```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import time
df = pd.read_csv('/content/new_appdata10.csv')
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

### Data PreProcessing

```
y = df['enrolled']
x = df.drop(columns='enrolled')
#splitting the data into training and testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x,y,
test_size=0.2, random_state=0)
# we do not need the user column in the model building but would be
needed in making prediction
# treating the user column
train_identifier = X_train['user']
X_train = X_train.drop(columns='user')

test_identifier = X_test['user']
X_test = X_test.drop(columns='user')

from sklearn.preprocessing import StandardScaler
std_sc = StandardScaler()

# standardizing the data set
X_train2 = pd.DataFrame(std_sc.fit_transform(X_train))
X_test2 = pd.DataFrame(std_sc.transform(X_test))

# setting the column names
X_train2.columns = X_train.columns.values
X_test2.columns = X_test.columns.values

#setting the index numbering
X_train2.index = X_train.index.values
X_test2.index = X_test.index.values

X_train = X_train2
X_test = X_test2
```

### Model Building

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, penalty = 'l1',
solver='liblinear') # we are adding the penlty L1 to change the
```

```
regression model from a regular logistic regression model to a L1
regularization regression model
# we applied this to penalize any variable that might be strongly
correlated with the response variable, similar to what we did in
funneling
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score,
f1_score, precision_score, recall_score
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[3759, 1182],
       [1178, 3881]])
```

```
accuracy_score(y_test, y_pred)
```

```
0.764
```

```
precision_score(y_test,y_pred)
```

```
0.7665415761406281
```

```
recall_score(y_test,y_pred)
```

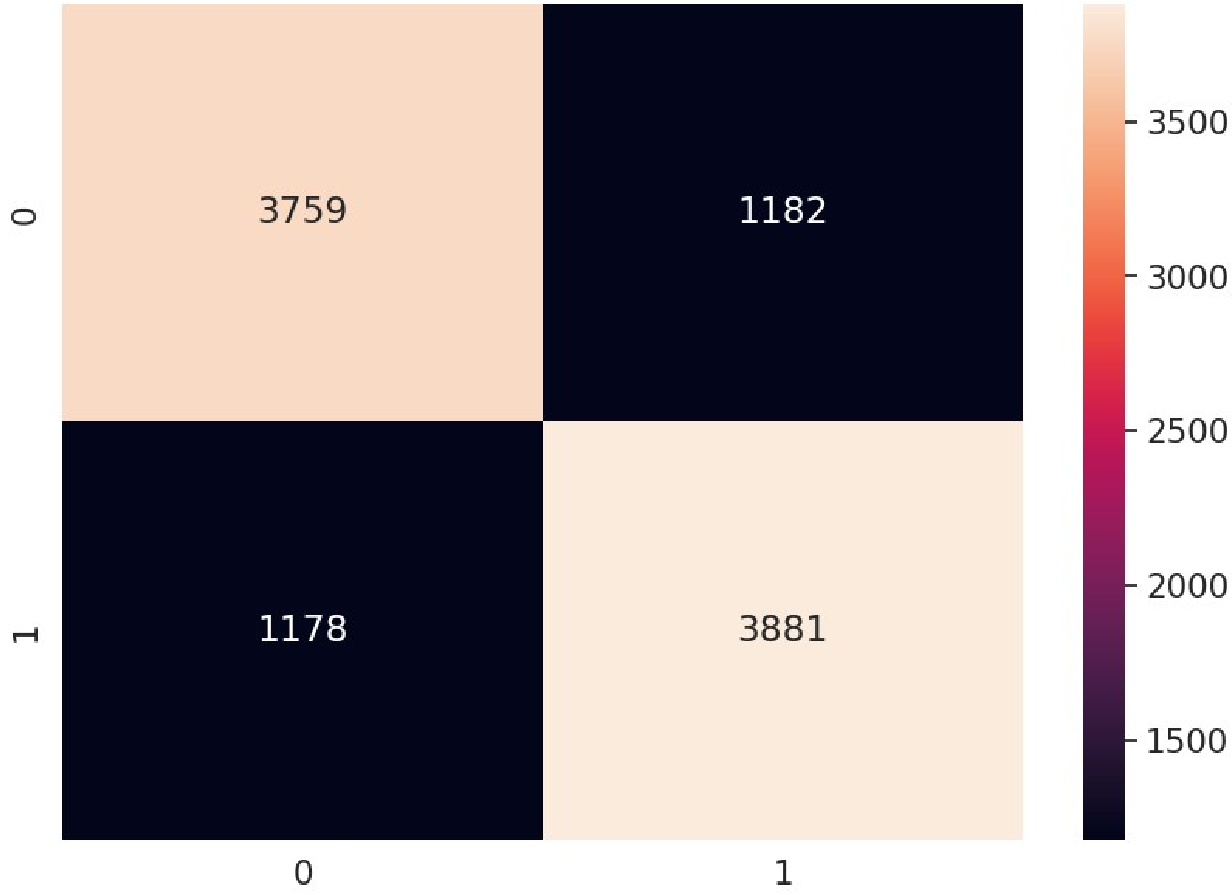
```
0.7671476576398498
```

```
f1_score(y_test,y_pred)
```

```
0.7668444971349536
```

```
df_cm = pd.DataFrame(cm,index=(0,1),columns=(0,1))
plt.figure(figsize=(10,7))
sns.set(font_scale=1.4)
sns.heatmap(cm, annot=True, fmt='g')
print('Test Data Accuracy: %0.4f' % accuracy_score(y_test,y_pred))
```

```
Test Data Accuracy: 0.7640
```



**K fold cross validation**

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator= clf, X= X_train, y= y_train,
cv = 10)
print("logistic Accuracy: %0.3f (+/- %0.3f)"%(accuracies.mean(),
accuracies.std()*2))
```

```
logistic Accuracy: 0.762 (+/- 0.011)
```

**Formatting Final results**

```
final_result = pd.concat([y_test, test_identifier], axis=1).dropna()
final_result['predicted_result']= y_pred
final_result[['user','enrolled','predicted_result']].reset_index(drop=
True)
```

{"summary":"{\n  \"name\":
\"final_result[['user','enrolled','predicted_result']]\",\n  \"rows\":
10000,\n  \"fields\": [\n    {\n      \"column\": \"user\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
107425,\n        \"min\": 23,\n        \"max\": 373639,\n

```
\"num_unique_values\": 9995,\n        \"samples\": [\n
144389,\n          253312,\n          190352\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n       \"column\": \"enrolled\",\n      \"properties\":
{\n         \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n       \"column\": \"predicted_result\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```