In [50]:
```
# 1.Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy import percentile
from numpy.random import rand
from scipy import stats
from sklearn.metrics import *
import seaborn as sn
```

In [38]:
```
# 2.Read the data as a data frame
dataset = pd.read_csv('E:/IISWBM_Materials/2nd_sem/Advance_analytics/bank-full.csv')

bankdata = pd.DataFrame(dataset)

print(bankdata)
```

```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
...    ...           ...       ...        ...     ...      ...     ...  ...
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

         contact  day month  duration  campaign  pdays  previous poutcome  \
0        unknown    5   may       261         1     -1         0  unknown
1        unknown    5   may       151         1     -1         0  unknown
2        unknown    5   may        76         1     -1         0  unknown
3        unknown    5   may        92         1     -1         0  unknown
4        unknown    5   may       198         1     -1         0  unknown
...          ...  ...   ...       ...       ...    ...       ...      ...
45206   cellular   17   nov       977         3     -1         0  unknown
45207   cellular   17   nov       456         2     -1         0  unknown
45208   cellular   17   nov      1127         5    184         3  success
45209  telephone   17   nov       508         4     -1         0  unknown
45210   cellular   17   nov       361         2    188        11    other

      Target
0         no
1         no
2         no
3         no
4         no
...      ...
45206    yes
45207    yes
45208    yes
45209     no
45210     no

[45211 rows x 17 columns]
```

In [39]:
```
# 3.a.Shape of the data

shape = bankdata.shape
print(shape)
```

```
(45211, 17)
```

In [40]:
```
# 3.b.Data type of each attribute

datatypes = bankdata.dtypes
print(datatypes)
```

```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
Target       object
dtype: object
```

In [41]:
```
# 3.c.Checking the presence of missing values and get rid of those missing values
bankdata.isnull()
bankdata.dropna()
bankdata = bankdata.fillna(100)
int_df = bankdata.select_dtypes(include=['int64']).copy()
float_df=bankdata.select_dtypes(include=['float64']).copy()
bankdata_int_float = pd.concat([float_df,int_df], axis=1, join_axes=[int_df.index])
obj_df = bankdata.select_dtypes(include=['object']).copy()
obj_df.head()
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(obj_df["job"].astype(str))
list(le.classes_)
obj_df_trf=obj_df.astype(str).apply(le.fit_transform)
bankdata_final = pd.concat([bankdata_int_float,obj_df_trf], axis=1, join_axes=[bankdata_int_float.index])
bankdata_final.head()
X = bankdata_final.iloc[:,0:16].values
print(X)
y = bankdata_final.iloc[:, 16].values
print(y)
```

```
D:\Conda\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: FutureWarning: The join_axes-keyword is deprecated. Use
.reindex or .reindex_like on the result to achieve the same functionality.
  import sys

[[  58 2143    5 ...    2    8    3]
 [  44   29    5 ...    2    8    3]
 [  33    2    5 ...    2    8    3]
 ...
 [  72 5715   17 ...    0    9    2]
 [  57  668   17 ...    1    9    3]
 [  37 2971   17 ...    0    9    1]]
[0 0 0 ... 1 0 0]

D:\Conda\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: FutureWarning: The join_axes-keyword is deprecated. Us
e .reindex or .reindex_like on the result to achieve the same functionality.
from ipykernel import kernelapp as app
```

In [42]:
```
# 3.d 5point summary of numerical attributes
age = bankdata.iloc[:,0].values
balance = bankdata.iloc[:,5].values
day = bankdata.iloc[:,9].values
duration = bankdata.iloc[:,11].values
campaign = bankdata.iloc[:,12].values
Pday = bankdata.iloc[:,13].values
age_quartiles = percentile(age, [25, 50, 75])
age_min, age_max = age.min(), age.max()
bal_quartiles = percentile(balance, [25, 50, 75])
bal_min, bal_max = balance.min(), balance.max()
day_quartiles = percentile(day, [25, 50, 75])
day_min, day_max = day.min(), day.max()
duration_quartiles = percentile(duration, [25, 50, 75])
duration_min, duration_max = duration.min(), duration.max()
campaign_quartiles = percentile(campaign, [25, 50, 75])
campaign_min, campaign_max = campaign.min(), campaign.max()
Pday_quartiles = percentile(Pday, [25, 50, 75])
Pday_min, Pday_max = Pday.min(), Pday.max()
print('AGE: ','Min: %.3f',age_min ,'Q1: %.3f',age_quartiles[0],'Median: %.3f',age_quartiles[1] ,
      'Q3: %.3f',age_quartiles[2],'Max: %.3f',age_max)
print('BALANCE: ','Min: %.3f',bal_min,'Q1: %.3f',bal_quartiles[0],'Median: %.3f',bal_quartiles[1],
      'Q3: %.3f',bal_quartiles[2], 'Max: %.3f',bal_max)
print('DAY: ','Min: %.3f',day_min,'Q1: %.3f',day_quartiles[0],'Median: %.3f',day_quartiles[1],
      'Q3: %.3f',day_quartiles[2],'Max: %.3f',day_max)
print('DURATION: ','Min: %.3f' , duration_min , 'Q1: %.3f' , duration_quartiles[0] , 'Median: %.3f' , duration_quartil
es[1] ,
      'Q3: %.3f' , duration_quartiles[2] , 'Max: %.3f' , duration_max)
print('CAMPAIGN: ','Min: %.3f' , campaign_min , 'Q1: %.3f' , campaign_quartiles[0] , 'Median: %.3f' , campaign_quartil
es[1] ,
      'Q3: %.3f' , campaign_quartiles[2] , 'Max: %.3f' , campaign_max)
print('PDAY: ','Min: %.3f' , Pday_min , 'Q1: %.3f' , Pday_quartiles[0] , 'Median: %.3f' , Pday_quartiles[1] ,
      'Q3: %.3f' , Pday_quartiles[2] , 'Max: %.3f' , Pday_max)
```

```
AGE:  Min: %.3f 18 Q1: %.3f 33.0 Median: %.3f 39.0 Q3: %.3f 48.0 Max: %.3f 95
BALANCE:  Min: %.3f -8019 Q1: %.3f 72.0 Median: %.3f 448.0 Q3: %.3f 1428.0 Max: %.3f 102127
DAY:  Min: %.3f 1 Q1: %.3f 8.0 Median: %.3f 16.0 Q3: %.3f 21.0 Max: %.3f 31
DURATION:  Min: %.3f 0 Q1: %.3f 103.0 Median: %.3f 180.0 Q3: %.3f 319.0 Max: %.3f 4918
CAMPAIGN:  Min: %.3f 1 Q1: %.3f 1.0 Median: %.3f 2.0 Q3: %.3f 3.0 Max: %.3f 63
PDAY:  Min: %.3f -1 Q1: %.3f -1.0 Median: %.3f -1.0 Q3: %.3f -1.0 Max: %.3f 871
```

In [43]:
```
# 3.e.Checking the presence of outliers
def drop_numerical_outliers(bankdata_final, z_thresh=3):
    # Constrains will contain `True` or `False` depending on if it is a value below the threshold.
    constrains = bankdata_final.select_dtypes(include=[np.number]).apply(lambda x: np.abs(stats.zscore(x)) < z_thresh,
reduce=False).all(axis=1)
    # Drop (inplace) values set to be rejected
    bankdata_final.drop(bankdata_final.index[~constrains], inplace=True)
```

In [44]:
```
dropdata = drop_numerical_outliers(bankdata_final)
print(dropdata)
```

```
None

D:\Conda\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: The reduce argument is deprecated and wi
ll be removed in a future version. You can specify result_type='reduce' to try to reduce the result to the original d
imensions
  after removing the cwd from sys.path.
```

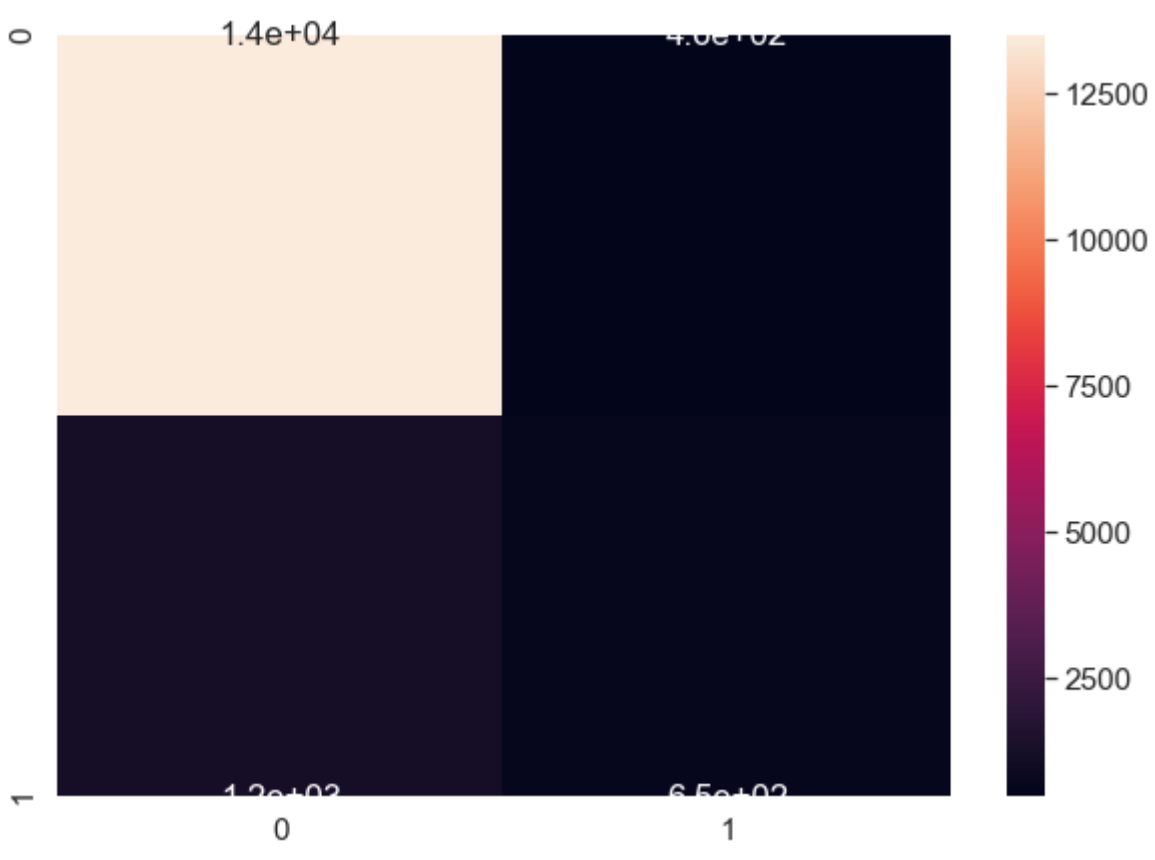
In [45]:
```
# 4.Prepare the data to train a model - check if data types are appropriate, get rid of the missing values etc
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.35, random_state = 0)
```

In [54]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
predrfc = classifier.predict_proba(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
Acc = accuracy_score(y_test, y_pred)
print("Accuracy score of Random Forest::: ",Acc)
cmrfc = pd.DataFrame(cm, columns=np.unique(y_test), index = np.unique(y_test))
cmrfc.index.name = 'Actual'
cmrfc.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm, annot=True)

plt.show()
```

```
[[13503   458]
 [ 1210   653]]
Accuracy score of Random Forest:::  0.8945904954499494
```

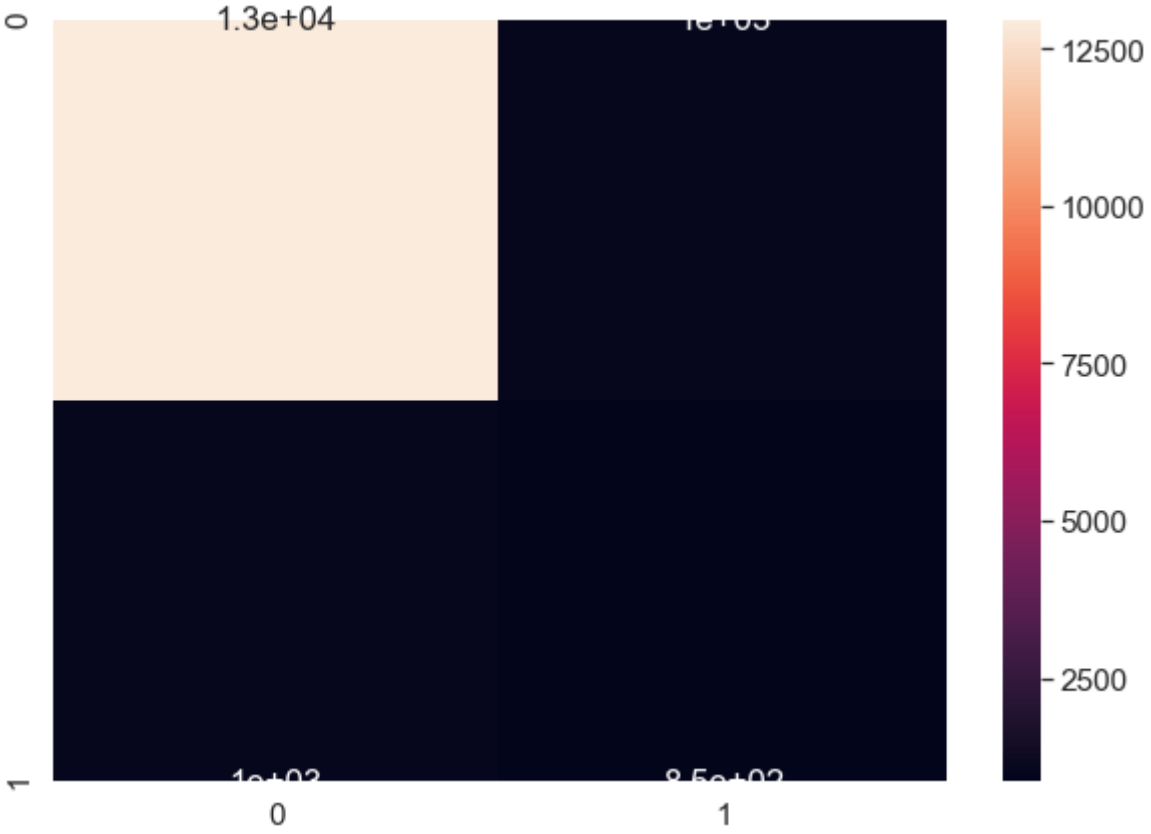


In [55]:
```
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier1.fit(X_train, y_train)

# Predicting the Test set results
y_pred1 = classifier1.predict(X_test)
preddtr = classifier.predict_proba(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred1)
print(cm1)
Acc = accuracy_score(y_test, y_pred1)
print("Accuracy score of DecisionTree::: ",Acc)
cmdtr = pd.DataFrame(cm1, columns=np.unique(y_test), index = np.unique(y_test))
cmdtr.index.name = 'Actual'
cmdtr.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm1, annot=True)

plt.show()
```

```
[[12966   995]
 [ 1011   852]]
Accuracy score of DecisionTree:::  0.8732305358948432
```



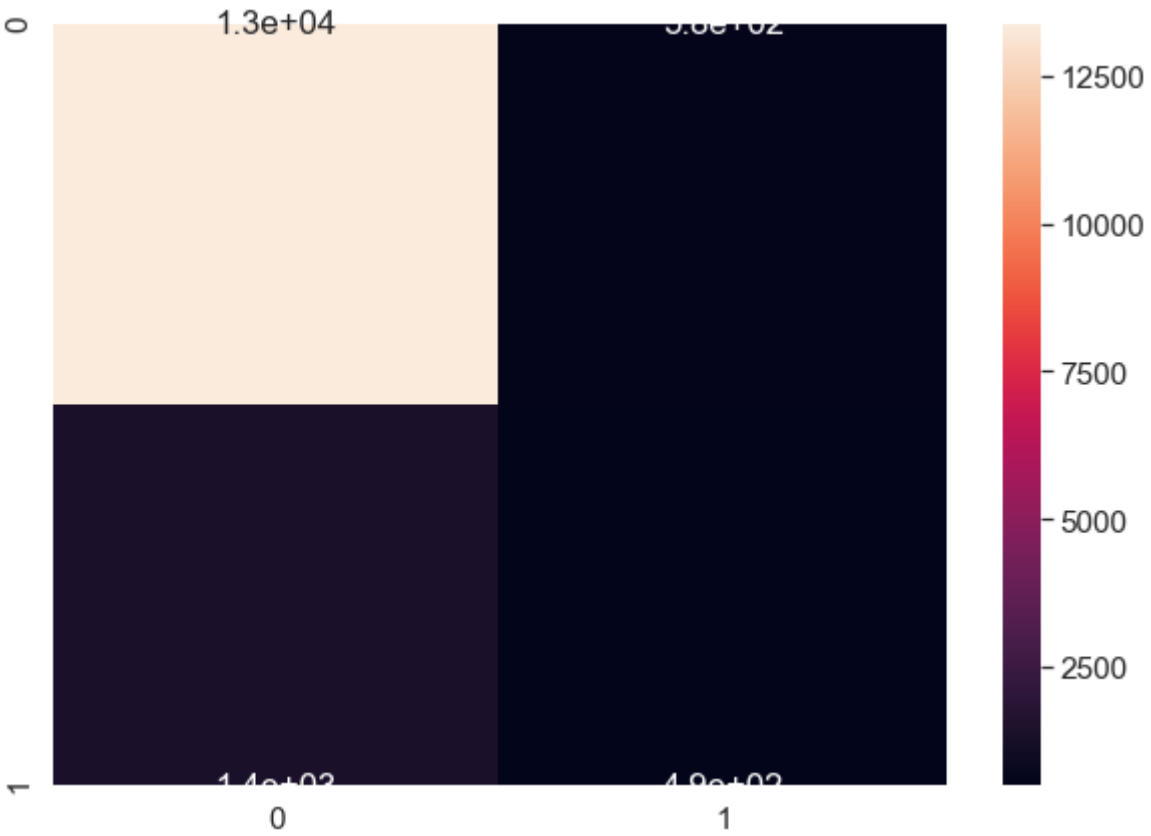
In [53]:
```
# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred2 = classifier.predict(X_test)
predknn = classifier.predict_proba(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm2 = confusion_matrix(y_test, y_pred2)
print(cm2)
Acc = accuracy_score(y_test, y_pred2)
print("Accuracy score of KNN::: ",Acc)
cmknn = pd.DataFrame(cm2, columns=np.unique(y_test), index = np.unique(y_test))
cmknn.index.name = 'Actual'
cmknn.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm2, annot=True)

plt.show()
```

```
[[13383   578]
 [ 1369   494]]
Accuracy score of KNN:::  0.8769590495449949
```



In [49]:
```
# 6.Build the ensemble models and compare the results with the base models.

final_pred = (predrfc+preddtr+predknn)/3
print(final_pred)
```

```
[[1.         0.        ]
 [0.46666667 0.53333333]
 [1.         0.        ]
 ...
 [0.93333333 0.06666667]
 [0.66666667 0.33333333]
 [1.         0.        ]]
```

In [ ]: