Importing Required Libraries

In [114]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_confusion_matrix, plot_roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

Importing Dataset

In [2]:

```
data = pd.read_csv(r'./UCI_Credit_Card.csv')
```

In [3]:

```
data
```

Out[3]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4
0	1	20000.0	2	2	1	24	2	2	-1	-1
1	2	120000.0	2	2	2	26	-1	2	0	0
2	3	90000.0	2	2	2	34	0	0	0	0
3	4	50000.0	2	2	1	37	0	0	0	0
4	5	50000.0	1	2	1	57	-1	0	-1	0
...	...	...	...	...	...	...	...	...	...	...
29995	29996	220000.0	1	3	1	39	0	0	0	0
29996	29997	150000.0	1	3	2	43	-1	-1	-1	-1
29997	29998	30000.0	1	2	2	37	4	3	2	-1
29998	29999	80000.0	1	3	1	41	1	-1	0	0
29999	30000	50000.0	1	2	1	46	0	0	0	0

30000 rows × 25 columns

Statistical Description

In [4]:

```
data.describe()
```

Out[4]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000

8 rows × 25 columns

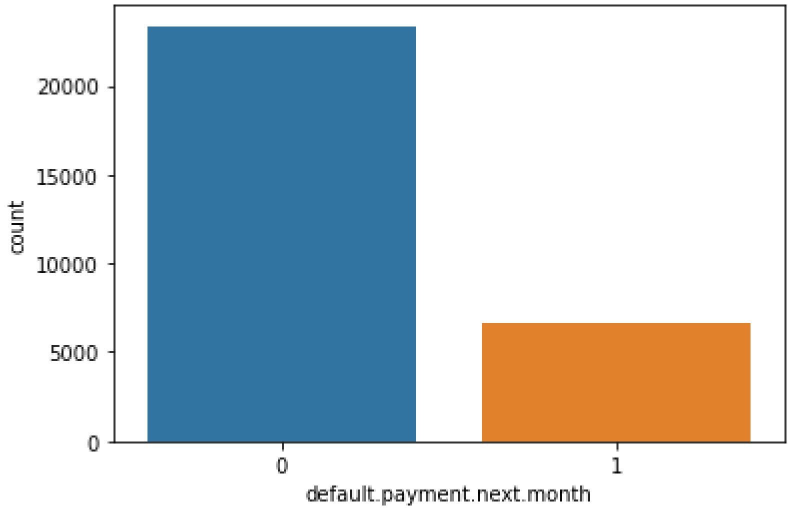
In [9]:

```
sns.countplot(data['default.payment.next.month'])
```

```
/home/ayush/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[9]:

```
<AxesSubplot:xlabel='default.payment.next.month', ylabel='count'>
```



Data types and Null value analysis

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   ID                          30000 non-null  int64  
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64  
 3   EDUCATION                   30000 non-null  int64  
 4   MARRIAGE                    30000 non-null  int64  
 5   AGE                         30000 non-null  int64  
 6   PAY_0                       30000 non-null  int64  
 7   PAY_2                       30000 non-null  int64  
 8   PAY_3                       30000 non-null  int64  
 9   PAY_4                       30000 non-null  int64  
 10  PAY_5                       30000 non-null  int64  
 11  PAY_6                       30000 non-null  int64  
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64  
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

Distribution plots based on individual features

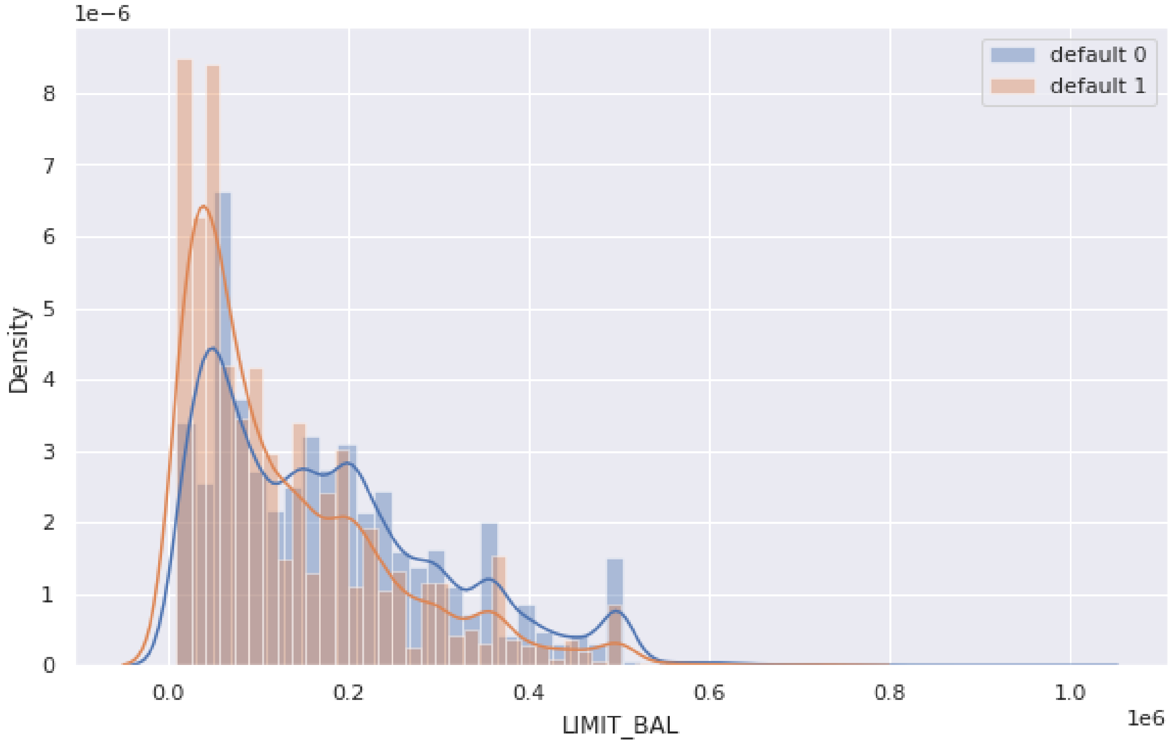
In [10]:

```
plt.figure(figsize=(10, 6))
sns.set(style="darkgrid")
sns.distplot(data[data['default.payment.next.month']==0]['LIMIT_BAL'], label='default 0')
sns.distplot(data[data['default.payment.next.month']==1]['LIMIT_BAL'], label='default 1')
plt.legend()
```

```
/home/ayush/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2
551: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `histplot` (an axes-level func
tion for histograms).
  warnings.warn(msg, FutureWarning)
/home/ayush/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2
551: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `histplot` (an axes-level func
tion for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<matplotlib.legend.Legend at 0x7f50d9cb8490>
```

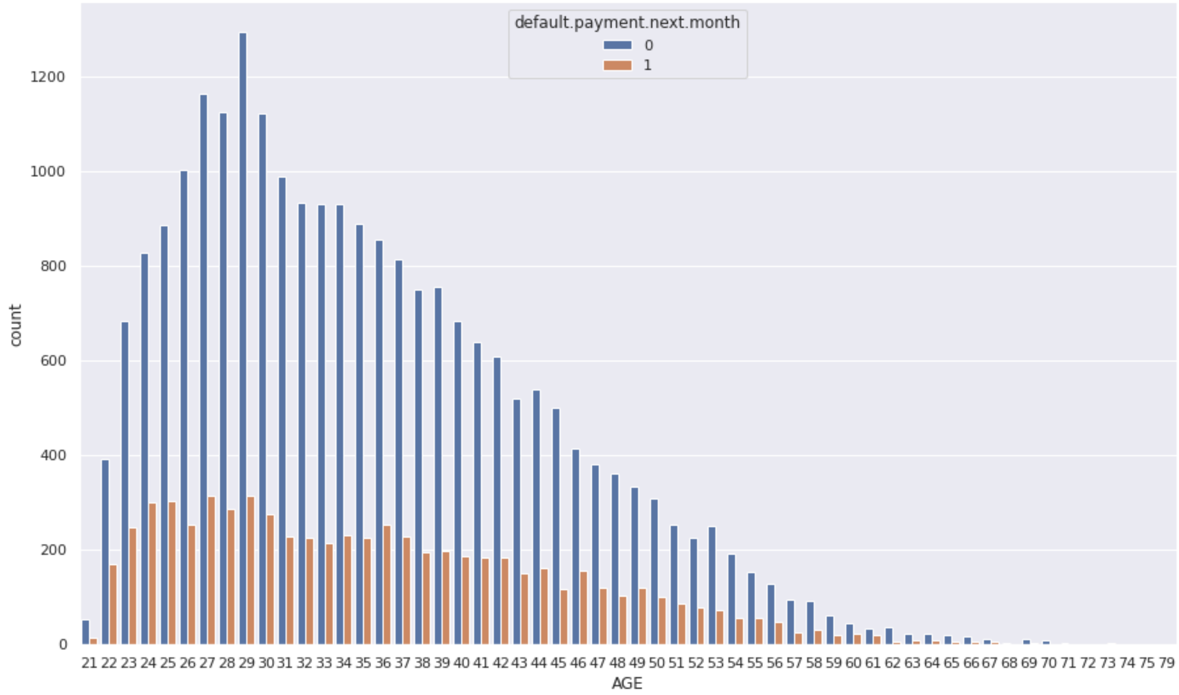


In [11]:

```
plt.figure(figsize=(15, 9))
sns.countplot(data=data, x='AGE', hue='default.payment.next.month')
```

Out[11]:

```
<AxesSubplot:xlabel='AGE', ylabel='count'>
```

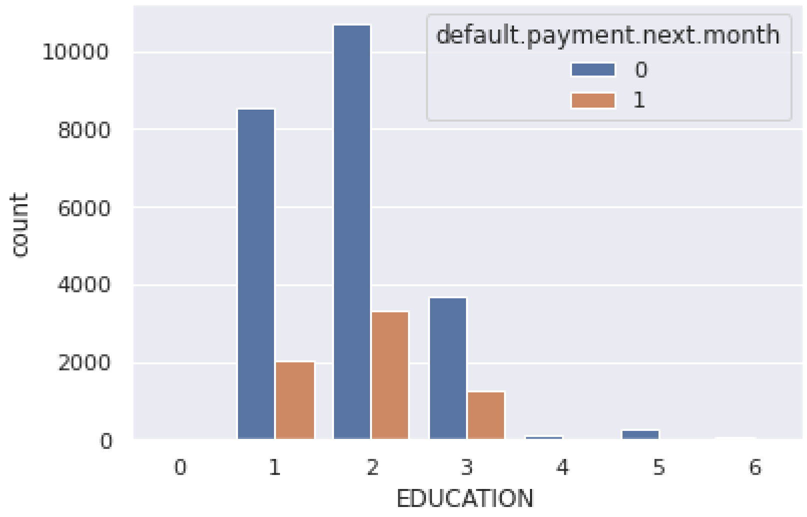


In [12]:

```
sns.countplot(data=data, x='EDUCATION', hue='default.payment.next.month')
```

Out[12]:

```
<AxesSubplot:xlabel='EDUCATION', ylabel='count'>
```

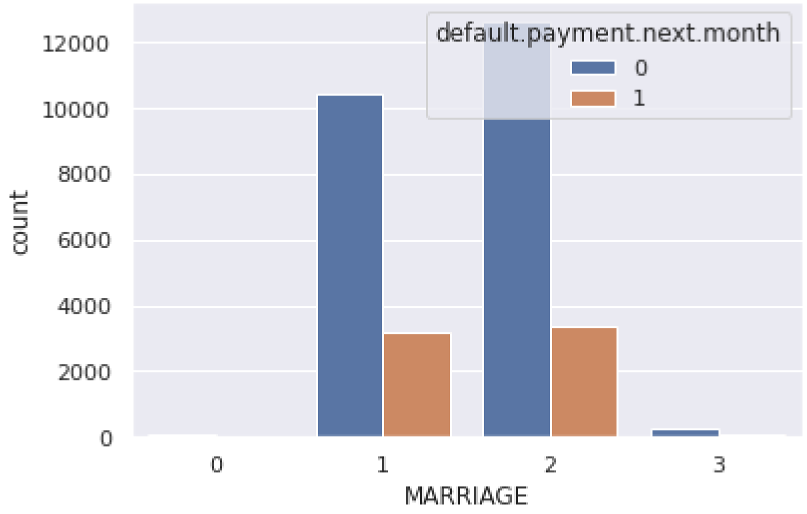


In [13]:

```
sns.countplot(data=data, x='MARRIAGE', hue='default.payment.next.month')
```

Out[13]:

```
<AxesSubplot:xlabel='MARRIAGE', ylabel='count'>
```



Splitting data into features and labels

In [19]:

```
X = data.drop(columns=['ID', 'default.payment.next.month'])
y = data['default.payment.next.month']
(X.shape, y.shape)
```

Out[19]:

```
((30000, 23), (30000,))
```

In [102]:

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Applying Random Forest Classifier with GridSearch for hyperparameter tuning

In [104]:

```
clf = RandomForestClassifier()
```

In [105]:

```
parameters = {
              'max_depth': [3, 4, 5],
                'n_estimators': [100, 500, 1000]
                    }
```

In [106]:

```
rf_grid = GridSearchCV(clf,
                        parameters,
                        cv = 3,
                        n_jobs = 5,
                        verbose=True)
```

In [107]:

```
rf_grid.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits

[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done  27 out of  27 | elapsed:   42.0s finished
```

Out[107]:

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=5,
             param_grid={'max_depth': [3, 4, 5],
                         'n_estimators': [100, 500, 1000]},
             verbose=True)
```

In [108]:

```
print("Best Score:", rf_grid.best_score_)
print("Best Parameters:", rf_grid.best_params_)
```

```
Best Score: 0.8151904761904762
Best Parameters: {'max_depth': 5, 'n_estimators': 100}
```

Model Training

In [109]:

```
clf = RandomForestClassifier(n_estimators=1000, max_depth=5)
clf.fit(x_train, y_train)
```

Out[109]:

```
RandomForestClassifier(max_depth=5, n_estimators=1000)
```

Model Evaluation

In [110]:

```
y_pred = clf.predict(x_test)
```

In [111]:

```
accuracy_score(y_test, y_pred)
```

Out[111]:

```
0.815
```

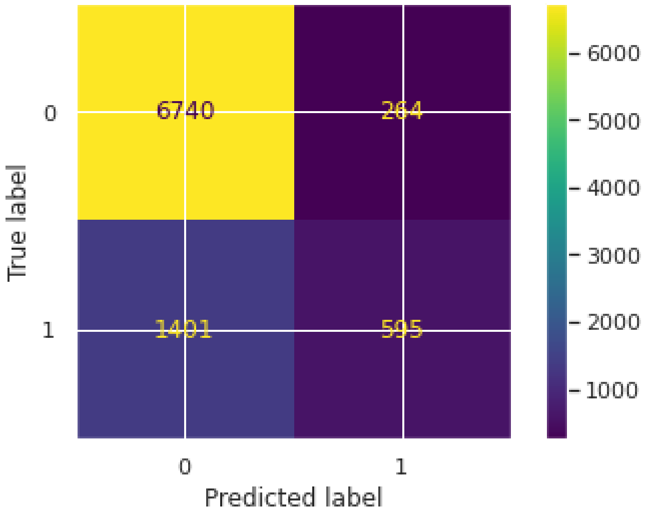
Confusion Matrix

In [112]:

```
plot_confusion_matrix(clf, x_test, y_test)
```

Out[112]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f50c982
eca0>
```



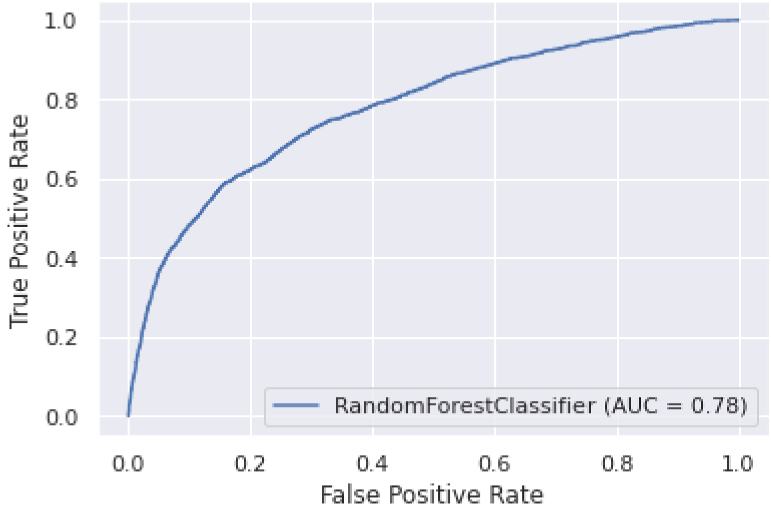
ROC Curve

In [113]:

```
plot_roc_curve(clf, x_test, y_test)
```

Out[113]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f50c807d1f0>
```


True Positive Rate
RandomForestClassifier (AUC = 0.78)
False Positive Rate