Model Development Phase Template

Date	09 JULY 2024
Team ID	SWTID1720174514
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

```
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)
```

```
# 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}")

# Print the classification report and confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))
print("\nClassification Report:")
print(classification_report(y_test, pred))
```

```
#Naive Bayes

#Initializing the Naive 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))

from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

```
import pickle
ensemble_model = VotingClassifier(estimators=[('svm', svm), ('logreg', logreg), ('decision_tree', decision_tree), ('

ensemble_model.fit(X_train, y_train)
# Save the model as a pickle file
with open('kd.pkl', 'wb') as file:
    pickle.dump(ensemble_model, file)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix

KNN	`# Calculate the accuracy`
	`accuracy = accuracy_score(y_test, pred)`
	`print(f"Accuracy: {accuracy}")`
	
	`# Print the classification report and confusion matrix`
	`print("Confusion Matrix:")`
	`print(confusion_matrix(y_test, pred))`
	`print("\nClassification Report:")`
	`print(classification_report(y_test, pred))`
		97.5%	`Accuracy: 0.975`
			`Confusion Matrix:`
			`[[51  1]`
			`[ 1 27]]`

LOGISTIC REGRESSION	`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)`
		98.75%	`Accuracy: 0.9875`
			`Classification Report:`
			`precision recall f1-score support`
			`0 0.98 1.00 0.99 52`
			`1 1.00 0.96 0.98 28`
			`accuracy 0.99 80`
			`macro avg 0.99 0.98 0.99 80`
			`weighted avg 0.99 0.99 0.99 80`
			`Confusion Matrix:`
			`[[52 0]`
			`[ 1 27]]`

NAIVE BAYES	`# Confusion Matrix`
	`print("Confusion Matrix:")`
	`print(metrics.confusion_matrix(y_test, pred))`
	
	`# Classification Report`
	`print("\nClassification Report:")`
	`print(metrics.classification_report(y_test, pred))`
		97.5%	`Confusion Matrix:`
			`[[44 8]`
			`[26 2]]`
			`Classification Report:`
			`precision recall f1-score support`
			`0 0.63 0.85 0.72 52`
			`1 0.20 0.07 0.11 28`
			`accuracy 0.57 80`
			`macro avg 0.41 0.46 0.41 80`
			`weighted avg 0.48 0.57 0.51 80`
			`Accuracy: 0.575`

SVM	`# Calculate the accuracy score`
	`accuracy = accuracy_score(y_test, y_pred)`
	`print("Accuracy:", accuracy)`
	
	`# Generate a classification report`
	`report = classification_report(y_test, y_pred)`
	`print("Classification Report:")`
	`print(report)`
		97.5%	`Accuracy: 0.975`
			`Classification Report:`
			`precision recall f1-score support`
			`0 0.96 1.00 0.98 52`
			`1 1.00 0.93 0.96 28`
			`accuracy 0.97 80`
			`macro avg 0.98 0.96 0.97 80`
			`weighted avg 0.98 0.97 0.97 80`