In [1]:

```
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```

In [2]:

```
#importing all neccesary packages
#setting the style and colour of the plot to be created
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE, f_regression
from sklearn.linear_model import (LinearRegression)
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

In [3]:

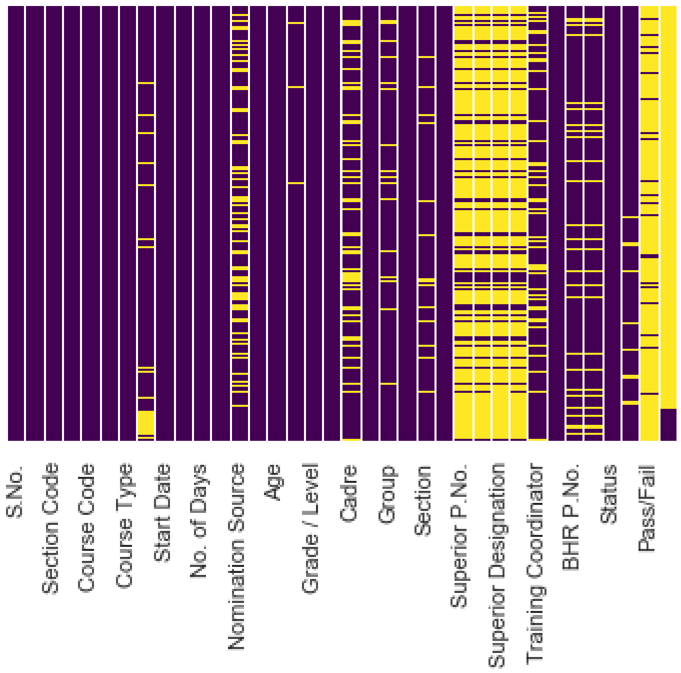
```
#accessing all datasets and storing them
dataset1= "desktop\Att18.xlsx"
dataset2="desktop\Att19.xlsx"
dataset3="desktop\Drop18.xlsx"
dataset4="desktop\Drop19.xlsx"
#reading the datasets using pandas and storing in different dataframes
df1= pd.read_excel(dataset1)
df2= pd.read_excel(dataset2)
df3= pd.read_excel(dataset3)
df4= pd.read_excel(dataset4)
#joining all dataframes into one
dataframe=[df1,df2,df3,df4]
df=pd.concat(dataframe, ignore_index=True, sort =False)
```

In [4]:
```
df.isnull().sum()
```

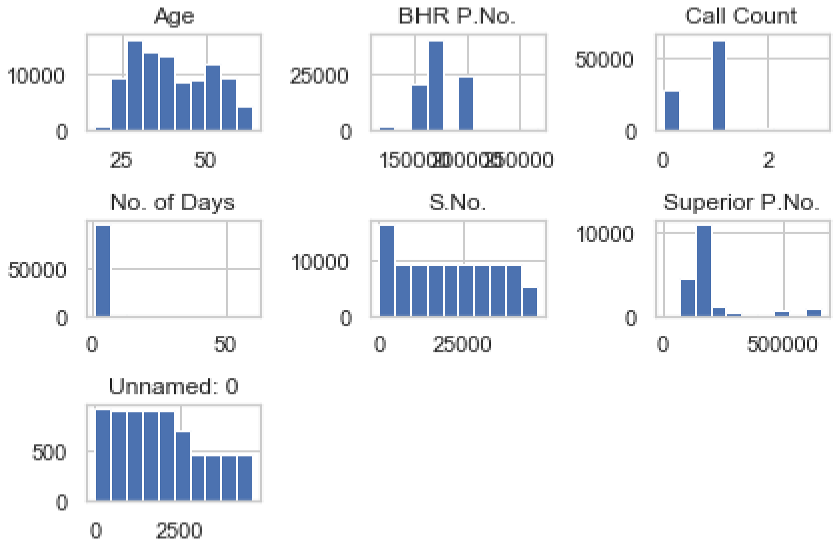
Out[4]:
```
S.No.                       0
Expert Group                0
Section Code                0
Reference No.               0
Course Code                 0
Course Description          0
Course Type                 0
Program Type             9294
Start Date                  0
End Date                    0
No. of Days                 0
Agency                      0
Nomination Source       23107
Gender                      0
Age                         1
Designation              1166
Grade / Level             151
Category                    0
Cadre                   19425
Executive Head             25
Group                    4133
Department                 41
Section                  5830
Employee Location           0
Superior P.No.          78964
Superior Name           78964
Superior Designation    78991
Superior level          78964
Training Coordinator    18362
Program Director           11
BHR P.No.                9773
BHR Name                 9773
Status                      0
Call Count               3911
Pass/Fail               87178
Unnamed: 0              90871
dtype: int64
```

In [5]:
```
#showing the attributes that have missing data
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

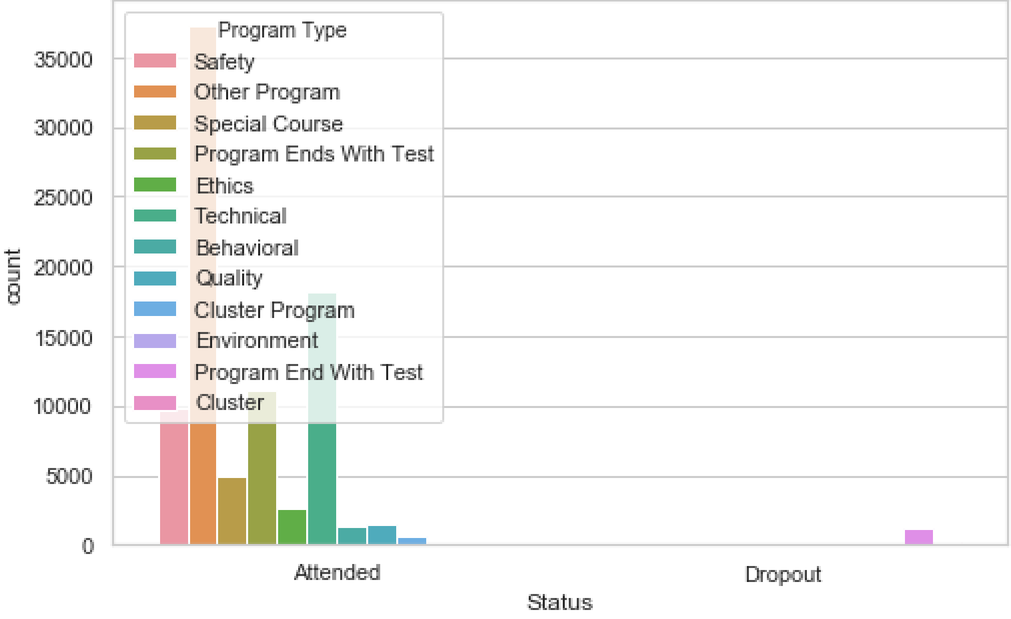
Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b7ecf10be0>
```



In [6]:
```
df.hist()
plt.tight_layout()
plt.show()
```

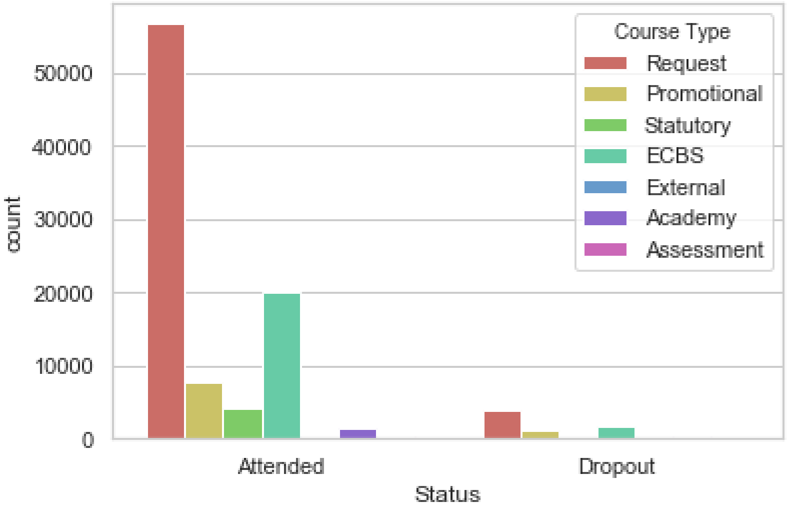


In [7]:
```
#count-plot of people who attended based on course type
plt.figure(figsize=(8,5)) # this creates a figure 8 inch wide, 4 inch high
sns.countplot(x='Status', hue='Program Type', data=df)
plt.show()
```

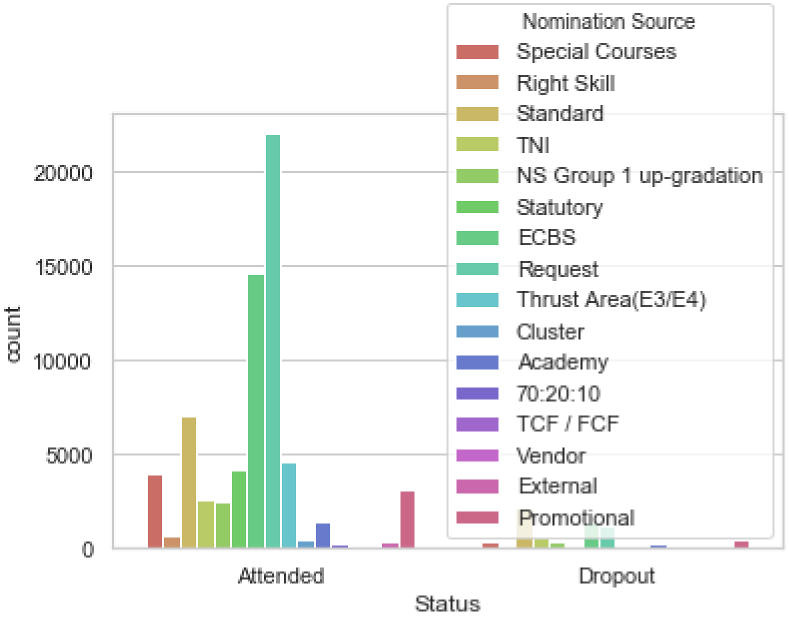


In [8]:
```
#count-plot of people who attended based on course type
sns.countplot(x='Status', hue='Course Type', data=df, palette='hls')
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b7eda77940>
```



```
In [9]: sns.countplot(x='Status', hue='Nomination Source', data=df, palette='hls')

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1b7ee47f1d0>
```

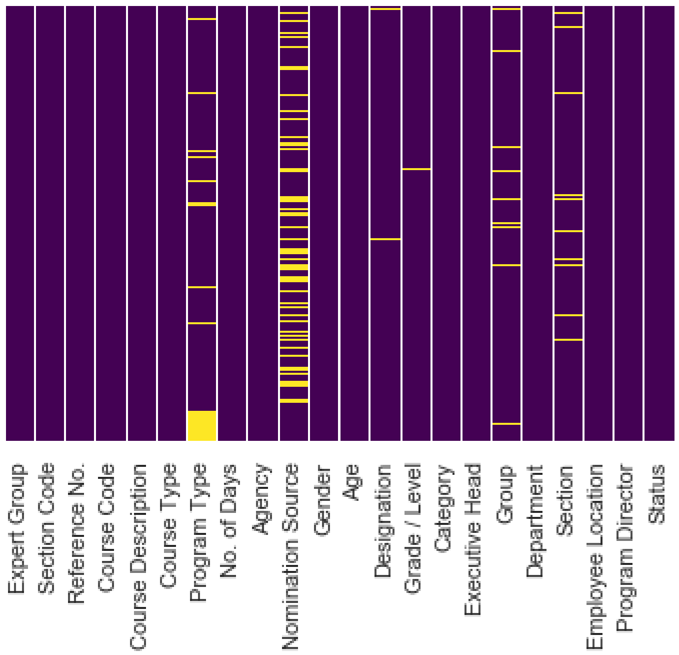


```
In [10]: to_drop= ['S.No.','Start Date','End Date','Superior Designation','Superior lev
         el', 'Training Coordinator','Cadre','Superior P.No.', 'Superior Name', 'BHR P.
         No.', 'BHR Name', 'Pass/Fail', 'Unnamed: 0', 'Call Count']
         df.drop(to_drop, inplace=True, axis=1)
```

```
In [11]: #df.drop(['Course Type', 'Program Type', 'Agency', 'Gender', 'Category'], inpl
         ace=True, axis=1)
```

In [12]:
```
#showing the attributes that have missing data
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b7eda6a6a0>
```



In [13]:
```
df.dtypes
```

Out[13]:
```
Expert Group            object
Section Code            object
Reference No.           object
Course Code             object
Course Description      object
Course Type             object
Program Type            object
No. of Days              int64
Agency                  object
Nomination Source       object
Gender                  object
Age                    float64
Designation             object
Grade / Level           object
Category                object
Executive Head          object
Group                   object
Department              object
Section                 object
Employee Location       object
Program Director        object
Status                  object
dtype: object
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Expert Group               0
Section Code               0
Reference No.              0
Course Code                0
Course Description         0
Course Type                0
Program Type            9294
No. of Days                0
Agency                     0
Nomination Source      23107
Gender                     0
Age                        1
Designation             1166
Grade / Level            151
Category                   0
Executive Head            25
Group                   4133
Department                41
Section                 5830
Employee Location          0
Program Director          11
Status                     0
dtype: int64
```

In [15]:
```
le = LabelEncoder()
#use of labeo encoder
```

In [16]:
```
df['Program Type'] = le.fit_transform(df['Program Type'].astype(str))
df['Designation'] = le.fit_transform(df['Designation'].astype(str))
df['Grade / Level'] = le.fit_transform(df['Grade / Level'].astype(str))
df['Executive Head'] = le.fit_transform(df['Executive Head'].astype(str))
df['Group'] = le.fit_transform(df['Group'].astype(str))
df['Department'] = le.fit_transform(df['Department'].astype(str))
df['Section'] = le.fit_transform(df['Section'].astype(str))
df['Program Director'] = le.fit_transform(df['Program Director'].astype(str))
df['Nomination Source'] = le.fit_transform(df['Nomination Source'].astype(str
))
```

Using Label Encoder to convert the categorical data to Numeric Data

In [17]:

```
# apply "le.fit_transform"
df_encoded = df.apply(le.fit_transform)
print(df_encoded)
```

```
       Expert Group  Section Code  Reference No.  Course Code  \
0                10            10           2712          568   
1                10            10           2712          568   
2                10            10           2712          568   
3                10            10           2712          568   
4                10            26           3826          862   
...             ...           ...            ...          ...   
98006             8             5           1549          218   
98007             0             4           4665         1023   
98008             0             2           3599          802   
98009             4            13           4773         1085   
98010             4            13           4759         1083   

       Course Description  Course Type  Program Type  No. of Days  Agency  \
0                     311            5             9            0       9   
1                     311            5             9            0       9   
2                     311            5             9            0       9   
3                     311            5             9            0       9   
4                     706            5             5            1      12   
...                   ...          ...           ...          ...     ...   
98006                 271            0            12            4      11   
98007                 958            0            12            1      11   
98008                 942            0            12            2      11   
98009                 776            2             6            0      11   
98010                  91            2            12            0      11   

       Nomination Source  ...  Designation  Grade / Level  Category  \
0                      9  ...         3292            183         2   
1                      9  ...         3292            102         1   
2                      9  ...          517             29         2   
3                      9  ...         3292            193         2   
4                      8  ...         3371            268         2   
...                  ...  ...          ...            ...       ...   
98006                  1  ...         4276             89         0   
98007                  1  ...         4231             89         0   
98008                  1  ...         2405             91         0   
98009                  3  ...         1786            231         2   
98010                  3  ...         1786            231         2   

       Executive Head  Group  Department  Section  Employee Location  \
0                  18     37         142      935                 32   
1                  18     37           1      931                 32   
2                  18     78          48      934                 32   
3                  18     37         640      935                 32   
4                  18     27         660      743                 62   
...               ...    ...         ...      ...                ...   
98006              20     48         300      514                 30   
98007              20     22         652      598                 30   
98008              20     22         368      493                 30   
98009              17     23          77      194                 29   
98010              17     23         650      506                 29   

       Program Director  Status  
0                    66       0  
1                    66       0  
2                    66       0  
3                    66       0  
```

```
4                   108       0
...                 ...     ...
98006               163       1
98007               238       1
98008               199       1
98009               122       1
98010               226       1

[98011 rows x 22 columns]
```

In [18]:

```
df = df_encoded.reindex(np.random.permutation(df_encoded.index))
df.head
```

```
Out[18]: <bound method NDFrame.head of        Expert Group  Section Code  Reference N
o.  Course Code  \
1517              1            24           4574         1000
59230             9            23             14            8
20191             4            14           1897          321
73425             2             9            495          119
28333             1            24           3288          688
...             ...           ...            ...          ...
8230              9            23            921          193
67571             1            24            675          182
12952             9            23           1716          265
88186             9            23           1469          200
40393             9            23           1692          247

       Course Description  Course Type  Program Type  No. of Days  Agency  \
1517                  992            2             5            0      11
59230                 214            2            11            0       0
20191                 273            5             5            0      11
73425                 227            5             5            0      11
28333                1078            2            11            0      11
...                   ...          ...           ...          ...     ...
8230                  295            5             5            0       3
67571                 320            4             7            2      11
12952                1046            5             8            0       1
88186                1073            0             5            0       4
40393                 435            5             9            0      10

       Nomination Source  ...  Designation  Grade / Level  Category  \
1517                   3  ...         4962            161         2
59230                 16  ...         1761             99         1
20191                 10  ...         3050            229         2
73425                 16  ...         3629             89         0
28333                  3  ...         3391            215         2
...                  ...  ...          ...            ...       ...
8230                  16  ...         3277             30         2
67571                 11  ...          627            209         2
12952                  9  ...         4184             89         0
88186                  3  ...         3157             89         0
40393                 16  ...         4269             89         0

       Executive Head  Group  Department  Section  Employee Location  \
1517               20     80         170      124                 30
59230              22      4         676     1098                 39
20191              17     13         589      637                 29
73425              14     88          78     1036                 30
28333              23     86         616      470                 36
...               ...    ...         ...      ...                ...
8230               18     38         196      456                 32
67571              21     46         692      258                 30
12952              21     45          72      802                 63
88186               6     88         184     1454                 44
40393              20     48         369      113                 30

       Program Director  Status
1517                 46       0
59230               149       0
20191               240       0
```

```
73425                  54       0
28333                 160       0
...                   ...     ...
8230                  253       0
67571                 109       0
12952                  53       0
88186                 227       0
40393                 185       0

[98011 rows x 22 columns]>
```

Logistic Regression Model

In [42]:

```
x = df.drop('Status', axis=1)
y = df['Status']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, rand
om_state=1)
#test and train sets created to be tested
logmodel = LogisticRegression()
logmodel.fit(x_train, y_train)
#training my model using train sets
```

Out[42]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [20]:

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
```

The classification report displays the Precision, Recall , F1 and Support scores for the model.

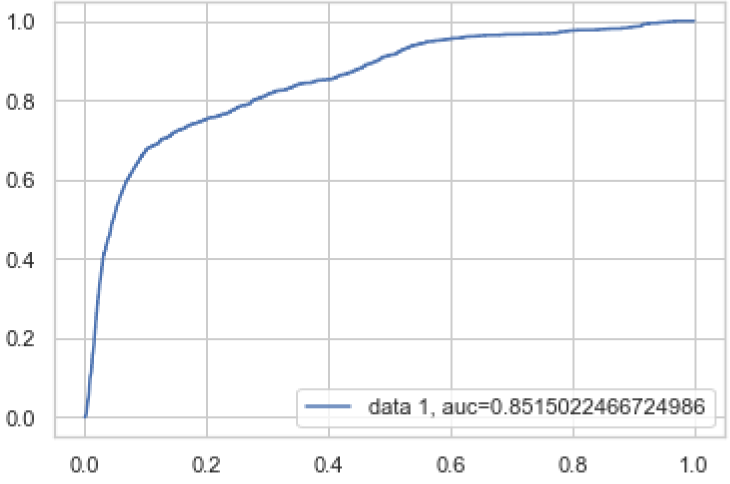
In [21]:

```
#Precision score means the the level up-to which the prediction made by the mo
del is precise.
#Recall is the amount up-to which the model can predict the outcome.
#F1 and Support scores are the amount of data tested for the predictions.
predictions = logmodel.predict(x_test)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(accuracy_score(y_test, predictions))
print("Accuracy:",metrics.accuracy_score(y_test, predictions))
print("Precision:",metrics.precision_score(y_test, predictions))
print("Recall:",metrics.recall_score(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     29972
           1       0.38      0.03      0.06      2372

    accuracy                           0.93     32344
   macro avg       0.65      0.51      0.51     32344
weighted avg       0.89      0.93      0.89     32344

[[29852   120]
 [ 2299    73]]
0.9252102399208508
Accuracy: 0.9252102399208508
Precision: 0.37823834196891193
Recall: 0.030775716694772345
```

In [22]:

```
y_pred_proba = logmodel.predict_proba(x_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



In [23]:
```
rfe = RFE(logmodel, n_features_to_select= None)
rfe = rfe.fit(x, y)
print(rfe.support_)
print(rfe.ranking_)
f = rfe.get_support(1) #the most important features
X = df[df.columns[f]] # final features
```

```
[ True False False False False  True  True  True  True  True  True False
 False False  True  True False False False  True False]
[ 1  6 10  3  5  1  1  1  1  1  1  2 11  9  1  1  4  7  8  1 12]
```

In [24]:
```
temp = pd.Series(rfe.support_,index = x.columns)
selected_features_rfe = temp[temp==True].index
print(selected_features_rfe)
```

```
Index(['Expert Group', 'Course Type', 'Program Type', 'No. of Days', 'Agenc
y',
       'Nomination Source', 'Gender', 'Category', 'Executive Head',
       'Employee Location'],
      dtype='object')
```

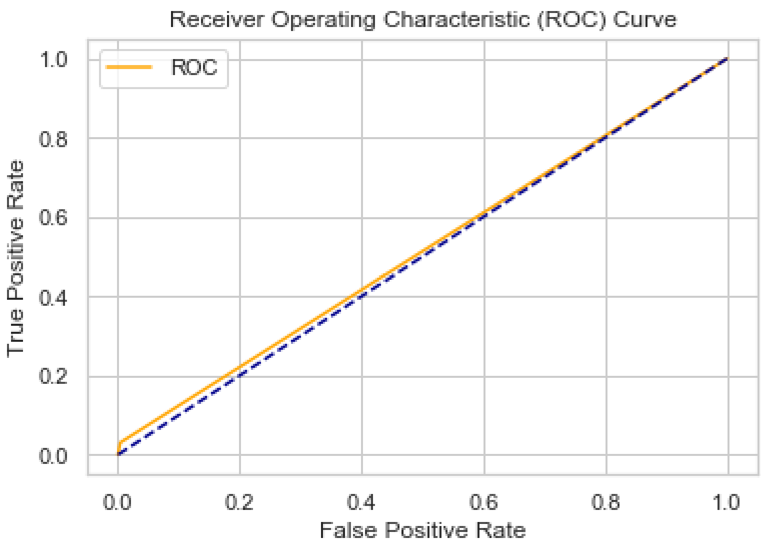
In [25]:
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

In [26]:
```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

In [27]:
```
auc = roc_auc_score(y_test, predictions)
print('AUC: %.2f' % auc)
```

```
AUC: 0.51
```

In [28]:
```
fpr, tpr, thresholds = roc_curve(y_test, predictions)
plot_roc_curve(fpr, tpr)
```



Decision Tree Model

In [29]:
```
from sklearn import tree
model= tree.DecisionTreeClassifier()
```

In [30]:
```
#Defining Features and lables
features= list(df.columns)
features.remove('Status')
```

In [31]:
```
X = df.drop('Status', axis=1)
Y = df['Status']
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2, ra
ndom_state = 100)
```

In [32]:
```
model.fit(X_train, y_train)
```

Out[32]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [33]:
```
model.score(X_test, y_test)
```

Out[33]:
```
0.9958169667907973
```

In [34]:
```
predictions_2 = model.predict(X_test)
print(classification_report(y_test, predictions_2))
print(confusion_matrix(y_test, predictions_2))
print(accuracy_score(y_test, predictions_2))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     18147
           1       0.97      0.98      0.97      1456

    accuracy                           1.00     19603
   macro avg       0.98      0.99      0.98     19603
weighted avg       1.00      1.00      1.00     19603

[[18101    46]
 [   36  1420]]
0.9958169667907973
```

In [35]:
```
print("Accuracy:",metrics.accuracy_score(y_test, predictions_2))
print("Precision:",metrics.precision_score(y_test, predictions_2))
print("Recall:",metrics.recall_score(y_test, predictions_2))
```

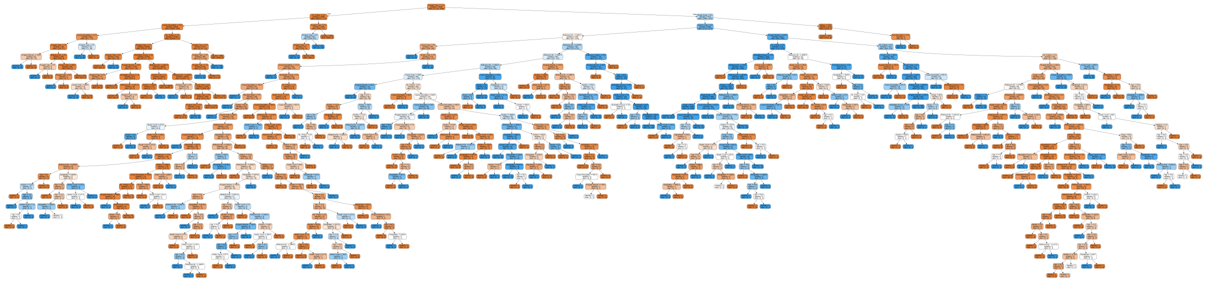
```
Accuracy: 0.9958169667907973
Precision: 0.9686221009549796
Recall: 0.9752747252747253
```

In [36]:
```
from IPython.display import Image
from sklearn.externals.six import StringIO
import pydotplus
```

```
C:\Users\Runa\Anaconda3\lib\site-packages\sklearn\externals\six.py:31: Deprec
ationWarning: The module is deprecated in version 0.21 and will be removed in
version 0.23 since we've dropped support for Python 2.7. Please rely on the o
fficial version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

In [37]:
```
dot_data = StringIO()
tree.export_graphviz(model,
                     out_file = dot_data,
                     feature_names = features,
                     filled=True, rounded=True,
                     impurity=False)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[37]:

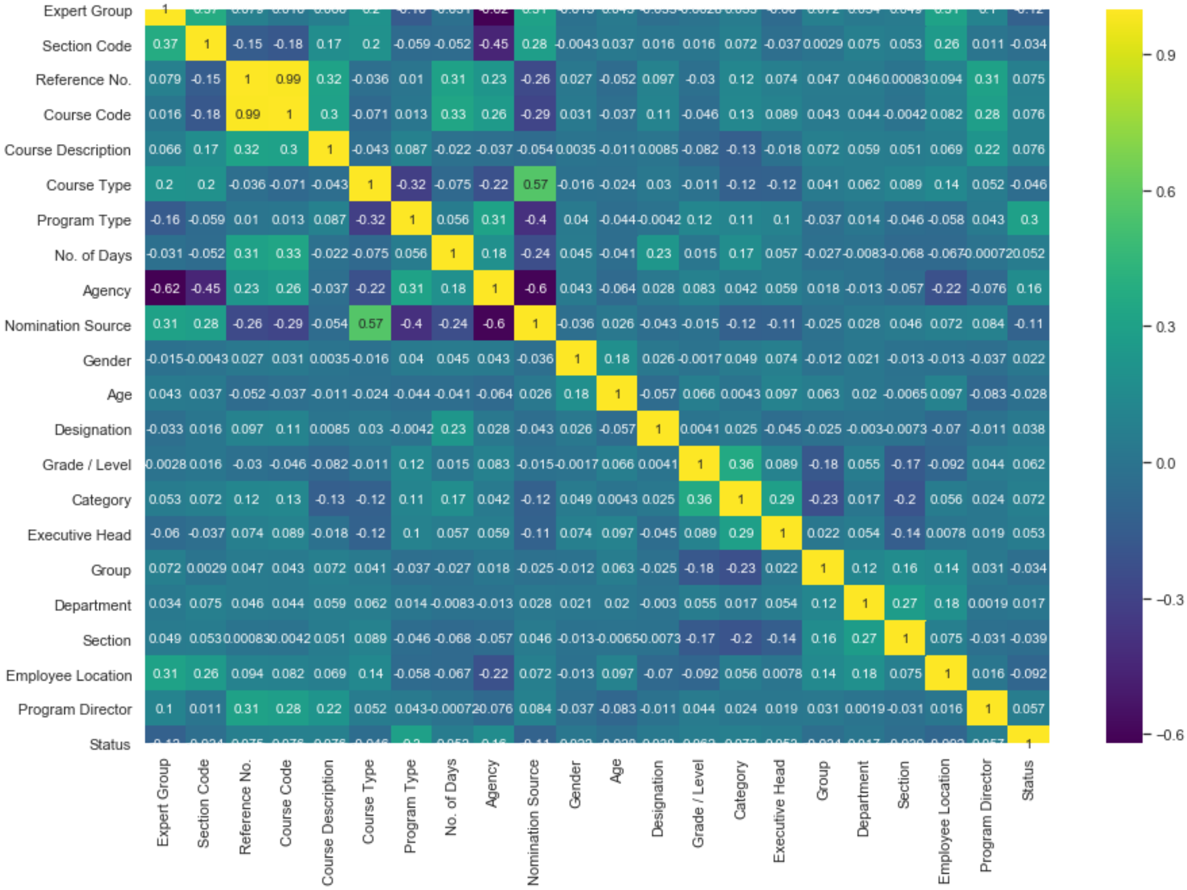


```
In [38]: # Create PDF
         graph.write_pdf("TATA_Data.pdf")

         # Create PNG
         graph.write_png("TATA_Data.png")
```

```
Out[38]: True
```

```
In [39]: plt.figure(figsize=(15,10))
         sns.heatmap(data= df.corr(), annot=True, cmap='viridis')
```

```
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x1b7edb04588>
```

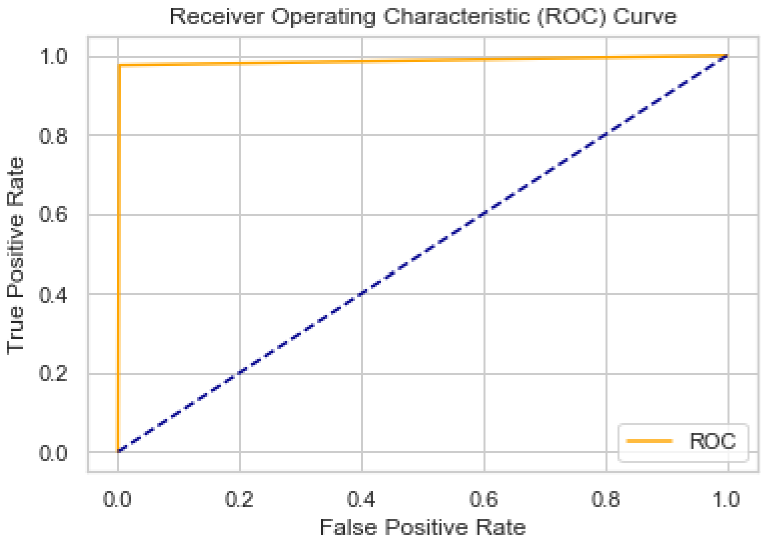


```
In [40]: auc = roc_auc_score(y_test, predictions_2)
         print('AUC: %.2f' % auc)
```

```
AUC: 0.99
```

In [41]:

```
fpr, tpr, thresholds = roc_curve(y_test, predictions_2)
plot_roc_curve(fpr, tpr)
```



In [ ]:

```
```