



Model Development Phase Template

Date	15 MARCH 2024
Team ID	LTVIP2024TMID25011
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

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
# Split the data
X train, X test, y train, y test = train_test_split(X, y, test_size=0.2, random state=42)
# Initialize and train the model
model = RandomForestClassifier(n estimators=20)
model.fit(X_train, y_train)
# Predict and calculate metrics
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
# Print the confusion matrix and accuracy
print("Confusion Matrix:")
print(cm)
print(f"Accuracy is {round(accuracy_score(y_test, y_pred) * 100, 2)}%")
# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```





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

# Print the accuracy score
print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test)) * 100, 2)}%")

# Print the confusion matrix
print("Confusion Matrix")
print(confusion_matrix(y_test, model.predict(X_test)))

# Print the classification report
print("Classification Report")
print(classification_report(y_test, model.predict(X_test)))
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy score, confusion matrix, classification report
from sklearn.model_selection import train_test_split
# Assuming X_train, X_test, y_train, y_test are already defined
model = LogisticRegression()
model.fit(X train, y train)
pred = model.predict(X_test)
# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))
# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, pred))
```





```
from sklearn.naive bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Assuming X train, X test, y train, y test are already defined
model = GaussianNB()
model.fit(X train, y train)
pred = model.predict(X test)
# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))
# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, pred))
 from sklearn.svm import SVC
 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
 # Assuming X train, X test, y train, y test are already defined
 model = svc()
 model.fit(X train, y train)
 pred = model.predict(X_test)
 # Calculate the accuracy
 accuracy = accuracy_score(y test, pred)
 print(f"Accuracy: {accuracy * 100:.2f}%")
 # Print the confusion matrix
 print("Confusion Matrix:")
 print(confusion_matrix(y test, pred))
 # Print the classification report
 print("\nClassification Report:")
 print(classification_report(y_test, pred))
import pickle
pickle.dump(model, open('kidney.pkl', 'wb'))
```

Model Validation and Evaluation Report:

Model	Classification Report	Accurac	Confusion Matrix
		y	





Random	from altern model, schetting invert train, test_uplit from siltern momelle inper't minderdressicianity from siltern ministe inper't medium getter, scowing, score, classification_report	100.0%	Confusion M	latri)	(:	
Forest	from siterm.metrics import confusion_metrix, accuracy_score, classification_report # spoit thin show X_train_X_lest_y_train_y_test = train_test_uplit(x, y, test_ulze=0.2, random_state=02)		[[23 0]			
Classifier	# Initiative and train the model model = Mandomoreseticastirer_estimators=280 model_first(rain_y_train) model_first(rain_y_train) model_first(rain_y_train) model_first(rain_y_train) model_first(rain_y_train) model_first(rain_y_train) model_first(rain_y_train_y_train) model_first(rain_y_train_y					
	# Product and calculate metrics y.good - model.predict(y.text) cm - confinent_matric(y.text), y.pred)		[0 9]]			
	# more the conjuste entries and occursory print(Confusion metric) print(Confusion metric) print("Scarcy's (confusioners, scarc(, test, y gree) * 180, 2))(")		Accuracy is	100.	. 0%	
	at the last of a contribution and a contribution an					
IZNINI	<pre>print("\u00fclassification Superit") print(classification_report()_test, y_red))</pre>	100.00/	Accuracy is 100.0%			
KNN	from sklearn.metrics import accuracy_score, confusion_matrix, classification_report	100.0%	Confusion Matrix [[23 0]			
	<pre># Print the accuracy score print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test)) * 100, 2);</pre>		[0 9]]			
	# Print the confusion matrix		Classification Report precision	recall f1	-score s	upport
	<pre>print("Confusion Matrix") print(confusion_matrix(y_test, model.predict(X_test)))</pre>		0 1.00	1.00	1.00	23
	<pre># Print the classification report print("Classification Report")</pre>		1 1.00	1.00	1.00	9
	<pre>print(classification_report(y_test, model.predict(X_test)))</pre>		accuracy macro avg 1.00	1.00	1.00	32 32
			weighted avg 1.00	1.00	1.00	32
Logistic	from sklearn.linear_model import LogisticRegression	96.88%	Accuracy: 96.88% Confusion Matrix:			
Regressio	<pre>from sklearn.metrics import accuracy_score, confusion_matrix, classification_report from sklearn.model_selection import train_test_split</pre>		[[23 0] [1 8]]			
n	<pre># Assuming X_train, X_test, y_train, y_test are already defined model = LogisticRegression()</pre>		Classification Report:			
	<pre>model.fit(X_train, y_train) pred = model.predict(X_test)</pre>			recall f1-s	core sup	pport
	<pre># Calculate the accuracy accuracy = accuracy_score(y test, pred)</pre>		0 0.96 1 1.00		0.98 0.94	23 9
	print(f"Accuracy: {accuracy * 100:.2f}%")					
	<pre># Print the confusion matrix print("Confusion Matrix:")</pre>		accuracy macro avg 0.98	0.94	0.97 0.96	32 32
	<pre>print(confusion_matrix(y_test, pred))</pre>		weighted avg 0.97	0.97	0.97	32
	<pre># Print the classification report print("Nclassification Report(") print(classification_report()_test, pred))</pre>					
Nawa	_	100.00/	Accuracy: 100.00%			
Naïve	<pre>from sklearn.naive_bayes import GaussianNB from sklearn.metrics import accuracy_score, confusion_matrix, classification_report</pre>	100.0%	Confusion Matrix: [[23 0]			
Baye's	<pre># Assuming X_train, X_test, y_train, y_test are already defined model = GaussianNB()</pre>		[0 9]]			
	<pre>model.fit(X_train, y_train) pred = model.predict(X_test)</pre>		Classification Report:			
	<pre># Calculate the accuracy accuracy = accuracy_score(y_test, pred)</pre>		precision	recall f	1-score	support
	<pre>print(f"Accuracy: (accuracy * 100:.2f)%") # Print the confusion matrix</pre>		0 1.00 1 1.00	1.00 1.00	1.00 1.00	23 9
	<pre>print("Confusion Matrix:") print(confusion_matrix(y_test, pred))</pre>					22
	<pre># Print the classification report print("\nclassification Report:")</pre>		accuracy macro avg 1.00	1.00	1.00 1.00	32 32
CVD 4	<pre>print(classification_report(y_test, pred))</pre>	01.050/	weighted avg 1.00 Accuracy: 81.25%	1.00	1.00	32
SVM	from sklearn.swm import SVC from sklearn.metrics import accuracy_score, confusion_matrix, classification_report	81.25%	Confusion Matrix:			
	# Assuming X_train, X_test, y_train, y_test are already defined model = SVC()		[[23 0] [6 3]]			
	<pre>model.fit(X_train, y_train) pred = model.predict(X_test)</pre>		Classification Report:			
	<pre># Calculate the accuracy accuracy = accuracy_score(y_test, pred) print(f"Accuracy: (accuracy * 100: .2F)%")</pre>		precision	recall	f1-score	supp
	# Print the confusion matrix		0 0.79	1.00	0.88	
	<pre>print("Confusion Matrix:") print(confusion_matrix(y_test, pred))</pre>		1 1.00	0.33	0.50	
	# Print the classification report print("Nclassification Report:") print("Nclassification Report: ")		accuracy		0.81	
	<pre>print(classification_report(y_test, pred))</pre>		macro avg 0.90 weighted avg 0.85	0.67 0.81	0.69 0.78	
				0.01	0.70	