In [ ]:

```
#*******************************************************************************************
#
#                            Session21 Machine Learning 3 Assignment
#
#*******************************************************************************************
# Problem Statement : Predict Classification for Each Women whether having affair

# Step 0 : importing the librabries
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline

from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split ,cross_val_score
from sklearn import metrics
#from sklearn.cross_validation import cross_val_score


# Step1 : Loading the Data

# Description of Variables
# The dataset contains 6366 observations of 9 variables:
# rate_marriage: woman's rating of her marriage (1 = very poor, 5 = very good)
# age: woman's age
# yrs_married: number of years married
# children: number of children
# religious: woman's rating of how religious she is (1 = not religious, 4 = strongly religious)
# educ: level of education (9 = grade school, 12 = high school, 14 = some college, 16 = college graduate, 17 = some g
# occupation: woman's occupation (1 = student, 2 = farming/semi-skilled/unskilled, 3 = "white collar", 4 = teacher/nu
# occupation_husb: husband's occupation (same coding as above)
# affairs: time spent in extra-marital affairs

# Loading the predefined DataSet DAta
dta = sm.datasets.fair.load_pandas().data


print(dta.head())

print("\nunique values in Occupation fields \n",'-'*80 )
print(dta.occupation.unique())
print(dta.occupation_husb.unique())

dta['affair'] = (dta.affairs > 0).astype(int)

y, X = dmatrices('affair ~ rate_marriage + age + yrs_married + children + religious + educ + C(occupation) + C(occupa
                  dta, return_type="dataframe")


print("The number of rows and columns in the Data \n")
print(y.shape)
print(X.shape)
print(X.columns)
#print()

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'})
y = np.ravel(y)

print("The Data with renamed Columns \n",'-'*80)
print(X.head(2))


```

In [ ]:

```
#Step2 : Analyse the Data

# knowing the size of the data
print(" Analyse the DAta with functions like describe, info , check for nulls \n",'-'*80)
print(X.shape)
print(y.shape)

# knowing the
print(X.describe())
print('-'*80)

print(X.info())

# checking if there are any null fields
print(X.isnull().sum())

print('-'*80,"\n No null values for the Data \n")
```

In [ ]:

```
# Step3 : Split the Data into Training and Test Set

print(X.columns)
features = ['Intercept','occ_2', 'occ_3', 'occ_4', 'occ_5', 'occ_6', 'occ_husb_2','occ_husb_3', 'occ_husb_4', 'occ_hu
X = X[features]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)

```

In [ ]:

```
# Step 4 : Doing the Histogram of all variables to visualise the Data better
X_train.hist()

```

In [ ]:

```
# Step 5: Build the Logistic Regrresssion Model and train it

clf_1 = LogisticRegression()
clf_1.fit(X_train, y_train)

# predict the Values
print(clf_1.predict(X_test))


```

In [ ]:

```
# Step 6 : Model Evaluation
from sklearn.metrics import roc_auc_score, accuracy_score
from sklearn import metrics


accuracy = accuracy_score(y_test, clf_1.predict(X_test))

rocauc= roc_auc_score(y_test, clf_1.predict(X_test))

confusion_matrix = metrics.confusion_matrix(y_test, clf_1.predict(X_test))

classification_report = metrics.classification_report(y_test, clf_1.predict(X_test))

print(accuracy)
print("-"*50)
print(rocauc)
print("-"*50)
print(confusion_matrix)
print("-"*50)
print(classification_report)



```

In [ ]:

```
## Model Evaluation by  cross_val_score

import time
start_time = time.time()
from sklearn.model_selection import cross_val_score
clf_2 = LogisticRegression()

scores= cross_val_score(clf_2, X, y, cv=10)
precision= cross_val_score(clf_2,  X, y, cv=10, scoring='precision')
recall= cross_val_score(clf_2,  X, y, cv=10, scoring='recall')

print(scores.mean())
print(precision.mean())
print(recall.mean())

print("--- %s seconds ---" % (time.time() - start_time))
```

In [ ]:

```
# ROC AUC score

from sklearn.metrics import roc_curve, auc
from sklearn import metrics

#X_train
probs = clf_1.predict_proba(X_train)
preds = probs[:,1]


fpr, tpr, threshold = metrics.roc_curve(y_train, preds)
roc_auc = metrics.auc(fpr, tpr)


#X_test
probs1 = clf_1.predict_proba(X_test)
preds1 = probs1[:,1]

fpr1, tpr1, threshold1 = metrics.roc_curve(y_test, preds1)
roc_auc1 = metrics.auc(fpr1, tpr1)
```

In [ ]:

```
import matplotlib.pyplot as plt

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.plot(fpr1, tpr1, 'g', label = 'AUC = %0.2f' % roc_auc1)

plt.legend(['training','test'],loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

plt.show()
```