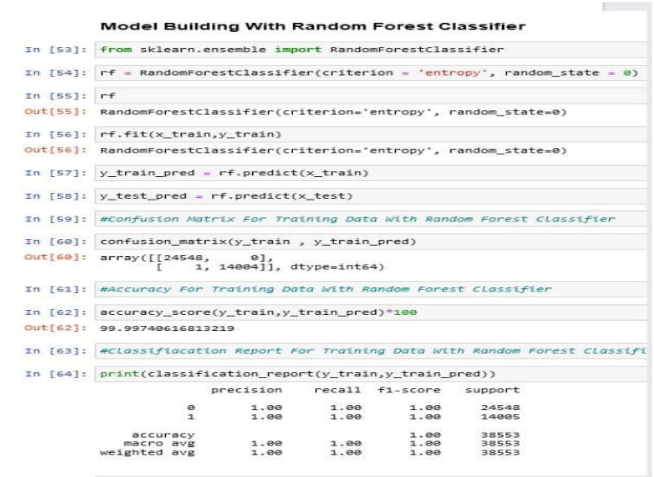
Model Development Phase Template

Date	21 June 2024
Team ID	739680
Project Title	Estimating Presence or Absence of Smoking through bio signals
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:



Model Building With Random Forest Classifier

```
In [53]: from sklearn.ensemble import RandomForestClassifier
In [54]: rf = RandomForestClassifier(criterion = 'entropy', random_state = 0)
In [55]: rf
Out[55]: RandomForestClassifier(criterion='entropy', random_state=0)
In [56]: rf.fit(x_train,y_train)
Out[56]: RandomForestClassifier(criterion='entropy', random_state=0)
In [57]: y_train_pred = rf.predict(x_train)
In [58]: y_test_pred = rf.predict(x_test)
In [59]: #Confusion Matrix For Training Data With Random Forest Classifier
In [60]: confusion_matrix(y_train , y_train_pred)
Out[60]: array([[24548,     0],
                [    1, 14004]], dtype=int64)
In [61]: #Accuracy For Training Data With Random Forest Classifier
In [62]: accuracy_score(y_train,y_train_pred)*100
Out[62]: 99.99740616813219
In [63]: #Classfiacation Report For Training Data With Random Forest Classifi
In [64]: print(classification_report(y_train,y_train_pred))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     24548
           1       1.00      1.00      1.00     14005

    accuracy                           1.00     38553
   macro avg       1.00      1.00      1.00     38553
weighted avg       1.00      1.00      1.00     38553
```

Model Building With Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dec1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
dec1.fit(x_train,y_train)
DecisionTreeClassifier(criterion='entropy', random_state=0)
y_train_pred = dec1.predict(x_train)
y_test_pred = dec1.predict(x_test)
#Confusion Matrix For Training Data With Decision Tree
confusion_matrix(y_train , y_train_pred)
array([[24548,     0],
       [    0, 14005]], dtype=int64)
#Accuracy For Training Data With Decision Tree
accuracy_score(y_train,y_train_pred)*100
100.0
```

Model Building with Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logi = LogisticRegression()
logi
LogisticRegression()
logi.fit(x_train, y_train)
LogisticRegression()
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
y_train_pred = logi.predict(x_train)
y_test_pred = logi.predict(x_test)
# Confusion Matrix For Training Data With Training Data
confusion_matrix(y_train ,y_train_pred)
array([[19115,  5433],
       [ 4376,  9630]], dtype=int64)
# Accuracy For Training Data With Logistic Regression
accuracy_score(y_train,y_train_pred)*100
74.55969704043784
```

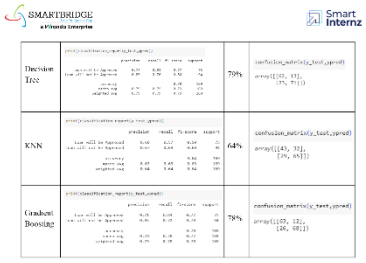
Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix

Random forest classifier	**Model Building With Random Forest Classifier**
	`from sklearn.ensemble import RandomForestClassifier`
	`rf = RandomForestClassifier(criterion = 'entropy', random_state = 0)`
	`rf`
	`RandomForestClassifier(criterion='entropy', random_state=0)`
	`rf.fit(x_train,y_train)`
	`RandomForestClassifier(criterion='entropy', random_state=0)`
	`y_train_pred = rf.predict(x_train)`
	`y_test_pred = rf.predict(x_test)`
	`#Confusion Matrix For Training Data With Random Forest Classifier`
	`confusion_matrix(y_train , y_train_pred)`
	`array([[24548, 0], [ 1, 14004]], dtype=int64)`
	`#Accuracy For Training Data With Random Forest Classifier`
	`accuracy_score(y_train,y_train_pred)*100`
	`99.99740616813219`
	`#Classifiacation Report For Training Data With Random Forest Classifi`
	`print(classification_report(y_train,y_train_pred))`
	precision recall f1-score support
	0 1.00 1.00 1.00 24548
	1 1.00 1.00 1.00 14005
	accuracy 1.00 38553
	macro avg 1.00 1.00 1.00 38553
	weighted avg 1.00 1.00 1.00 38553
		69%	`confusion_matrix(y_test, y_test_pred)`
			`array([[8915, 1476], [1388, 4744]], dtype=int64)`

Decision tree	**Model Building With Decision Tree**
	`from sklearn.tree import DecisionTreeClassifier`
	`dect = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)`
	`dect.fit(x_train,y_train)`
	`DecisionTreeClassifier(criterion='entropy', random_state=0)`
	`y_train_pred = dect.predict(x_train)`
	`y_test_pred = dect.predict(x_test)`
	`#Confusion Matrix For Training Data With Decision Tree`
	`confusion_matrix(y_train , y_train_pred)`
	`array([[24548, 0], [ 0, 14005]], dtype=int64)`
	`#Accuracy For Training Data With Decision Tree`
	`accuracy_score(y_train,y_train_pred)*100`
	`100.0`
		64%	`In [83]: confusion_matrix(y_test, y_test_pred)`
			`Out[83]: array([[8467, 1924], [1942, 4190]], dtype=int64)`

Logistic Regression	**Model Building with Logistic Regression**
	`from sklearn.linear_model import LogisticRegression`
	`logi = LogisticRegression()`
	`logi`
	`LogisticRegression()`
	`logi.fit(x_train, y_train)`
	`LogisticRegression()`
	`from sklearn.metrics import confusion_matrix, accuracy_score, classification_report`
	`y_train_pred = logi.predict(x_train)`
	`y_test_pred = logi.predict(x_test)`
	`# Confusion Matrix For Training Data with Training Data`
	`confusion_matrix(y_train ,y_train_pred)`
	`array([[19115, 5433],`
	`[ 4375, 9630]], dtype=int64)`
	`# Accuracy For Training Data with Logistic Regression`
	`accuracy_score(y_train,y_train_pred)*100`
	`74.55969704043784`
		76.4%	In [83]: `confusion_matrix(y_test, y_test_pred)`
			Out[83]: `array([[5467, 1914],`
			`[1942, 4190]], dtype=int64)`

Gradient Boosting		75%