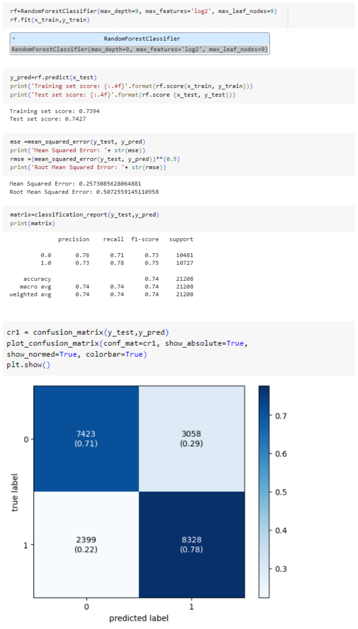
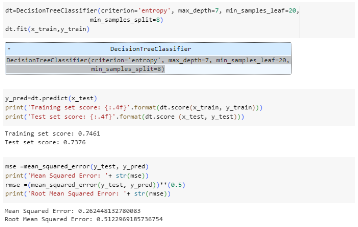
Model Performance Test

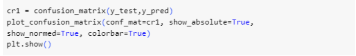
Date	5th November 2023
Team ID	592321
Project Name	Diabetes Prediction using Machine Learning

Model Performance Testing:

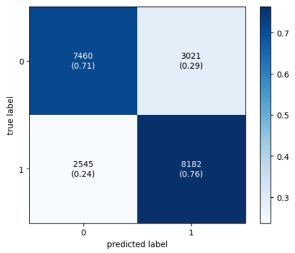
S.No.	Parameter	Values	Screenshot

1.	Metrics	**Regression Model:**
		MAE - , MSE - , RMSE - , R2 score -
		
		Classification Model:
		Confusion Matrix - , Accuray Score- & Classification Report -
			Random Forest:
			
			`rf=RandomForestClassifier(max_depth=9, max_features='log2', max_leaf_nodes=9)`
			`rf.fit(x_train,y_train)`
			`RandomForestClassifier`
			`RandomForestClassifier(max_depth=9, max_features='log2', max_leaf_nodes=9)`
			`y_pred=rf.predict(x_test)`
			`print('Training set score: {:.4f}'.format(rf.score(x_train, y_train)))`
			`print('Test set score: {:.4f}'.format(rf.score (x_test, y_test)))`
			`Training set score: 0.7394`
			`Test set score: 0.7427`
			`mse =mean_squared_error(y_test, y_pred)`
			`print('Mean Squared Error: '+ str(mse))`
			`rmse =(mean_squared_error(y_test, y_pred))**(0.5)`
			`print('Root Mean Squared Error: '+ str(rmse))`
			`Mean Squared Error: 0.2573085628064881`
			`Root Mean Squared Error: 0.5072559145110958`
			`matrix=classification_report(y_test,y_pred)`
			`print(matrix)`
			`              precision    recall  f1-score   support`
			`         0.0       0.76      0.71      0.73     10481`
			`         1.0       0.73      0.78      0.75     10727`
			`    accuracy                           0.74     21208`
			`   macro avg       0.74      0.74      0.74     21208`
			`weighted avg       0.74      0.74      0.74     21208`
			`cr1 = confusion_matrix(y_test,y_pred)`
			`plot_confusion_matrix(conf_mat=cr1, show_absolute=True,`
			`show_normed=True, colorbar=True)`
			`plt.show()`
			 7423
			(0.71)
			3058
			(0.29)
			2399
			(0.22)
			8328
			(0.78)
			0
			1
			true label
			0
			1
			predicted label
			0.7
			0.6
			0.5
			0.4
			0.3
			
			Decision Tree:
			
			`dt=DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=20,`
			`                          min_samples_split=8)`
			`dt.fit(x_train,y_train)`
			`DecisionTreeClassifier`
			`DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=20,`
			`                       min_samples_split=8)`
			`y_pred=dt.predict(x_test)`
			`print('Training set score: {:.4f}'.format(dt.score(x_train, y_train)))`
			`print('Test set score: {:.4f}'.format(dt.score (x_test, y_test)))`
			`Training set score: 0.7461`
			`Test set score: 0.7376`
			`mse =mean_squared_error(y_test, y_pred)`
			`print('Mean Squared Error: '+ str(mse))`
			`rmse =(mean_squared_error(y_test, y_pred))**(0.5)`
			`print('Root Mean Squared Error: '+ str(rmse))`
			`Mean Squared Error: 0.262448132780083`
			`Root Mean Squared Error: 0.5122969185736754`

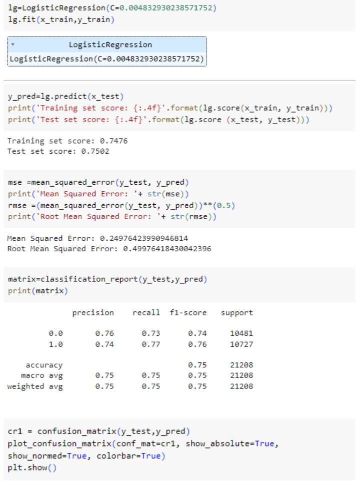
<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>



```
cr1 = confusion_matrix(y_test,y_pred)
plot_confusion_matrix(conf_mat=cr1, show_absolute=True,
show_normed=True, colorbar=True)
plt.show()
```



Logistic Regression:



```
lg=LogisticRegression(C=0.004832930238571752)
lg.fit(x_train,y_train)
```

```
▾                LogisticRegression
LogisticRegression(C=0.004832930238571752)
```

```
y_pred=lg.predict(x_test)
print('Training set score: {:.4f}'.format(lg.score(x_train, y_train)))
print('Test set score: {:.4f}'.format(lg.score (x_test, y_test)))
```

```
Training set score: 0.7476
Test set score: 0.7502
```

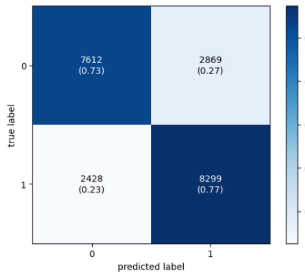
```
mse =mean_squared_error(y_test, y_pred)
print('Mean Squared Error: '+ str(mse))
rmse =(mean_squared_error(y_test, y_pred))**(0.5)
print('Root Mean Squared Error: '+ str(rmse))
```

```
Mean Squared Error: 0.24976423990946814
Root Mean Squared Error: 0.49976418430042396
```

```
matrix=classification_report(y_test,y_pred)
print(matrix)
```

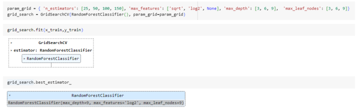
```
              precision    recall  f1-score   support

         0.0       0.76      0.73      0.74     10481
         1.0       0.74      0.77      0.76     10727

    accuracy                           0.75     21208
   macro avg       0.75      0.75      0.75     21208
weighted avg       0.75      0.75      0.75     21208
```

```
cr1 = confusion_matrix(y_test,y_pred)
plot_confusion_matrix(conf_mat=cr1, show_absolute=True,
show_normed=True, colorbar=True)
plt.show()
```



</td>
</tr>
<tr>
<td>2.</td>
<td>Tune the Model</td>
<td>Hyperparameter Tuning -
Validation Method -</td>
<td>

Random Forest:



```
param_grid = { 'n_estimators': [25, 50, 100, 150], 'max_features': ['sqrt', 'log2', None], 'max_depth': [3, 6, 9],  'max_leaf_nodes': [3, 6, 9]}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid=param_grid)
```

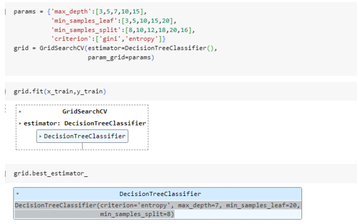
```
grid_search.fit(x_train,y_train)
```

```
▸           GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

```
grid_search.best_estimator_
```

```
▾                         RandomForestClassifier
RandomForestClassifier(max_depth=9, max_features='log2', max_leaf_nodes=9)
```

Decision Tree:



```
params = {'max_depth':[3,5,7,10,15],
          'min_samples_leaf':[3,5,10,15,20],
          'min_samples_split':[8,10,12,18,20,16],
          'criterion':['gini','entropy']}
grid = GridSearchCV(estimator=DecisionTreeClassifier(),
                    param_grid=params)
```

```
grid.fit(x_train,y_train)
```

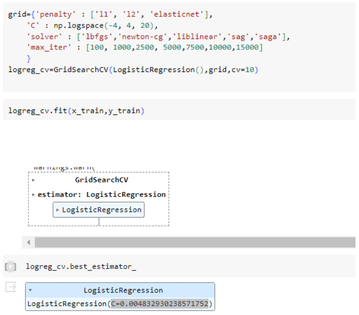
```
▸           GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

```
grid.best_estimator_
```

```
▾                         DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=20,
                       min_samples_split=8)
```

Logistic Regression:

</td>
</tr>
</table>

			
			`grid={'penalty' : ['l1', 'l2', 'elasticnet'],`
			`    'C' : np.logspace(-4, 4, 20),`
			`    'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],`
			`    'max_iter' : [100, 1000,2500, 5000,7500,10000,15000]`
			`    }`
			`logreg_cv=GridSearchCV(LogisticRegression(),grid,cv=10)`
			
			`logreg_cv.fit(x_train,y_train)`
			
			`warnings.warn(`
			`GridSearchCV`
			`estimator: LogisticRegression`
			`LogisticRegression`
			
			`[ ] logreg_cv.best_estimator_`
			
			`LogisticRegression`
			`LogisticRegression(C=0.004832930238571752)`