tr>
<tr>
<td>XGBoost Classifier</td>
<td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.95</td><td>0.94</td><td>0.95</td><td>84</td></tr>
<tr><td>1</td><td>0.86</td><td>0.89</td><td>0.88</td><td>36</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.93</td><td>120</td></tr>
<tr><td>macro avg</td><td>0.91</td><td>0.91</td><td>0.91</td><td>120</td></tr>
<tr><td>weighted avg</td><td>0.93</td><td>0.93</td><td>0.93</td><td>120</td></tr>
</table>
</td>
<td>92.50%</td>
<td>88.88%</td>
<td> 
</td>
</tr>
</table>