In [1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_a
uc_score
import seaborn as sns
```

In [2]:
```
dta = sm.datasets.fair.load_pandas().data
#add "affair" column: 1 represents having affairs, 0  represents not
dta['affair'] = (dta.affairs > 0).astype(int)
y, X = dmatrices('affair ~ C(rate_marriage) + age + yrs_married + children + C
(religious) + C(educ) + C(occupation) + C(occupation_husb)', dta, return_type=
"dataframe")
X = X.rename(columns =  { 'C(occupation)[T.2.0]':'occ_2',
                          'C(occupation)[T.3.0]':'occ_3',
                          'C(occupation)[T.4.0]':'occ_4',
                          'C(occupation)[T.5.0]':'occ_5',
                          'C(occupation)[T.6.0]':'occ_6',
                          'C(occupation_husb)[T.2.0]':'occ_husb_2',
                          'C(occupation_husb)[T.3.0]':'occ_husb_3',
                          'C(occupation_husb)[T.4.0]':'occ_husb_4',
                          'C(occupation_husb)[T.5.0]':'occ_husb_5',
                          'C(occupation_husb)[T.6.0]':'occ_husb_6',
                         'C(rate_marriage)[T.2.0]': 'rate_marriage_2',
                         'C(rate_marriage)[T.3.0]': 'rate_marriage_3',
                         'C(rate_marriage)[T.4.0]': 'rate_marriage_4',
                         'C(rate_marriage)[T.5.0]': 'rate_marriage_5',
                         'C(educ)[T.12.0]': 'edu_12',
                         'C(educ)[T.14.0]': 'edu_14',
                         'C(educ)[T.16.0]': 'edu_16',
                         'C(educ)[T.17.0]': 'edu_17',
                         'C(educ)[T.20.0]': 'edu_20',
                         'C(religious)[T.2.0]': 'religious_2',
                         'C(religious)[T.3.0]': 'religious_3',
                         'C(religious)[T.4.0]': 'religious_4'})
y = np.ravel(y)
```

In [3]:
```
dta.head()
```

Out[3]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	a
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0	0.1
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0	3.2
2	4.0	22.0	2.5	0.0	1.0	16.0	3.0	5.0	1.4
3	4.0	37.0	16.5	4.0	3.0	16.0	5.0	5.0	0.7
4	5.0	27.0	9.0	1.0	1.0	14.0	3.0	4.0	4.6

In [4]:
```
X.head()
```

Out[4]:

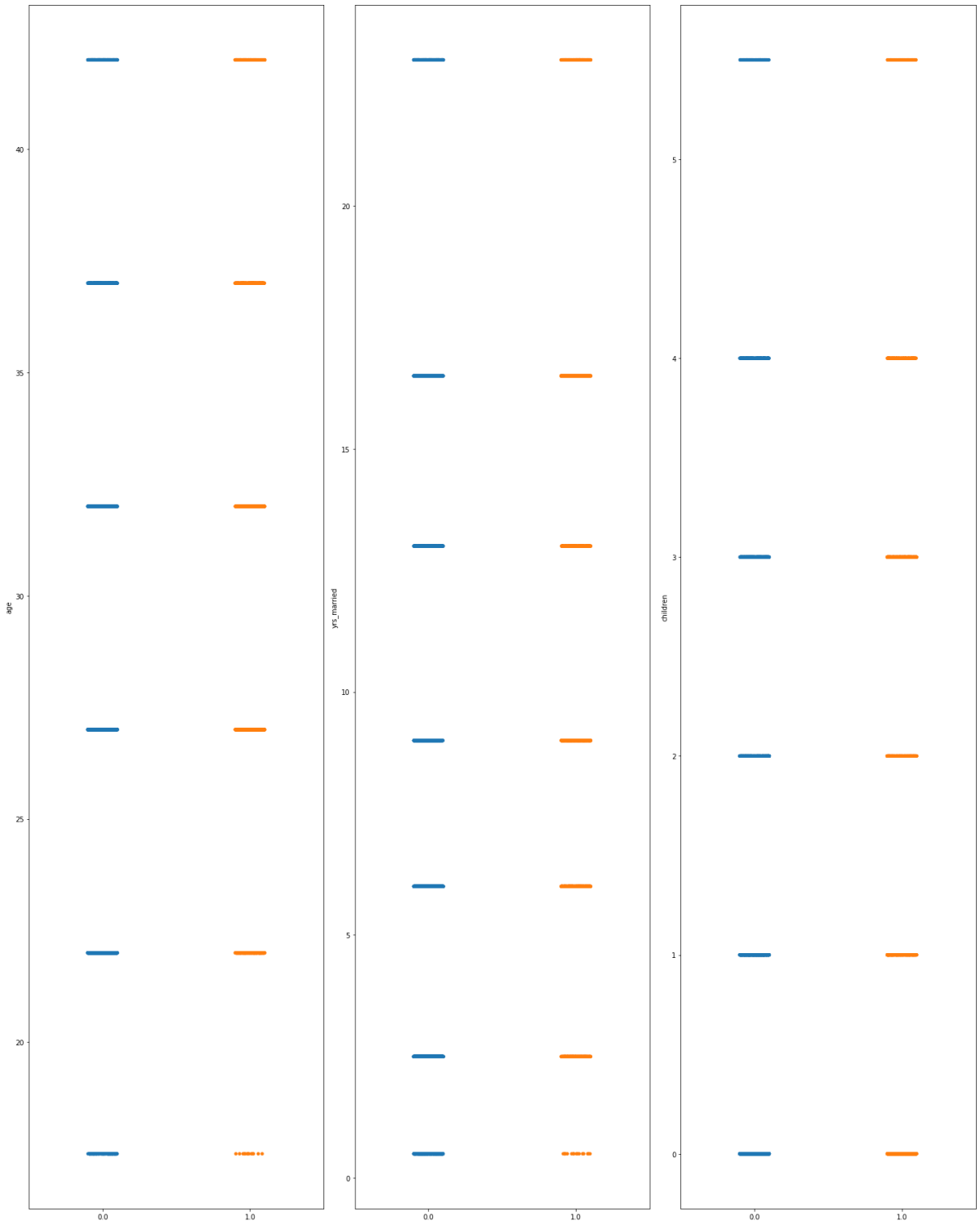
	Intercept	rate_marriage_2	rate_marriage_3	rate_marriage_4	rate_marriage_5	religious_2	reli
0	1.0	0.0	1.0	0.0	0.0	0.0	
1	1.0	0.0	1.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	1.0	0.0	0.0	
3	1.0	0.0	0.0	1.0	0.0	0.0	
4	1.0	0.0	0.0	0.0	1.0	0.0	

5 rows × 26 columns

In [5]:
```
# let's see how data is distributed for every column
plt.figure(figsize=(20,25), facecolor='white')
plotnumber = 1

for column in ['age', 'yrs_married', 'children']:
    if plotnumber<=3 :
        ax = plt.subplot(1, 3, plotnumber)
        sns.stripplot(y, X[column])
    plotnumber+=1
plt.tight_layout()
```

```
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorato
rs.py:43: FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`, and pas
sing other arguments without an explicit keyword will result in an error or m
isinterpretation.
  FutureWarning
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorato
rs.py:43: FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`, and pas
sing other arguments without an explicit keyword will result in an error or m
isinterpretation.
  FutureWarning
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorato
rs.py:43: FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`, and pas
sing other arguments without an explicit keyword will result in an error or m
isinterpretation.
  FutureWarning
```


age
40
35
30
25
20
0.0
1.0
yrs_married
20
15
10
5
0
0.0
1.0
children
5
4
3
2
1
0
0.0
1.0

In [6]:
```
X.describe()
```

Out[6]:

	Intercept	rate_marriage_2	rate_marriage_3	rate_marriage_4	rate_marriage_5	religious_2
count	6366.0	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000
mean	1.0	0.054665	0.155985	0.352183	0.421615	0.356111
std	0.0	0.227344	0.362870	0.477688	0.493856	0.478886
min	1.0	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.0	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.0	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.0	0.000000	0.000000	1.000000	1.000000	1.000000
max	1.0	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 26 columns

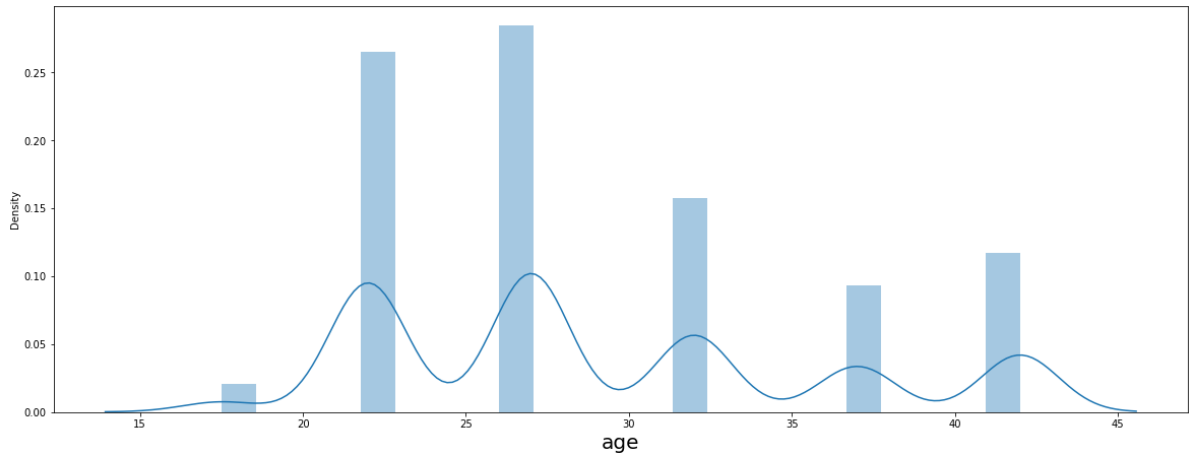
In [7]:
```
X.isna().sum()
```

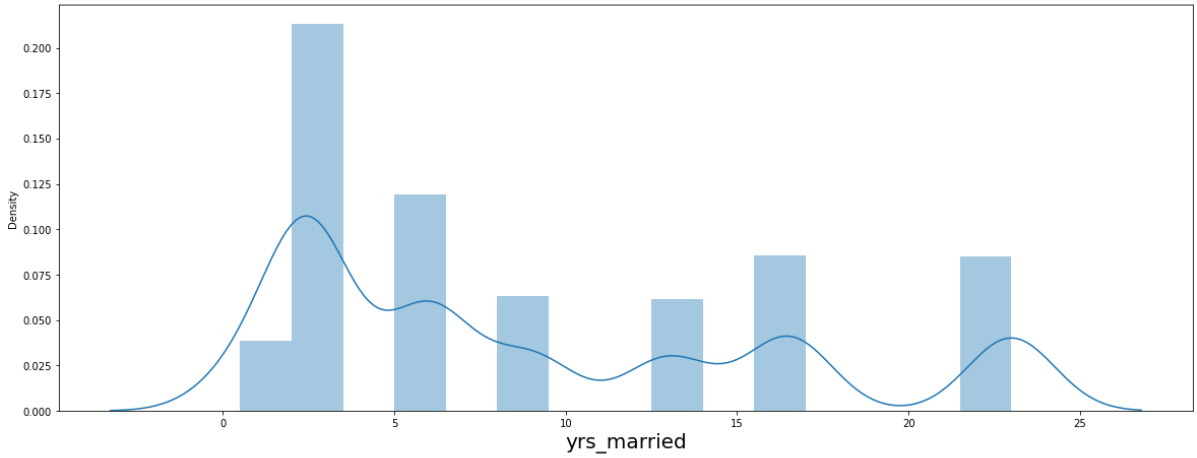
Out[7]:
```
Intercept          0
rate_marriage_2    0
rate_marriage_3    0
rate_marriage_4    0
rate_marriage_5    0
religious_2        0
religious_3        0
religious_4        0
edu_12             0
edu_14             0
edu_16             0
edu_17             0
edu_20             0
occ_2              0
occ_3              0
occ_4              0
occ_5              0
occ_6              0
occ_husb_2         0
occ_husb_3         0
occ_husb_4         0
occ_husb_5         0
occ_husb_6         0
age                0
yrs_married        0
children           0
dtype: int64
```

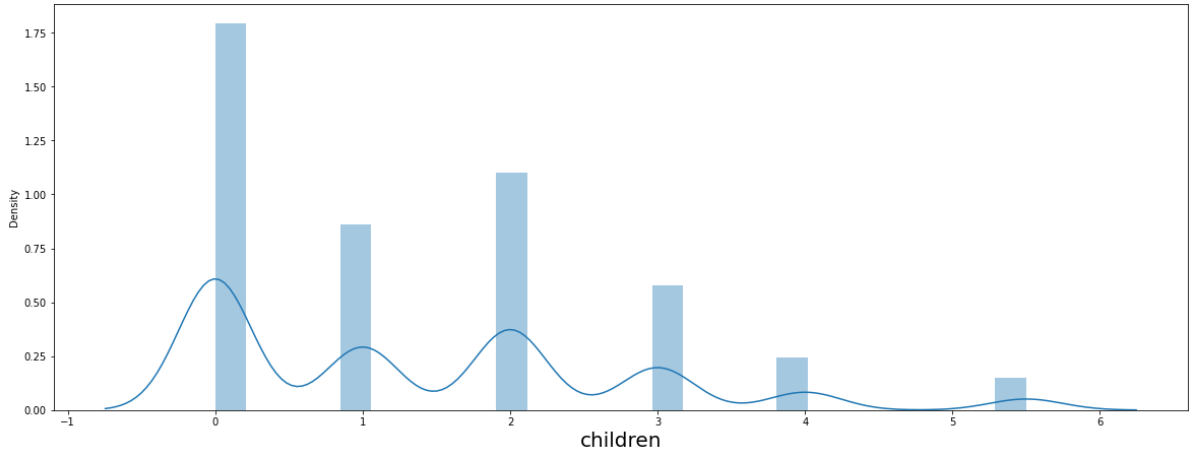
In [8]:
```
# let's see how data is distributed for every column
plt.figure(figsize=(20,25), facecolor='white')
plotnumber = 1

for column in ['age', 'yrs_married', 'children']:
    if plotnumber<=3 :     # as there are 9 columns in the data
        ax = plt.subplot(3, 1,plotnumber)
        sns.distplot(X[column])
        plt.xlabel(column,fontsize=20)
        #plt.ylabel('Salary',fontsize=20)
    plotnumber+=1
plt.show()
```

```
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\distribut
ions.py:2551: FutureWarning: `distplot` is a deprecated function and will be
removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-lev
el function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\distribut
ions.py:2551: FutureWarning: `distplot` is a deprecated function and will be
removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-lev
el function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\distribut
ions.py:2551: FutureWarning: `distplot` is a deprecated function and will be
removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-lev
el function for histograms).
  warnings.warn(msg, FutureWarning)
```







In [9]:
```
X.drop(columns=['Intercept'], inplace=True)
```

In [10]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, ra
ndom_state = 123)
```

In [11]:
```
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(x_train, y_train)
```

Out[11]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [12]:
```
y_pred = log_reg.predict(x_test)
```

In [13]:
```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[13]:
```
0.7154522613065326
```

In [14]:
```
# Confusion Matrix
conf_mat = confusion_matrix(y_test,y_pred)
conf_mat
```

Out[14]:
```
array([[950, 116],
       [337, 189]], dtype=int64)
```

In [15]:
```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

In [16]:
```
# Breaking down the formula for Accuracy
Accuracy = (true_positive + true_negative) / (true_positive +false_positive +
false_negative + true_negative)
Accuracy
```

Out[16]:
```
0.7154522613065326
```

In [17]:
```
# Precison
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[17]:
```
0.8911819887429644
```

In [18]:
```
# Recall
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[18]:
```
0.7381507381507382
```

In [19]:
```
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

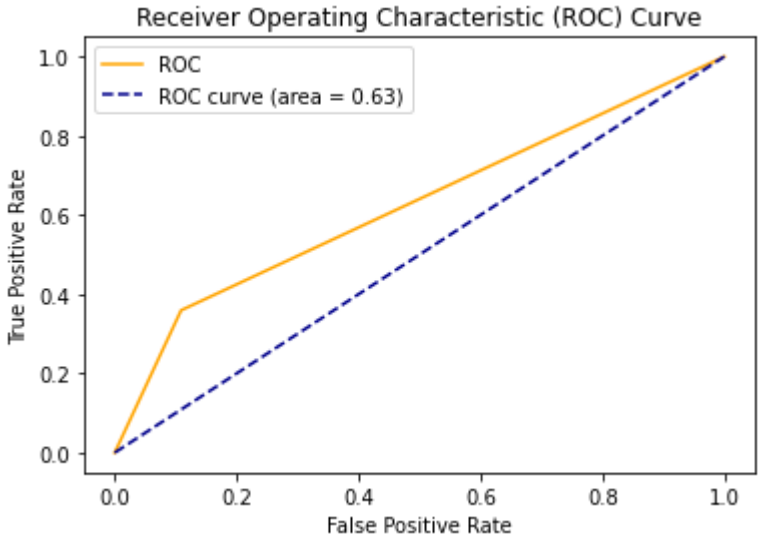
Out[19]:
```
0.8074798130046749
```

In [20]:
```
# Area Under Curve
auc = roc_auc_score(y_test, y_pred)
auc
```

Out[20]: 0.6252487890482883

In [21]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

In [22]:
```
plt.plot(fpr, tpr, color='orange', label='ROC')
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--', label='ROC curve (area = %0.2f)' % auc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```



Project Done By: Urvi Gadda

mailto: urvigada96@gmail.com

In [ ]: