In [54]:
```
# 1.Importing the libraries
import sys
from ipykernel import kernelapp as app
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 [26]:
```
# 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 [55]:
```
# 3.a.Shape of the data

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

```
(45211, 17)
```

In [56]:
```
# 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 [57]:
```
# 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
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

[[  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]
```

In [58]:
```
# 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 [59]:
```
# 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)
```

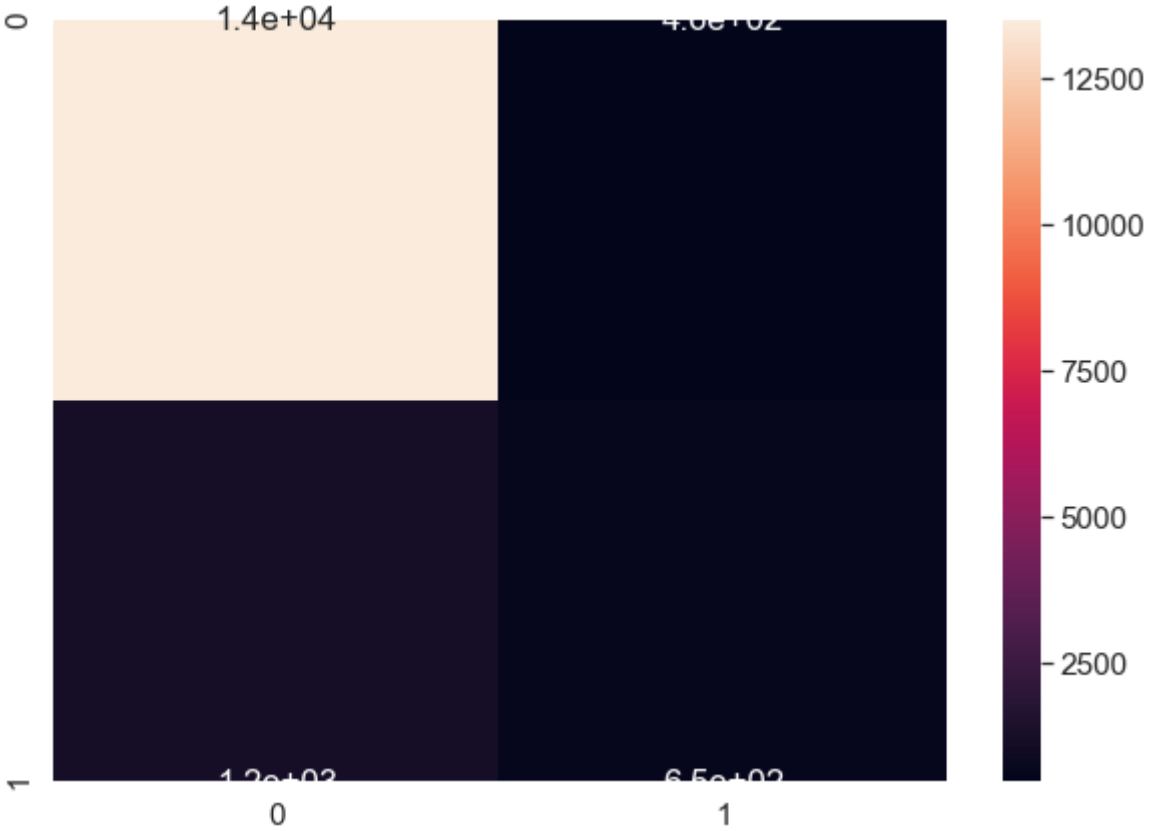
In [60]:
```
dropdata = drop_numerical_outliers(bankdata_final)
print(dropdata)
```

```
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 [61]:
```
# 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)

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)

#X_train.head()
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

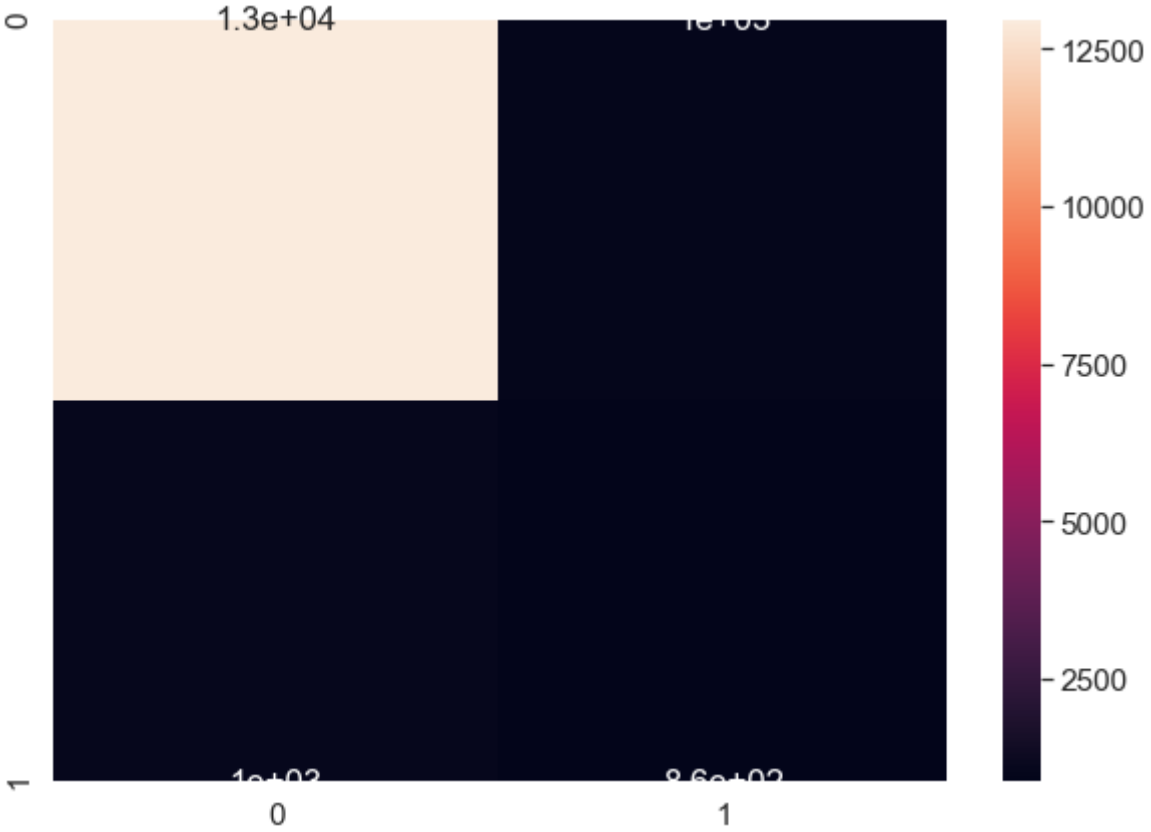
In [62]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# 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()
RFCCV = (cross_val_score(classifier, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

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



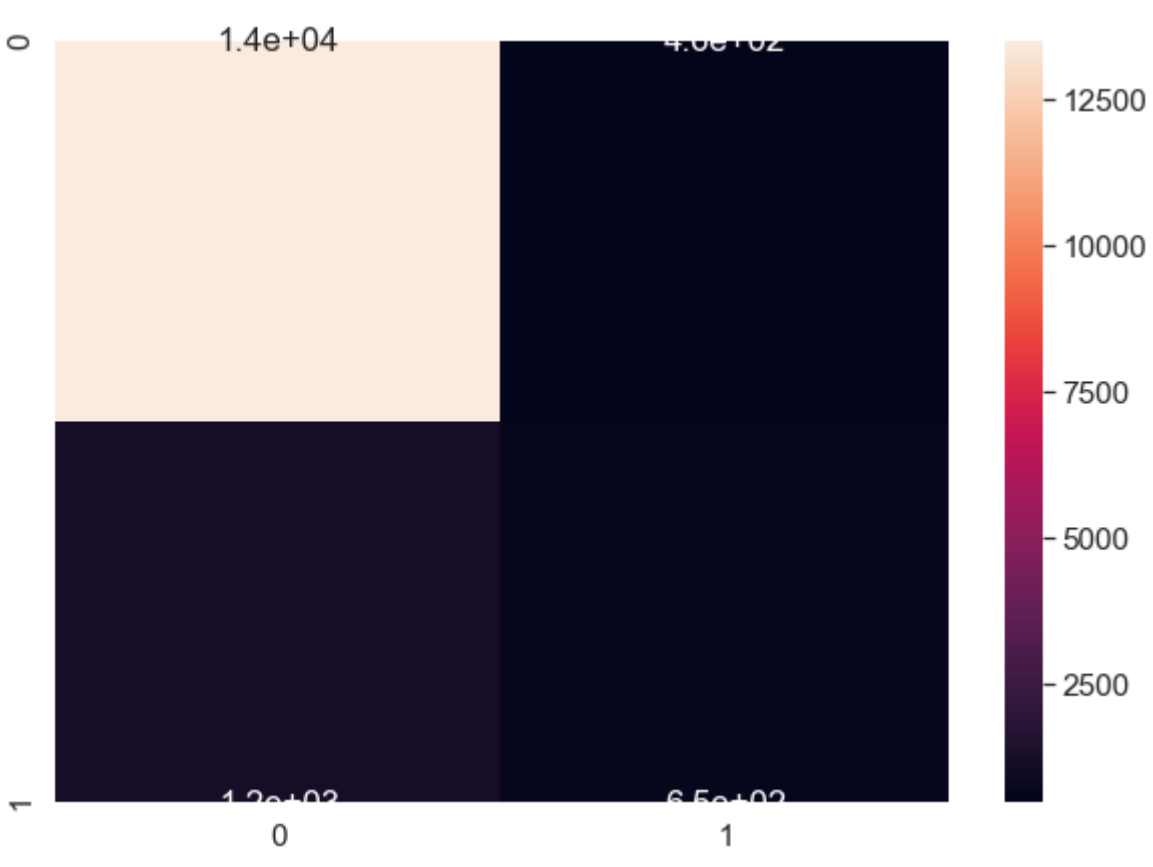
In [63]:
```
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier1.fit(X_train, y_train)

# 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()
DTREECV = (cross_val_score(classifier1, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
[[12965   996]
 [ 1006   857]]
Accuracy score of DecisionTree:::  0.8734833164812943
```



In [64]:
```
# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier2 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier2.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()
KNNCV = (cross_val_score(classifier2, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

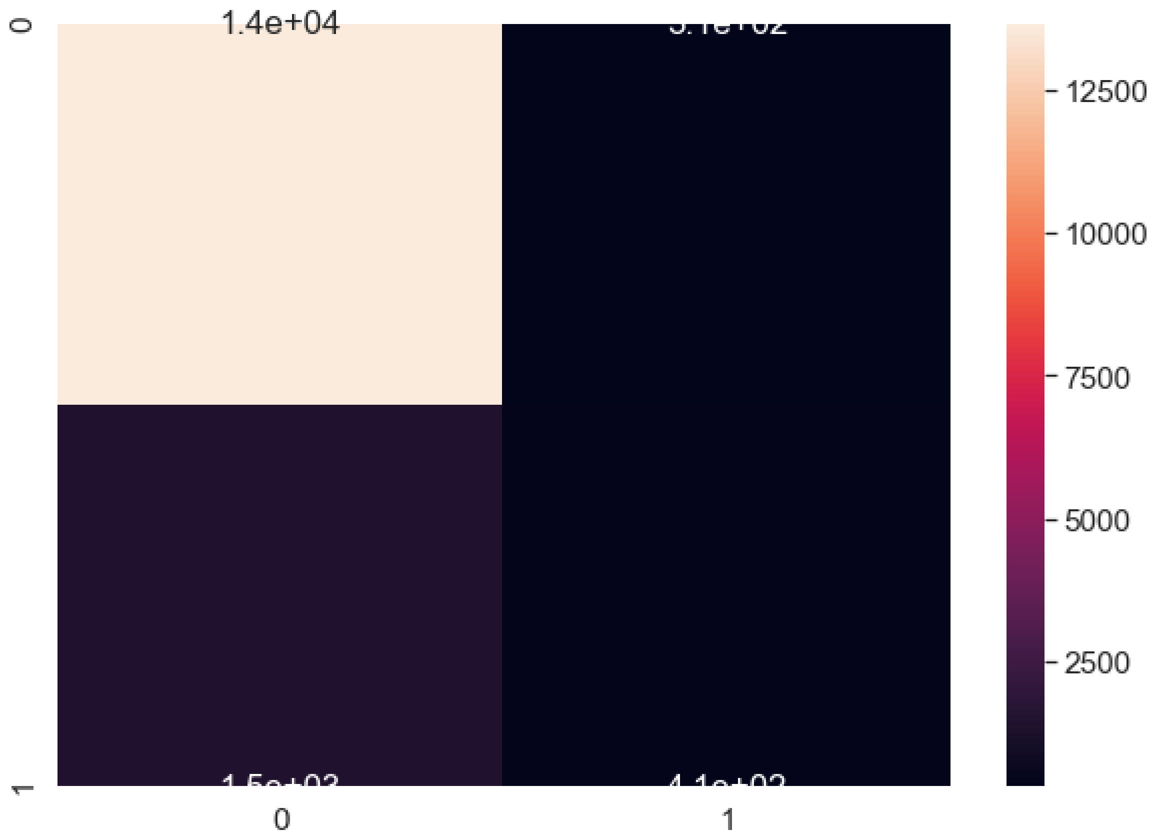
```
[[13503   458]
 [ 1210   653]]
Accuracy score of KNN:::  0.8945904954499494
```



In [65]:

```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
logpred = logmodel.predict(X_test)

cm3 = confusion_matrix(y_test, logpred)
print(cm3)
Acc = accuracy_score(y_test, logpred)
print("Accuracy score of Logistic::: ",Acc)
cmlog = pd.DataFrame(cm2, columns=np.unique(y_test), index = np.unique(y_test))
cmlog.index.name = 'Actual'
cmlog.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm3, annot=True)

plt.show()
LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

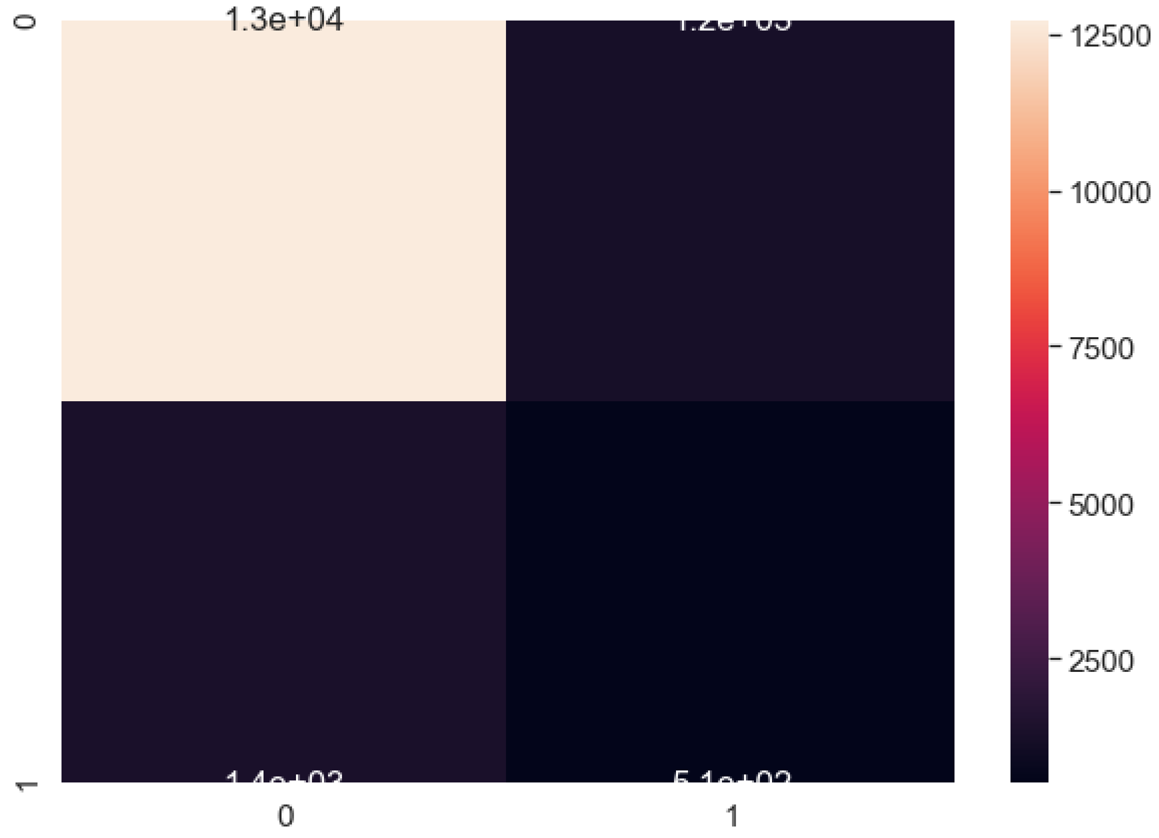
```
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

[[13654   307]
 [ 1453   410]]
Accuracy score of Logistic:::  0.8887765419615774
```




```
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [66]:
```
# Support vector machine Model
from sklearn.svm import SVC
svc= SVC(kernel = 'sigmoid')
svc.fit(X_train, y_train)
svcpred = svc.predict(X_test)
cm4 = confusion_matrix(y_test, svcpred)
print(cm4)
acc4 = round(accuracy_score(y_test, svcpred),2)*100
print(acc4)
#SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
cmsvc = pd.DataFrame(cm4, columns=np.unique(y_test), index = np.unique(y_test))
cmsvc.index.name = 'Actual'
cmsvc.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm4, annot=True)

plt.show()
SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

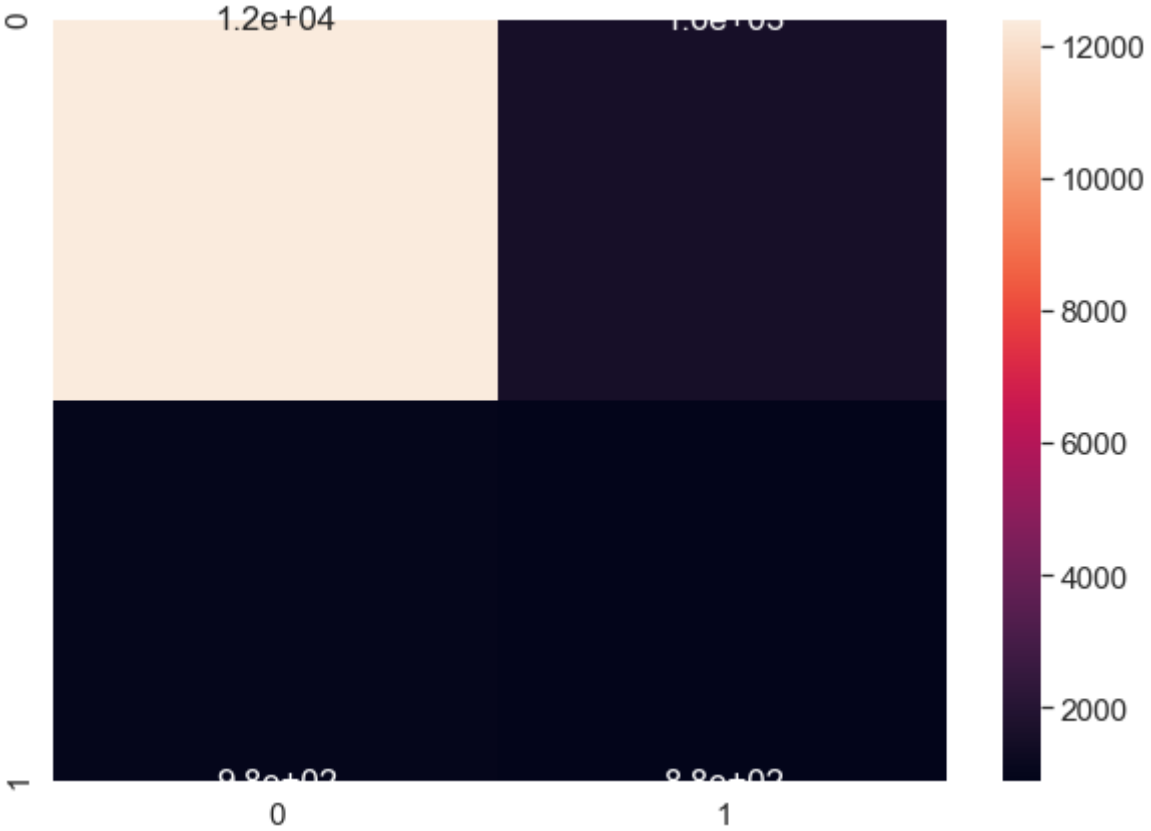
```
[[12720  1241]
 [ 1354   509]]
84.0
```




```
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Conda\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

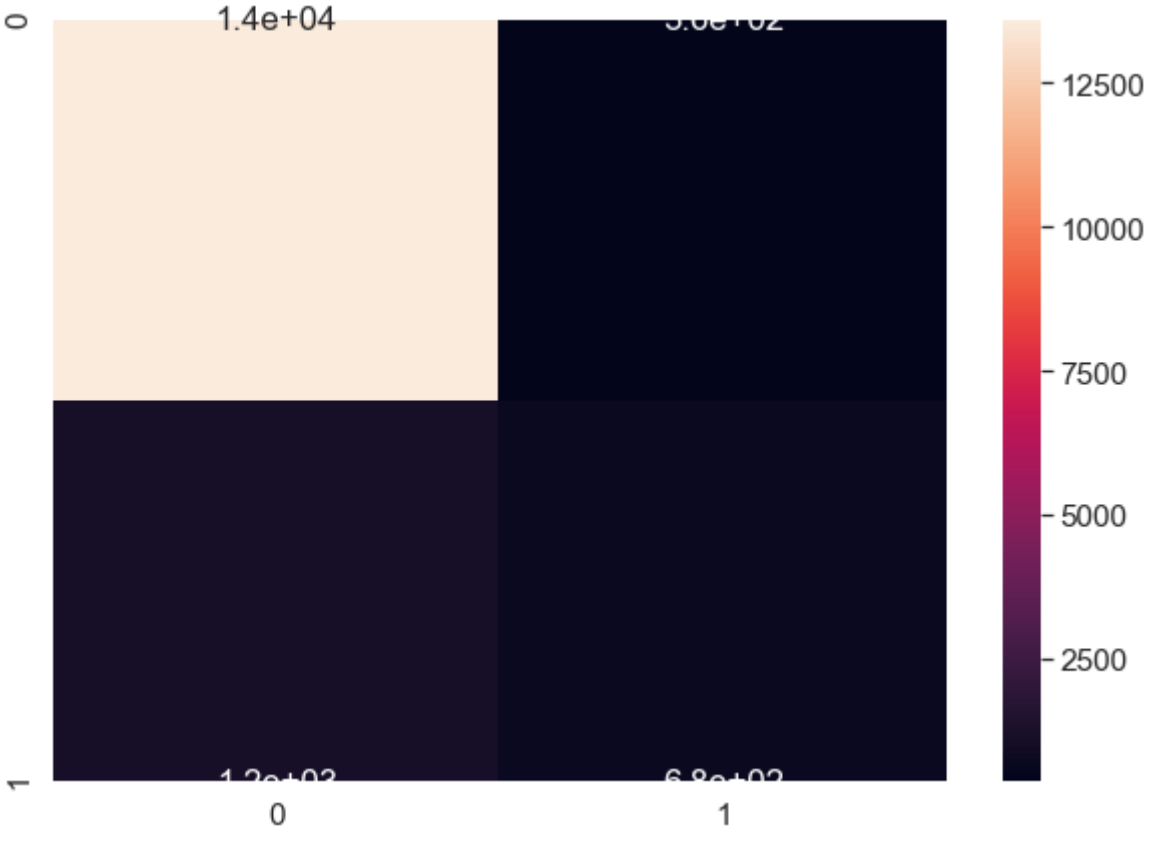
In [67]:
```
# Naive Bayes Classifier Model
from sklearn.naive_bayes import GaussianNB
gaussiannb= GaussianNB()
gaussiannb.fit(X_train, y_train)
gaussiannbpred = gaussiannb.predict(X_test)
probs = gaussiannb.predict(X_test)
cm5 = confusion_matrix(y_test, probs)
print(cm5)
acc5 = round(accuracy_score(y_test, probs),2)*100
print(acc5)
#SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
cmnv = pd.DataFrame(cm5, columns=np.unique(y_test), index = np.unique(y_test))
cmnv.index.name = 'Actual'
cmnv.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm5, annot=True)

plt.show()
GAUSIAN = (cross_val_score(gaussiannb, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
[[12386  1575]
 [  983   880]]
84.0
```



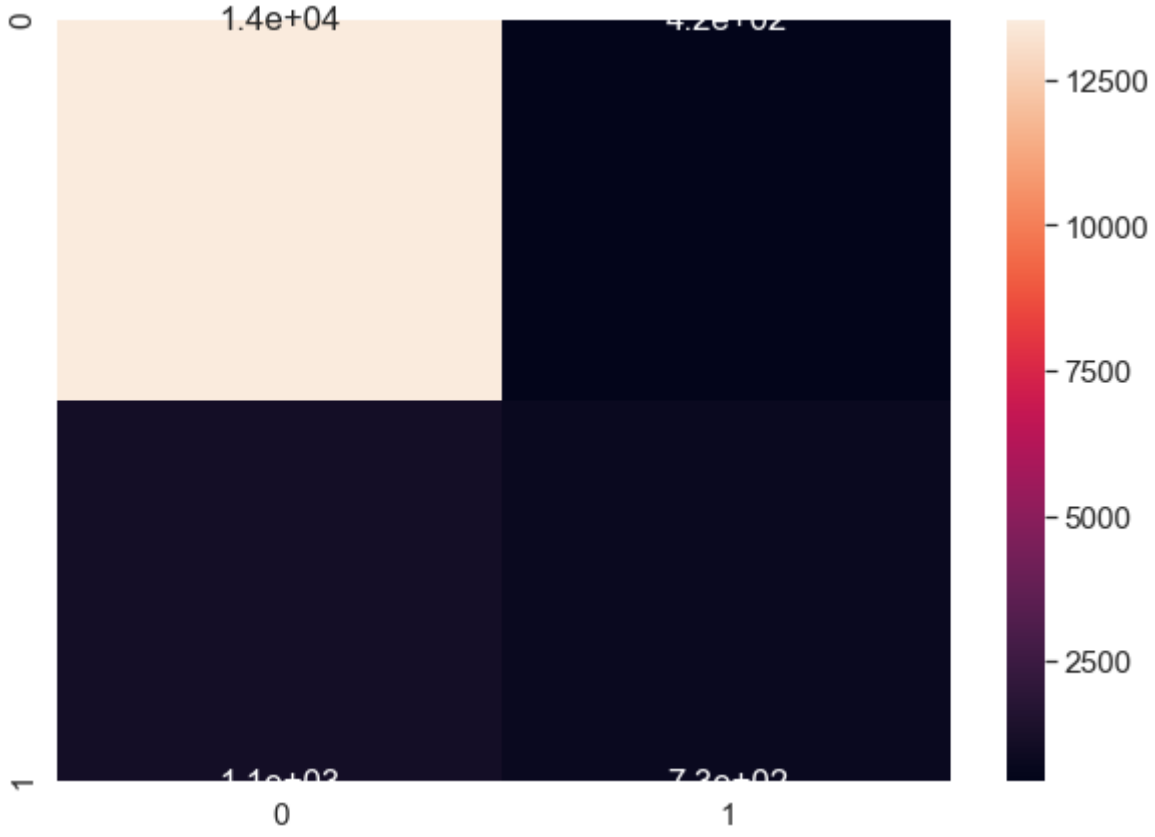
In [68]:
```
# XGBOOST Classifier Model
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
xgbprd = xgb.predict(X_test)

cm6 = confusion_matrix(y_test, xgbprd)
print(cm6)
acc6 = round(accuracy_score(y_test, xgbprd),2)*100
print(acc6)
#SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
cmxg = pd.DataFrame(cm6, columns=np.unique(y_test), index = np.unique(y_test))
cmxg.index.name = 'Actual'
cmxg.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm6, annot=True)

plt.show()
XGB = (cross_val_score(estimator = xgb, X = X_train, y = y_train, cv = 10).mean())
```

```
[[13596   365]
 [ 1184   679]]
90.0
```



In [69]:
```
# Gradient Boosting Model

from sklearn.ensemble import GradientBoostingClassifier
gbk = GradientBoostingClassifier()
gbk.fit(X_train, y_train)
gbkpred = gbk.predict(X_test)

cm7 = confusion_matrix(y_test, gbkpred)
print(cm7)
acc7 = round(accuracy_score(y_test, gbkpred),2)*100
print(acc7)
#SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
cmgbk = pd.DataFrame(cm7, columns=np.unique(y_test), index = np.unique(y_test))
cmgbk.index.name = 'Actual'
cmgbk.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(cm7, annot=True)

plt.show()
GBKCV = (cross_val_score(gbk, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
[[13537   424]
 [ 1131   732]]
90.0
```

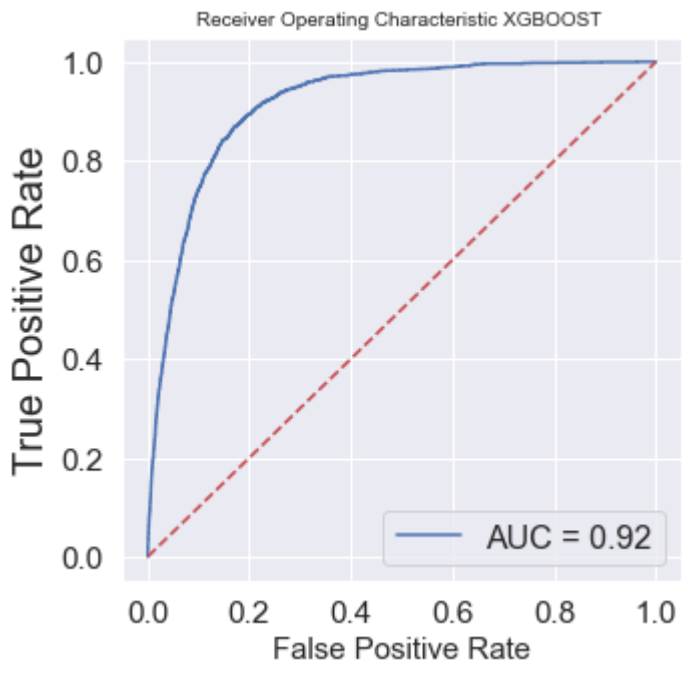


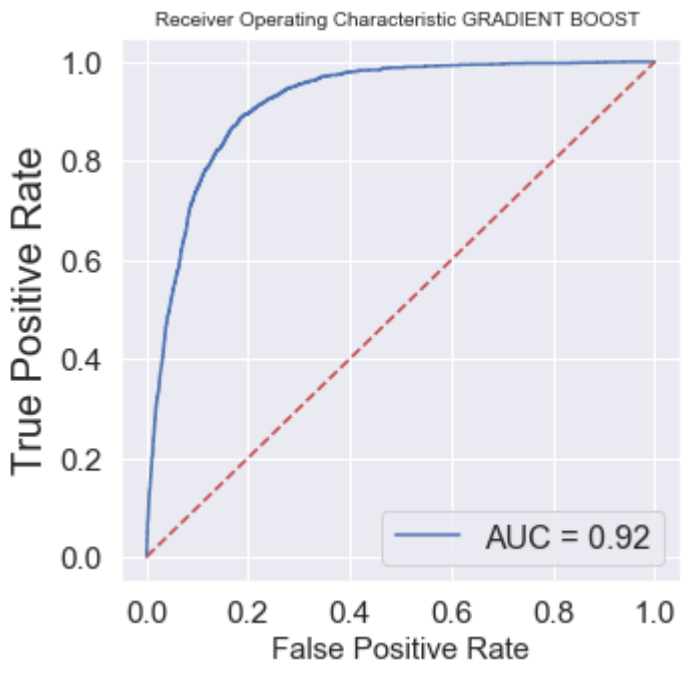
In [70]:
```
# Comparison of Different Model Performance

models = pd.DataFrame({
                'Models': ['Random Forest Classifier', 'Decision Tree Classifier', 'Support Vector Machine',
                           'K-Near Neighbors', 'Logistic Model', 'Gausian NB', 'XGBoost', 'Gradient Boosting'],
                'Score':  [RFCCV, DTREECV, SVCCV, KNNCV, LOGCV, GAUSIAN, XGB, GBKCV]})

models.sort_values(by='Score', ascending=False)
```

Out[70]:

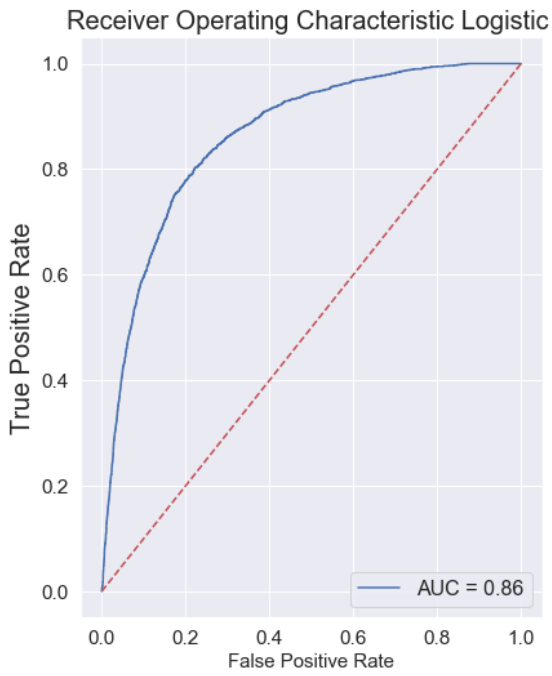
	Models	Score
7	Gradient Boosting	0.906217
6	XGBoost	0.905571
0	Random Forest Classifier	0.900093
3	K-Near Neighbors	0.894750
4	Logistic Model	0.892164
1	Decision Tree Classifier	0.874469
5	Gausian NB	0.843503
2	Support Vector Machine	0.834383

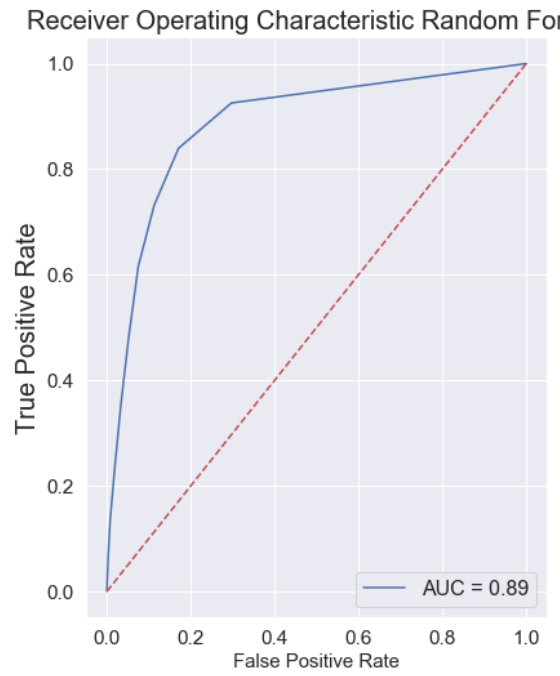
In [71]:
```
# XGBOOST ROC/ AUC , BEST MODEL
from sklearn import metrics
fig, (ax, ax1) = plt.subplots(nrows = 1, ncols = 2, figsize = (15,5))
probs = xgb.predict_proba(X_test)
preds = probs[:,1]
fprxgb, tprxgb, thresholdxgb = metrics.roc_curve(y_test, preds)
roc_aucxgb = metrics.auc(fprxgb, tprxgb)

ax.plot(fprxgb, tprxgb, 'b', label = 'AUC = %0.2f' % roc_aucxgb)
ax.plot([0, 1], [0, 1],'r--')
ax.set_title('Receiver Operating Characteristic XGBOOST ',fontsize=10)
ax.set_ylabel('True Positive Rate',fontsize=20)
ax.set_xlabel('False Positive Rate',fontsize=15)
ax.legend(loc = 'lower right', prop={'size': 16})

#Gradient
probs = gbk.predict_proba(X_test)
preds = probs[:,1]
fprgbk, tprgbk, thresholdgbk = metrics.roc_curve(y_test, preds)
roc_aucgbk = metrics.auc(fprgbk, tprgbk)

ax1.plot(fprgbk, tprgbk, 'b', label = 'AUC = %0.2f' % roc_aucgbk)
ax1.plot([0, 1], [0, 1],'r--')
ax1.set_title('Receiver Operating Characteristic GRADIENT BOOST ',fontsize=10)
ax1.set_ylabel('True Positive Rate',fontsize=20)
ax1.set_xlabel('False Positive Rate',fontsize=15)
ax1.legend(loc = 'lower right', prop={'size': 16})

plt.subplots_adjust(wspace=1)
```

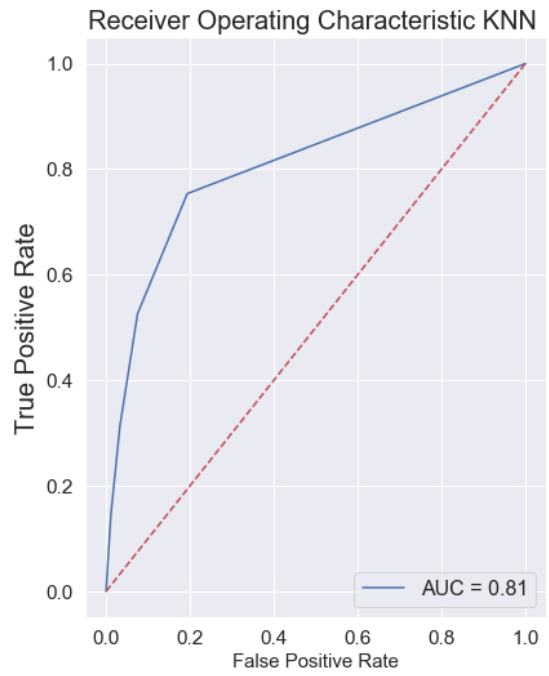


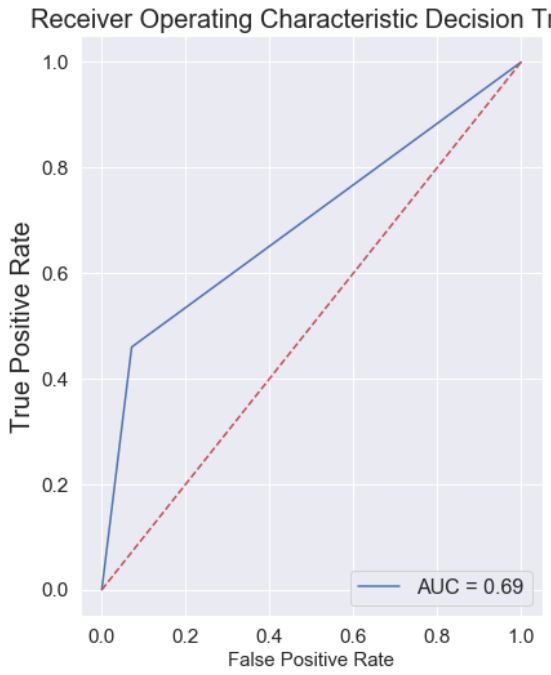


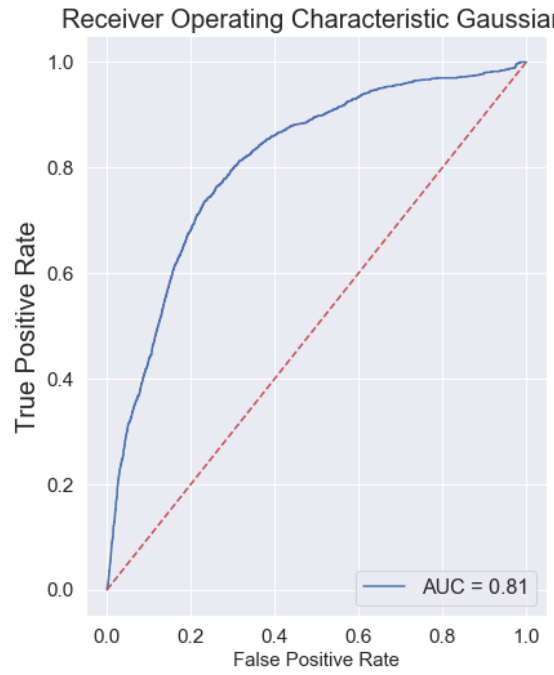
In [72]:
```
#fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(nrows = 2, ncols = 3, figsize = (15, 4))
fig, ax_arr = plt.subplots(nrows = 2, ncols = 3, figsize = (20,15))

#LOGMODEL
probs = logmodel.predict_proba(X_test)
preds = probs[:,1]
fprlog, tprlog, thresholdlog = metrics.roc_curve(y_test, preds)
roc_auclog = metrics.auc(fprlog, tprlog)

ax_arr[0,0].plot(fprlog, tprlog, 'b', label = 'AUC = %0.2f' % roc_auclog)
ax_arr[0,0].plot([0, 1], [0, 1],'r--')
ax_arr[0,0].set_title('Receiver Operating Characteristic Logistic ',fontsize=20)
ax_arr[0,0].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[0,0].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[0,0].legend(loc = 'lower right', prop={'size': 16})

#RANDOM FOREST --------------------
probs = classifier.predict_proba(X_test)
preds = probs[:,1]
fprrfc, tprrfc, thresholdrfc = metrics.roc_curve(y_test, preds)
roc_aucrfc = metrics.auc(fprrfc, tprrfc)

ax_arr[0,1].plot(fprrfc, tprrfc, 'b', label = 'AUC = %0.2f' % roc_aucrfc)
ax_arr[0,1].plot([0, 1], [0, 1],'r--')
ax_arr[0,1].set_title('Receiver Operating Characteristic Random Forest ',fontsize=20)
ax_arr[0,1].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[0,1].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[0,1].legend(loc = 'lower right', prop={'size': 16})

#KNN----------------------
probs = classifier2.predict_proba(X_test)
preds = probs[:,1]
fprknn, tprknn, thresholdknn = metrics.roc_curve(y_test, preds)
roc_aucknn = metrics.auc(fprknn, tprknn)

ax_arr[0,2].plot(fprknn, tprknn, 'b', label = 'AUC = %0.2f' % roc_aucknn)
ax_arr[0,2].plot([0, 1], [0, 1],'r--')
ax_arr[0,2].set_title('Receiver Operating Characteristic KNN ',fontsize=20)
ax_arr[0,2].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[0,2].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[0,2].legend(loc = 'lower right', prop={'size': 16})

#DECISION TREE --------------------
probs = classifier1.predict_proba(X_test)
preds = probs[:,1]
fprdtree, tprdtree, thresholddtree = metrics.roc_curve(y_test, preds)
roc_aucdtree = metrics.auc(fprdtree, tprdtree)

ax_arr[1,0].plot(fprdtree, tprdtree, 'b', label = 'AUC = %0.2f' % roc_aucdtree)
ax_arr[1,0].plot([0, 1], [0, 1],'r--')
ax_arr[1,0].set_title('Receiver Operating Characteristic Decision Tree ',fontsize=20)
ax_arr[1,0].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[1,0].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[1,0].legend(loc = 'lower right', prop={'size': 16})

#GAUSSIAN --------------------
probs = gaussiannb.predict_proba(X_test)
preds = probs[:,1]
fprgau, tprgau, thresholdgau = metrics.roc_curve(y_test, preds)
roc_aucgau = metrics.auc(fprgau, tprgau)

ax_arr[1,1].plot(fprgau, tprgau, 'b', label = 'AUC = %0.2f' % roc_aucgau)
ax_arr[1,1].plot([0, 1], [0, 1],'r--')
ax_arr[1,1].set_title('Receiver Operating Characteristic Gaussian ',fontsize=20)
ax_arr[1,1].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[1,1].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[1,1].legend(loc = 'lower right', prop={'size': 16})

#ALL PLOTS ----------------------------------
ax_arr[1,2].plot(fprgau, tprgau, 'b', label = 'Gaussian', color='black')
ax_arr[1,2].plot(fprdtree, tprdtree, 'b', label = 'Decision Tree', color='blue')
ax_arr[1,2].plot(fprknn, tprknn, 'b', label = 'Knn', color='brown')
ax_arr[1,2].plot(fprrfc, tprrfc, 'b', label = 'Random Forest', color='green')
ax_arr[1,2].plot(fprlog, tprlog, 'b', label = 'Logistic', color='grey')
ax_arr[1,2].set_title('Receiver Operating Comparison ',fontsize=20)
ax_arr[1,2].set_ylabel('True Positive Rate',fontsize=20)
ax_arr[1,2].set_xlabel('False Positive Rate',fontsize=15)
ax_arr[1,2].legend(loc = 'lower right', prop={'size': 16})

plt.subplots_adjust(wspace=0.2)
plt.tight_layout()
```

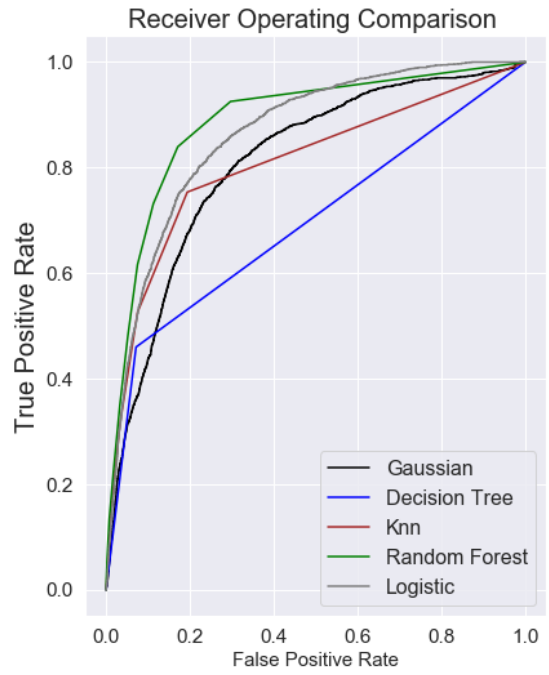

Receiver Operating Characteristic Logistic
True Positive Rate
False Positive Rate
AUC = 0.86


Receiver Operating Characteristic Random Forest
True Positive Rate
False Positive Rate
AUC = 0.89


Receiver Operating Characteristic KNN
True Positive Rate
False Positive Rate
AUC = 0.81


Receiver Operating Characteristic Decision Tree
True Positive Rate
False Positive Rate
AUC = 0.69


Receiver Operating Characteristic Gaussian
True Positive Rate
False Positive Rate
AUC = 0.81


Receiver Operating Comparison
True Positive Rate
False Positive Rate
Gaussian
Decision Tree
Knn
Random Forest
Logistic

In [ ]: