



Model Development Phase Template

Date	09 JULY 2024
Team ID	SWTID1720190579
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:

```
#KNN
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model selection import train test split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Sample data (replace with your actual data)
# x = ...
# y = ...
# Ensure x and y are numpy arrays or pandas DataFrame/Series
x = np.array(x)
y = np.array(y)
# Training the model
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
knn = KNeighborsClassifier()
knn.fit(x train, y train)
```





```
# Test the model
pred = knn.predict(x_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy}")
```

```
#Initializing the Maive Bayes model
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()

#Train the model
nb.fit(x_train,y_train)

#test the model
pred=nb.predict(x_test)
pred

# Evaluate the Model performance
from sklearn import metrics
metrics.confusion_matrix(y_test,pred)

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





```
#SVM
                                                                   回个个中中
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Create an SVM classifier object
svm = SVC()
# Train the model
svm.fit(X_train, y_train)
# Make predictions on the test set
y_pred = svm.predict(X_test)
# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```





```
# Logistic Regression
                                                                   回个小古早
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Model Validation and Evaluation Report:

Model Classification Report	F1 Score	Confusion Matrix
-----------------------------	----------	------------------





KNN	# Print the classification report and confusion matrix print("Confusion Matrix:") print(confusion_matrix(y_test, pred)) print("Inclassification Report:") print(classification Report:") print(classification report(y_test, pred)) Classification Report:	97%	Accuracy: 0.975 Confusion Matrix: [[51 1] [1 27]]
Naive Bayes	from skleann.metrics import accuracy_score accuracy_score(y_test,pred) print("Confusion Netris:") print(confusion_metris(y_test, pred)) print(confusion_metris(y_test, pred)) print(classification_meport:") print(classification_report(y_test, pred)) precision recall f1-score support 0 0.88 0.92 0.95 52 1 0.87 0.96 0.92 28 accuracy 0.94 0.94 80 macro avg 0.93 0.94 0.93 80 weighted avg 0.94 0.94 80	96%	Confusion Matrix: [[48 4] [1 27]]
SVM	# Generate a classification report report s classification report(y_test, y_pred) print(Classification Report(y') print(Confusion Natrikx') print(Confusion Natrikx') print(Confusion Natrikx') print(Confusion Natriky', test, pred)) print("Nuclassification Report(y_test, pred)) Accuracy: 0.975 Classification Report(y_test, pred)) Accuracy: 0.978 0 0.98 0.98 0.98 52 1 0.96 0.96 0.96 28 accuracy 0 0.98 0.98 0.98 52 accuracy 0 0.97 0.97 0.97 80 narco avg 0.97 0.97 0.97 80 weighted avg 0.97 0.97 0.97 80	97%	Confusion Matrix: [[33 19] [16 12]]
Logistic Regression	print("Accuracy:", accuracy) report = classification_report(y_text, y_pred) print("Classification_report(y_text, y_pred) print("Classification_Report:") print("Confusion_Natrix(y_text, pred)) print("Confusion_Natrix(y_text, pred)) print("Classification_Report:") print("Classification_report(y_text, pred)) Accuracy: 0.979 Classification_Report: 0 0.080 0.98 0.98 52 1 0.96 0.96 0.96 28 3 ccuracy 1 0.96 0.96 0.97 28 3 ccuracy 3 ccuracy 3 ccuracy 4 0.97 0.97 0.97 80 3 ccuracy 4 0.97 0.97 0.97 80 3 ccuracy 5 0.97 0.97 80 5 0.97 0.97 80 5 0.97 0.97 80	97%	Confusion Matrix: [[33 19] [16 12]]
Decision tree	print("Accuracy:", accuracy) report a classification_report(y_test, y_pred) print("Classification_Report(") print("Classification_Report:") Accuracy: 0.975 Classification_Report: recall f1-score support 0 0.96 1.00 0.98 52 1 1.00 0.93 0.96 28 1 1.00 0.93 0.96 0.97	97%	Confusion Matrix: [[33 19] [16 12]]