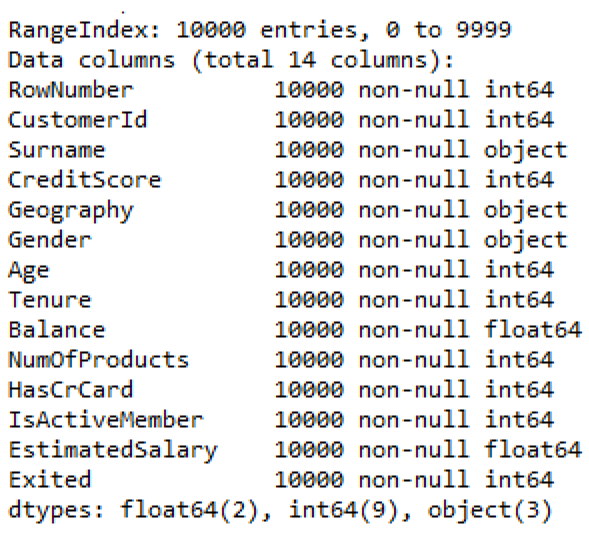
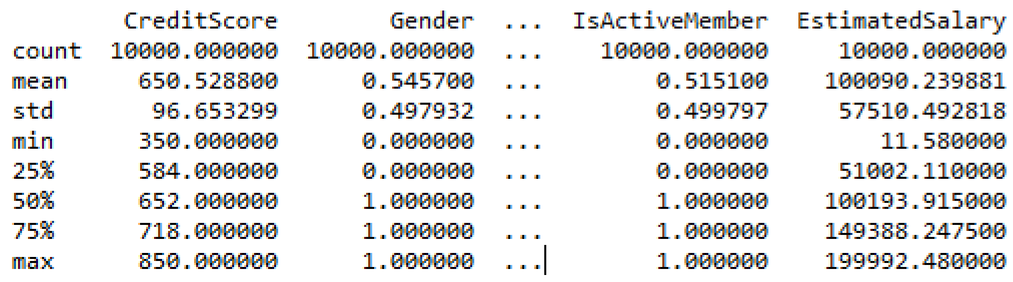


```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
```



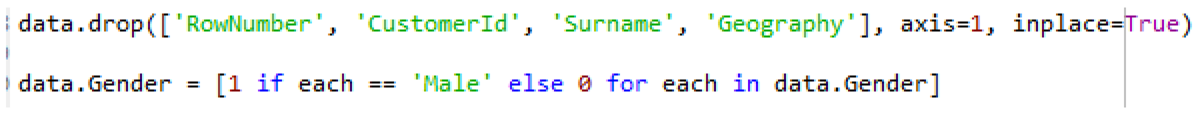
```
        CreditScore        Gender  ...  IsActiveMember  EstimatedSalary
count  10000.000000  10000.000000  ...    10000.000000     10000.000000
mean     650.528800      0.545700  ...        0.515100    100090.239881
std       96.653299      0.497932  ...        0.499797     57510.492818
min      350.000000      0.000000  ...        0.000000        11.580000
25%      584.000000      0.000000  ...        0.000000     51002.110000
50%      652.000000      1.000000  ...        1.000000    100193.915000
75%      718.000000      1.000000  ...        1.000000    149388.247500
max      850.000000      1.000000  ...        1.000000    199992.480000
```

R code:

data.info()

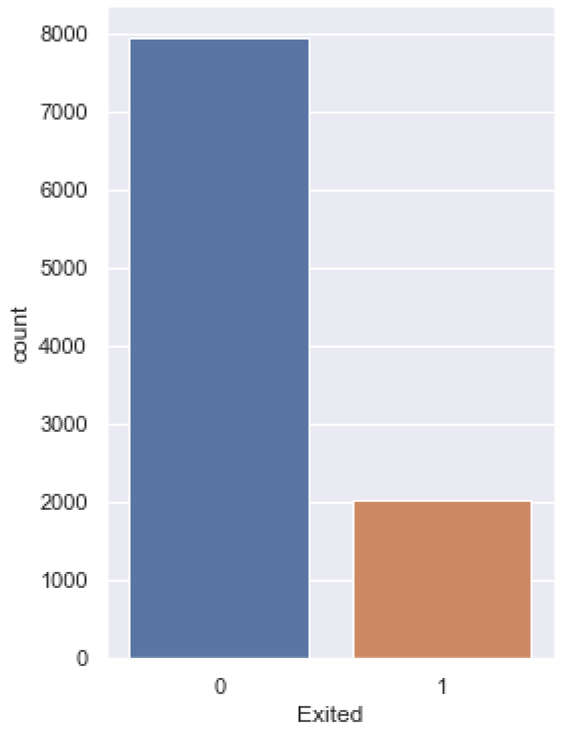
We have 10,000 customer/entries in the dataset with information in above variables.

R code:



```
data.drop(['RowNumber', 'CustomerId', 'Surname', 'Geography'], axis=1, inplace=True)

data.Gender = [1 if each == 'Male' else 0 for each in data.Gender]
```

We do not need RowNumber, CustomerId, Surname, Geography so we drop those variables. We also change Gender data types from object to integer (binary in this case)



```
   CreditScore  Gender       Age  ...  HasCrCard  IsActiveMember  EstimatedSalary
0        0.538     0.0  0.324324  ...        1.0             1.0         0.506735
1        0.516     0.0  0.310811  ...        0.0             1.0         0.562709
2        0.304     0.0  0.324324  ...        1.0             0.0         0.569654
3        0.698     0.0  0.283784  ...        0.0             0.0         0.469120
4        1.000     0.0  0.337838  ...        1.0             1.0         0.395400
```

We use the bar plot to graph the information of exited customer which is about 20% compared to the total customer. We also scale variables by using this function

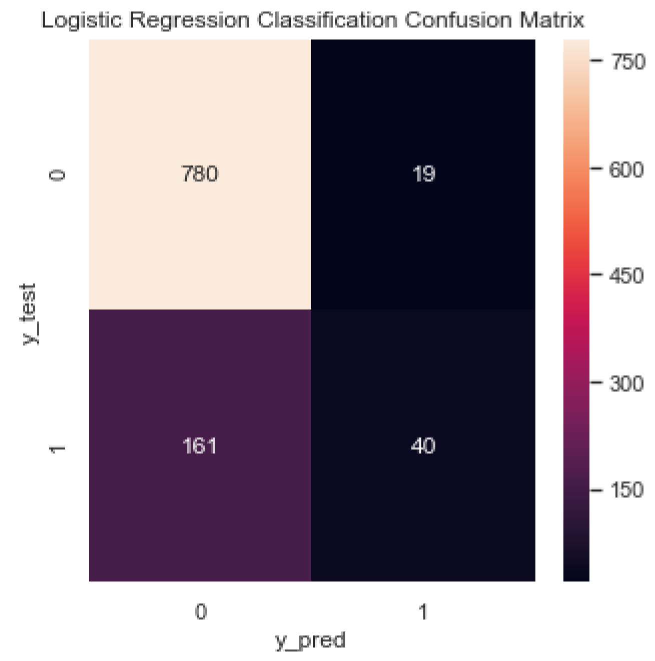
(x_data - np.min(x_data)) / (np.max(x_data)-np.min(x_data))

R code:

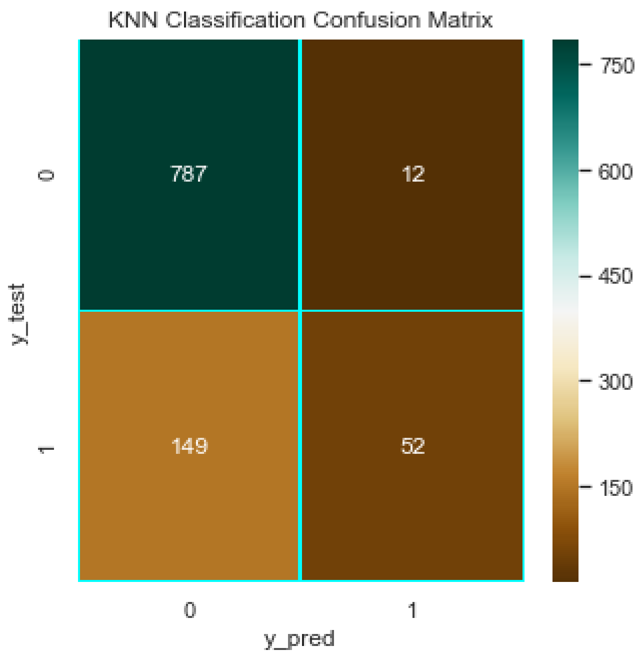
```
plt.figure(figsize=[4,6])
sns.set(style='darkgrid')
ax = sns.countplot(x='Exited', data=data)
print(data.loc[:,'Exited'].value_counts())

y = data.Exited.values
x_data = data.drop(['Exited'], axis=1)
print(x_data.describe())
x = (x_data - np.min(x_data)) / (np.max(x_data)-np.min(x_data))
print(x.head())
```

PREDICTIVE

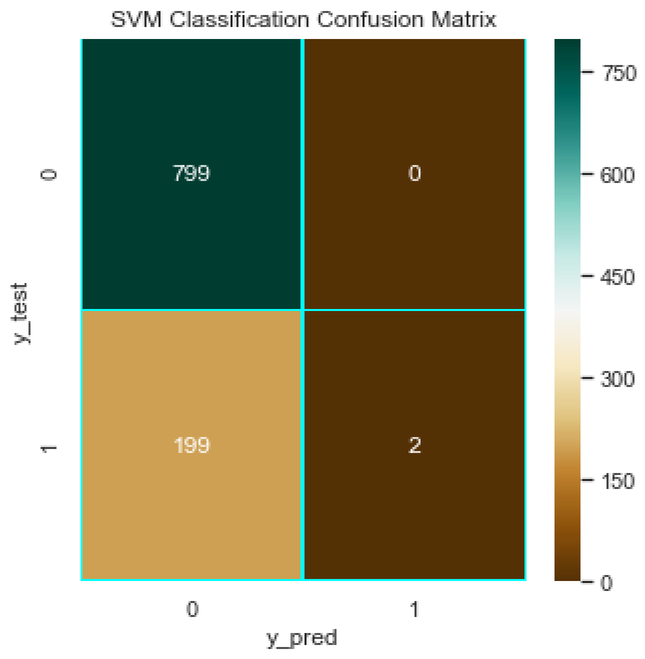
We compare the accuracy of many methods such as: Logistic Regression, KNN, SVM, Decision Tree, Random Forest by using Confusion Matrix.



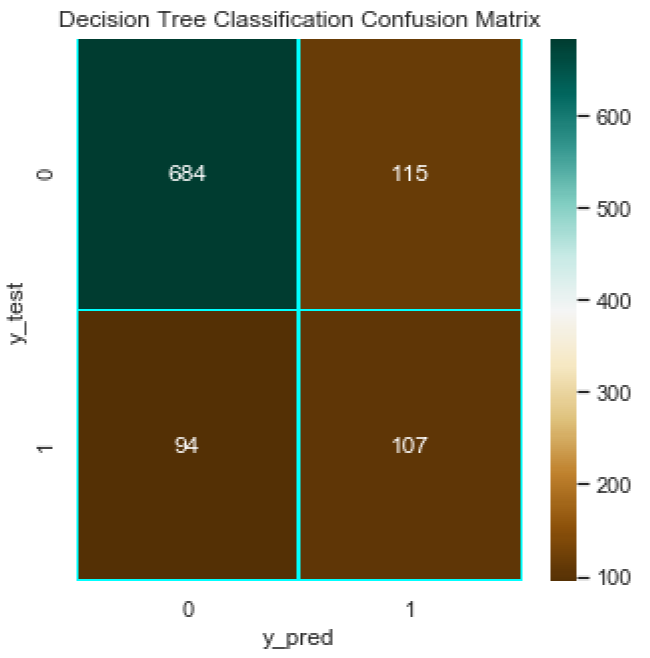
```
0.82
```



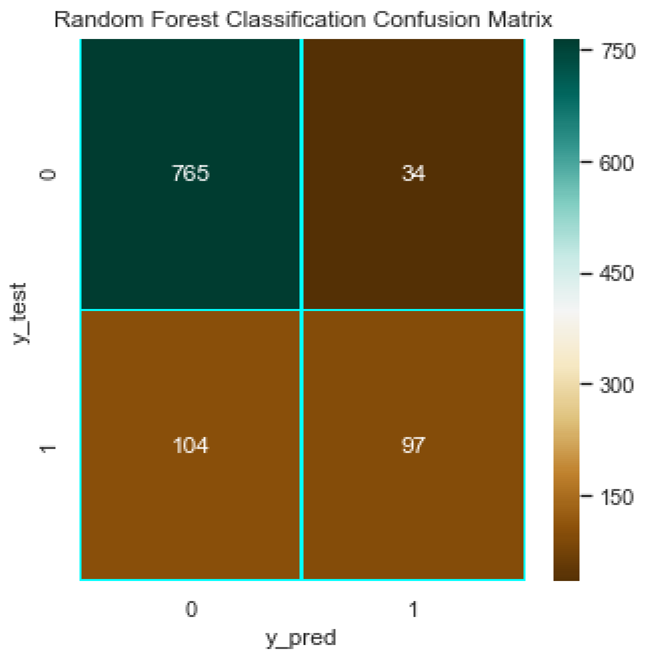
```
0.839
```



```
0.801
```



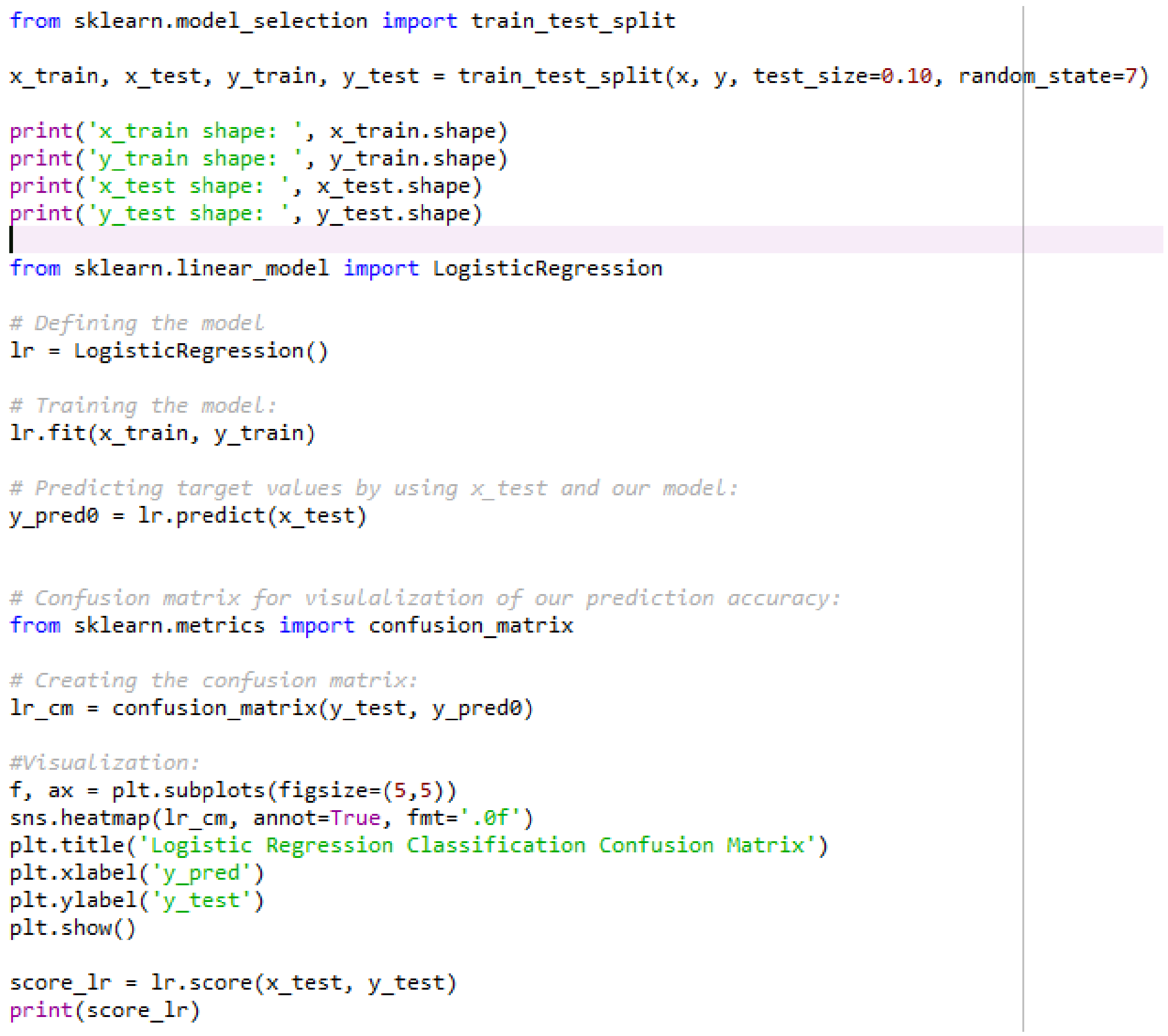
```
0.791
```



```
0.862
```

The Random Forest method has the highest accuracy while the Decision Tree method has the lowest one.

R code:



```python
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.10, random_state=7)

print('x_train shape: ', x_train.shape)
print('y_train shape: ', y_train.shape)
print('x_test shape: ', x_test.shape)
print('y_test shape: ', y_test.shape)

from sklearn.linear_model import LogisticRegression

# Defining the model
lr = LogisticRegression()

# Training the model:
lr.fit(x_train, y_train)

# Predicting target values by using x_test and our model:
y_pred0 = lr.predict(x_test)


# Confusion matrix for visulalization of our prediction accuracy:
from sklearn.metrics import confusion_matrix

# Creating the confusion matrix:
lr_cm = confusion_matrix(y_test, y_pred0)

#Visualization:
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(lr_cm, annot=True, fmt='.0f')
plt.title('Logistic Regression Classification Confusion Matrix')
plt.xlabel('y_pred')
plt.ylabel('y_test')
plt.show()

score_lr = lr.score(x_test, y_test)
print(score_lr)
```