Risk Analysis of credit to a customer

In [65]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

loading the train dataset

In [2]:
```
df_train = pd.read_csv(r"C:\Users\Ram Chander\Downloads\creditanalysis_train.csv")
df_train.head()
```

Out[2]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X19	X20	
0	300000	1	3	2	31	0	0	0	0	0	...	86263	87238	89176	4000	4000	4100	3
1	20000	1	2	2	24	0	0	0	0	0	...	14694	16914	14074	1313	2110	4000	6
2	90000	2	2	2	35	-1	-1	-1	-2	-2	...	0	0	0	2667	0	0	
3	300000	2	2	1	40	1	-2	-2	-2	-2	...	0	0	0	0	0	0	
4	70000	2	2	2	36	0	0	0	0	0	...	29314	28844	29443	3340	2044	1773	1

5 rows × 24 columns

loading the test dataset

In [3]:
```
df_test = pd.read_csv(r"C:\Users\Ram Chander\Downloads\creditanalysis_test.csv")
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
df_test.head()
```

Out[3]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X14	X15	X16	X17	X18	X19
0	30000	1	2	2	25	0	0	0	0	0	...	11581	12580	13716	14828	1500	2000
1	150000	2	1	2	26	0	0	0	0	0	...	116684	101581	77741	77264	4486	4235
2	70000	2	3	1	32	0	0	0	0	0	...	68530	69753	70111	70212	2431	3112
3	130000	1	3	2	49	0	0	0	0	0	...	16172	16898	11236	6944	1610	1808
4	50000	2	2	2	36	0	0	0	0	0	...	42361	19574	20295	19439	2000	1500

5 rows × 23 columns

In [4]:
```
df_train.count()
```

Out[4]:
```
X1     22500
X2     22500
X3     22500
X4     22500
X5     22500
X6     22500
X7     22500
X8     22500
X9     22500
X10    22500
X11    22500
X12    22500
X13    22500
X14    22500
X15    22500
X16    22500
X17    22500
X18    22500
X19    22500
X20    22500
X21    22500
X22    22500
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
X23    22500
y      22500
dtype: int64
```

In [5]:
```
df_test.count()
```

Out[5]:
```
X1     7500
X2     7500
X3     7500
X4     7500
X5     7500
X6     7500
X7     7500
X8     7500
X9     7500
X10    7500
X11    7500
X12    7500
X13    7500
X14    7500
X15    7500
X16    7500
X17    7500
X18    7500
X19    7500
X20    7500
X21    7500
X22    7500
X23    7500
dtype: int64
```

In [6]:
```
df_train.keys()
df_test.keys()
```

Out[6]:
```
Index(['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10', 'X1
1',
       'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20',
'X21',
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
       'X22', 'X23'],
      dtype='object')
```

In [7]:
```
df_train.info()
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22500 entries, 0 to 22499
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      22500 non-null  int64
 1   X2      22500 non-null  int64
 2   X3      22500 non-null  int64
 3   X4      22500 non-null  int64
 4   X5      22500 non-null  int64
 5   X6      22500 non-null  int64
 6   X7      22500 non-null  int64
 7   X8      22500 non-null  int64
 8   X9      22500 non-null  int64
 9   X10     22500 non-null  int64
 10  X11     22500 non-null  int64
 11  X12     22500 non-null  int64
 12  X13     22500 non-null  int64
 13  X14     22500 non-null  int64
 14  X15     22500 non-null  int64
 15  X16     22500 non-null  int64
 16  X17     22500 non-null  int64
 17  X18     22500 non-null  int64
 18  X19     22500 non-null  int64
 19  X20     22500 non-null  int64
 20  X21     22500 non-null  int64
 21  X22     22500 non-null  int64
 22  X23     22500 non-null  int64
 23  y       22500 non-null  int64
dtypes: int64(24)
memory usage: 4.1 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

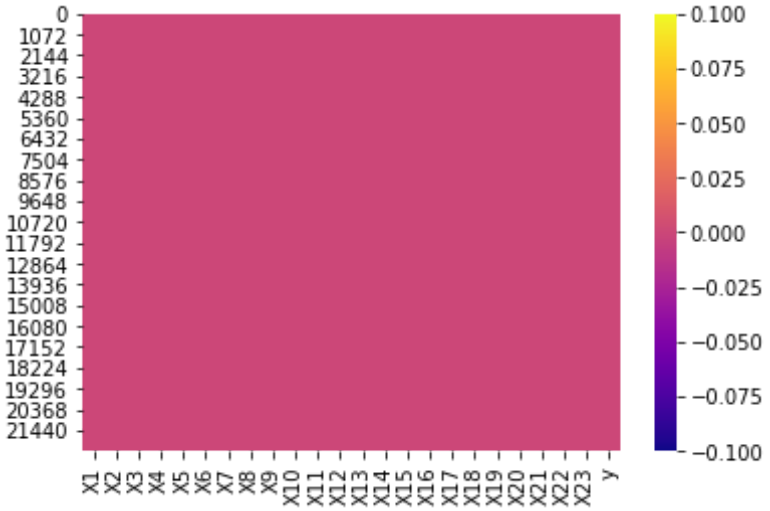
```
Data columns (total 23 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      7500 non-null   int64
 1   X2      7500 non-null   int64
 2   X3      7500 non-null   int64
 3   X4      7500 non-null   int64
 4   X5      7500 non-null   int64
 5   X6      7500 non-null   int64
 6   X7      7500 non-null   int64
 7   X8      7500 non-null   int64
 8   X9      7500 non-null   int64
 9   X10     7500 non-null   int64
 10  X11     7500 non-null   int64
 11  X12     7500 non-null   int64
 12  X13     7500 non-null   int64
 13  X14     7500 non-null   int64
 14  X15     7500 non-null   int64
 15  X16     7500 non-null   int64
 16  X17     7500 non-null   int64
 17  X18     7500 non-null   int64
 18  X19     7500 non-null   int64
 19  X20     7500 non-null   int64
 20  X21     7500 non-null   int64
 21  X22     7500 non-null   int64
 22  X23     7500 non-null   int64
dtypes: int64(23)
memory usage: 1.3 MB
```

checking the relation between independent and dependent variables by using seaborn library

In [9]:
```
sns.heatmap(df_train.isnull(),cmap='plasma')
```

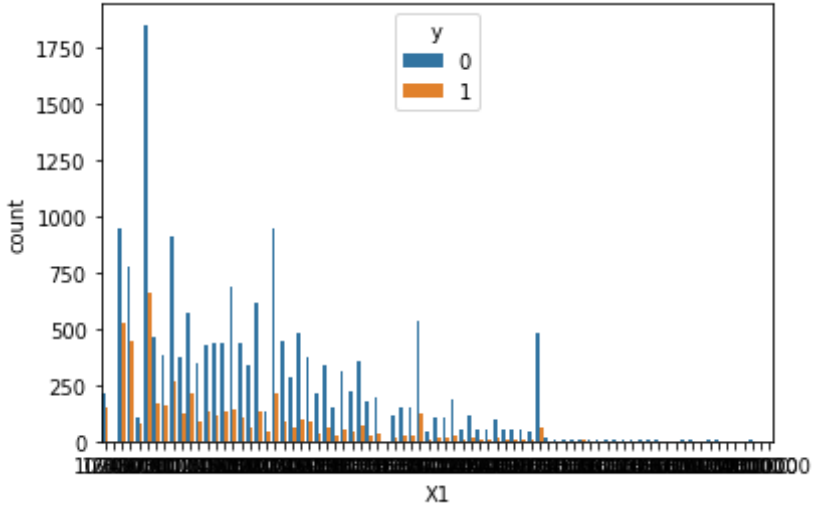
Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26911e93088>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [10]:
```
sns.countplot(x='X1',hue='y',data=df_train)
```

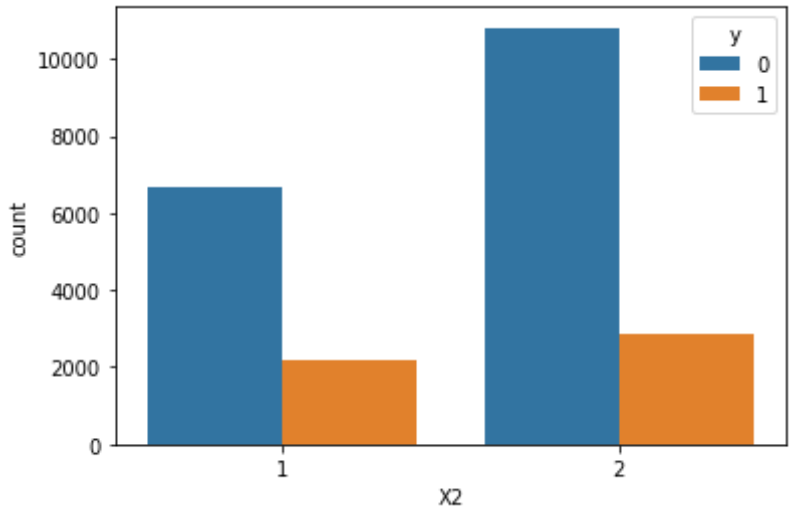
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2691271f8c8>



In [11]:
```
sns.countplot(x='X2',hue='y',data=df_train)
```

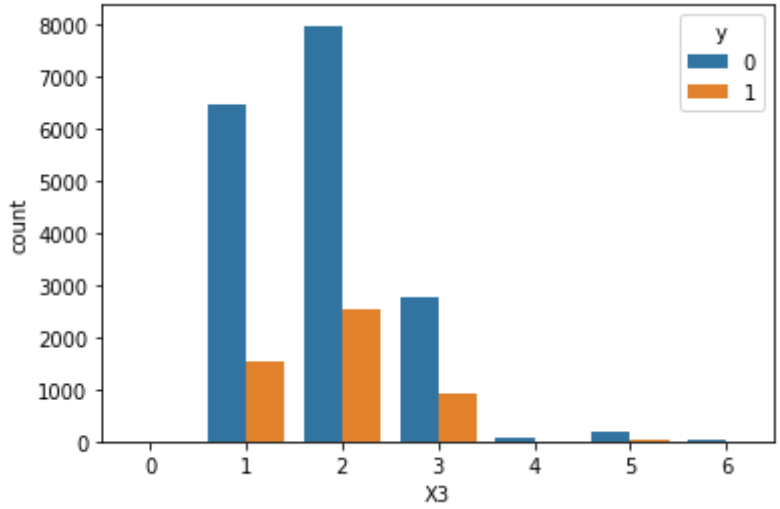
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x269129be8c8>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [12]:
```
sns.countplot(x='X3',hue='y',data=df_train)
```

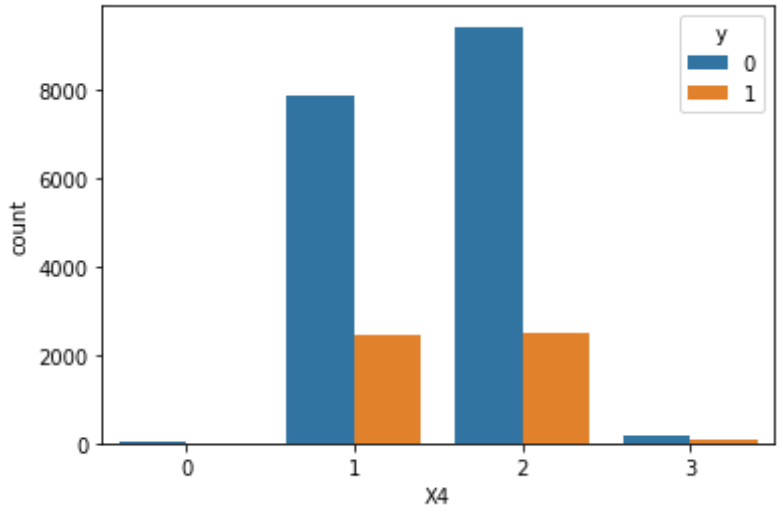
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x26912a53948>



In [13]:
```
sns.countplot(x='X4',hue='y',data=df_train)
```

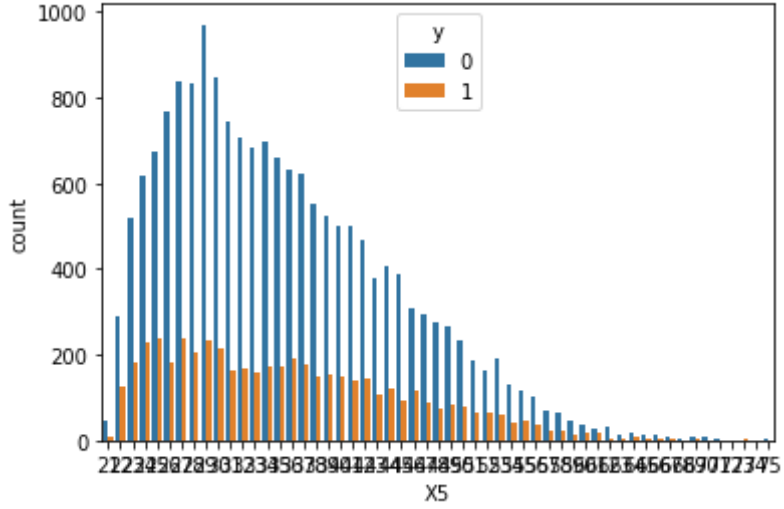
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x26912aeb108>



In [14]:
```
sns.countplot(x='X5',hue='y',data=df_train)
```

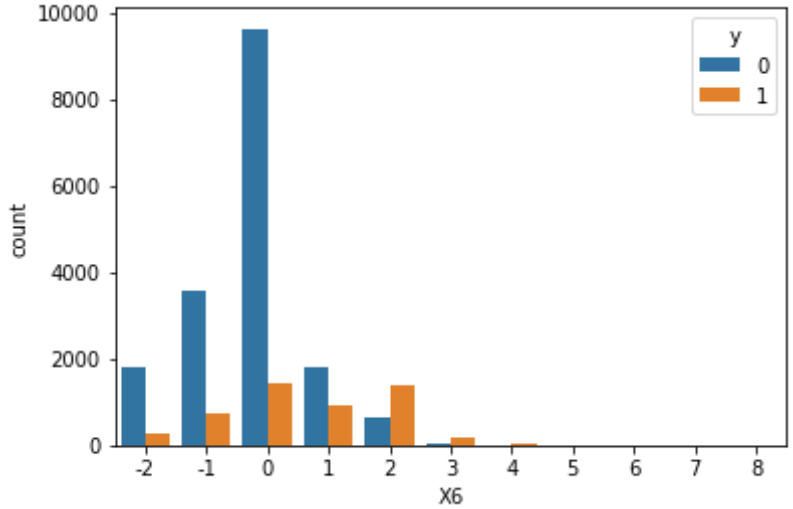
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x26912b58e08>



In [15]:
```
sns.countplot(x='X6',hue='y',data=df_train)
```

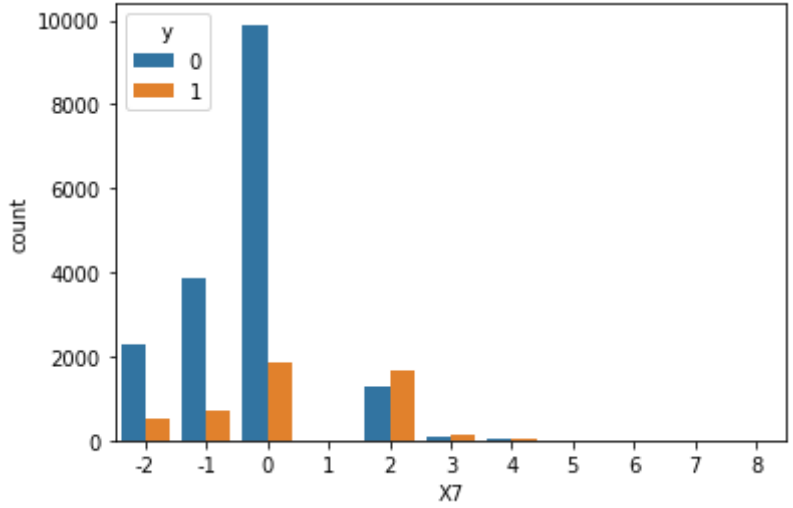
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26912b3e088>



In [16]:
```
sns.countplot(x='X7',hue='y',data=df_train)
```

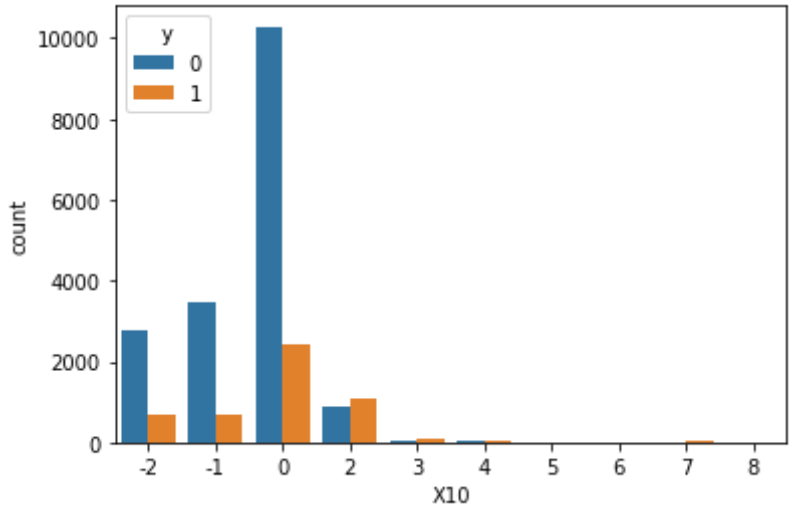
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x26912e33748>



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

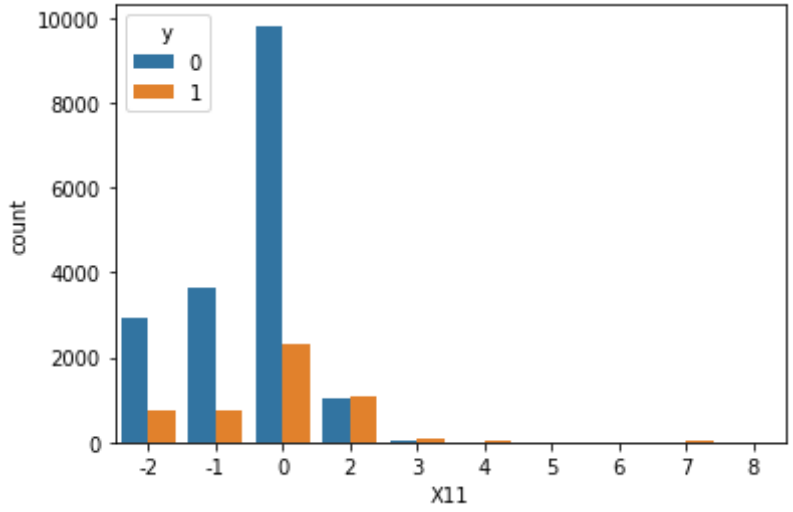
In [17]:
```
sns.countplot(x='X10',hue='y',data=df_train)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x26912e33248>



In [18]:
```
sns.countplot(x='X11',hue='y',data=df_train)
```

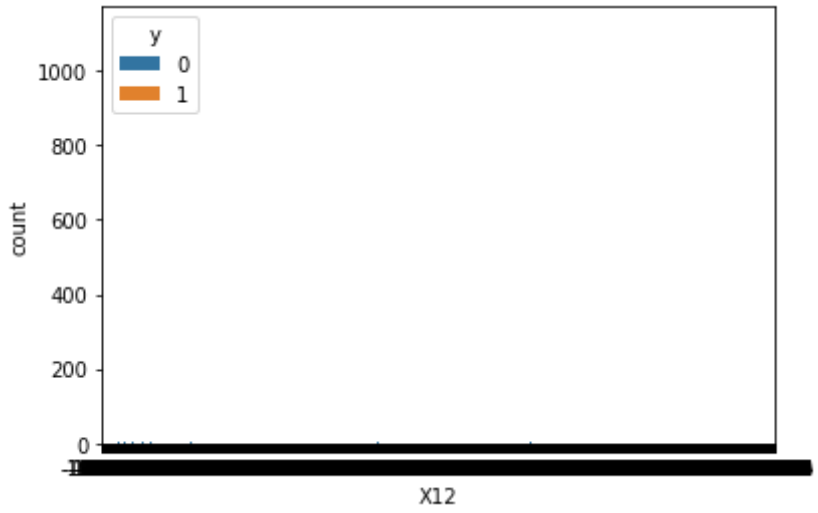
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x26912f74b08>



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [19]:
```
sns.countplot(x='X12',hue='y',data=df_train)
```

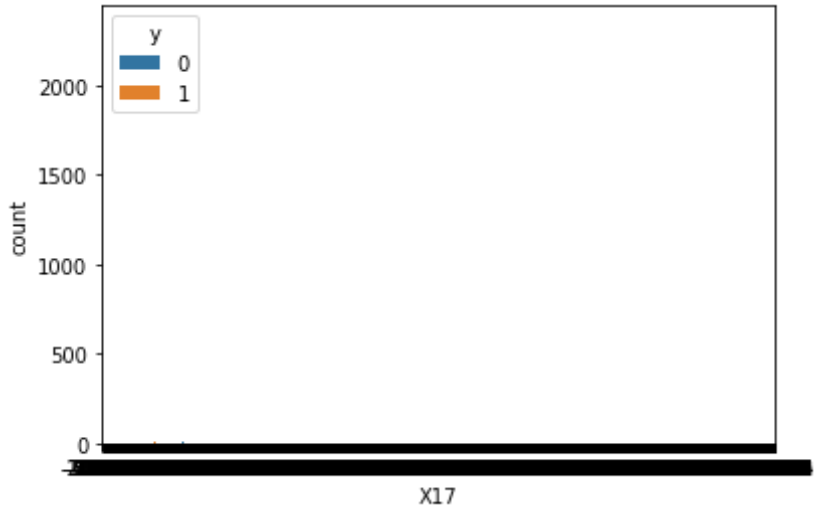
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x26916b4db48>



In [20]:
```
sns.countplot(x='X17',hue='y',data=df_train)
```

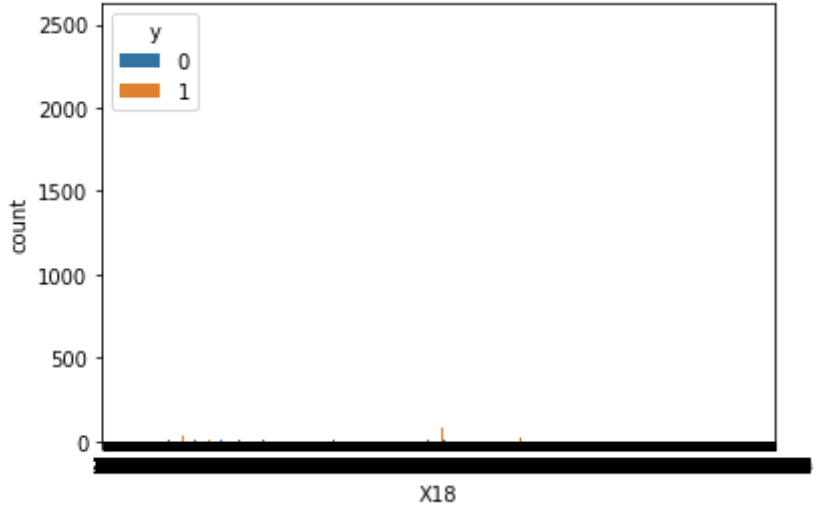
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2694e573f08>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [21]:
```
sns.countplot(x='X18',hue='y',data=df_train)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x2697ba538c8>



since we have observed few variables doesn't effect the target variable , hence we can drop those columns

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [23]:
```
df_train.drop(['X19','X20','X21','X22'],axis=1,inplace=True)
```

In [25]:
```
df_train.drop(['X8','X9','X10','X11'],axis=1,inplace=True)
```

In [27]:
```
df_train.head()
```

Out[27]:

	X1	X2	X3	X4	X5	X6	X7	X12	X13	X14	X15	X16	X17	X18	X23	y
0	300000	1	3	2	31	0	0	80928	82690	84462	86263	87238	89176	4000	3500	0
1	20000	1	2	2	24	0	0	15730	16776	35036	14694	16914	14074	1313	174	0
2	90000	2	2	2	35	-1	-1	2667	2667	0	0	0	0	2667	0	0
3	300000	2	2	1	40	1	-2	0	0	0	0	0	0	0	0	0
4	70000	2	2	2	36	0	0	68028	67864	59165	29314	28844	29443	3340	1297	1

In [29]:
```
df_train.drop(['X13','X14','X15','X16'],axis=1,inplace=True)
```

In [30]:
```
df_train.head()
```

Out[30]:

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23	y
0	300000	1	3	2	31	0	0	80928	89176	4000	3500	0
1	20000	1	2	2	24	0	0	15730	14074	1313	174	0
2	90000	2	2	2	35	-1	-1	2667	0	2667	0	0
3	300000	2	2	1	40	1	-2	0	0	0	0	0
4	70000	2	2	2	36	0	0	68028	29443	3340	1297	1

converting the negative values to positive by using abs() function

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [31]:
```
df_train['X6'] = df_train['X6'].abs()
```

In [32]:
```
df_train['X7'] = df_train['X7'].abs()
```

In [33]:
```
df_train
```

Out[33]:

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23	y
0	300000	1	3	2	31	0	0	80928	89176	4000	3500	0
1	20000	1	2	2	24	0	0	15730	14074	1313	174	0
2	90000	2	2	2	35	1	1	2667	0	2667	0	0
3	300000	2	2	1	40	1	2	0	0	0	0	0
4	70000	2	2	2	36	0	0	68028	29443	3340	1297	1
...	...	...	...	...	...	...	...	...	...	...	...	...
22495	50000	1	2	2	32	0	0	52475	6119	2000	73421	0
22496	200000	1	1	2	37	2	2	157131	172084	13500	4000	1
22497	50000	1	1	2	26	2	2	0	0	0	0	0
22498	70000	2	2	2	25	0	0	73939	28039	3000	1200	1
22499	160000	2	2	1	36	2	2	-20	14129	0	1500	1

22500 rows × 12 columns

In [34]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22500 entries, 0 to 22499
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      22500 non-null  int64
 1   X2      22500 non-null  int64
 2   X3      22500 non-null  int64
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 3   X4      22500 non-null  int64
 4   X5      22500 non-null  int64
 5   X6      22500 non-null  int64
 6   X7      22500 non-null  int64
 7   X12     22500 non-null  int64
 8   X17     22500 non-null  int64
 9   X18     22500 non-null  int64
 10  X23     22500 non-null  int64
 11  y       22500 non-null  int64
dtypes: int64(12)
memory usage: 2.1 MB
```

In [35]:
```
df_train.isnull()
```

Out[35]:

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23	y
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
22495	False	False	False	False	False	False	False	False	False	False	False	False
22496	False	False	False	False	False	False	False	False	False	False	False	False
22497	False	False	False	False	False	False	False	False	False	False	False	False
22498	False	False	False	False	False	False	False	False	False	False	False	False
22499	False	False	False	False	False	False	False	False	False	False	False	False

22500 rows × 12 columns

In [36]:
```
X_train=df_train.drop(['y'],axis=1).values
y_train = df_train['y'].values
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [37]:
```
X_train
```

Out[37]:
```
array([[300000,      1,      3, ...,  89176,   4000,   3500],
       [ 20000,      1,      2, ...,  14074,   1313,    174],
       [ 90000,      2,      2, ...,      0,   2667,      0],
       ...,
       [ 50000,      1,      1, ...,      0,      0,      0],
       [ 70000,      2,      2, ...,  28039,   3000,   1200],
       [160000,      2,      2, ...,  14129,      0,   1500]], dtype=in
t64)
```

In [38]:
```
y_train
```

Out[38]:
```
array([0, 0, 0, ..., 0, 1, 1], dtype=int64)
```

splitting the train dataset into train and test sets further

In [40]:
```
from sklearn.model_selection import train_test_split
X_train_train, X_train_test, y_train_train, y_train_test = train_test_s
plit(X_train, y_train, test_size = .2, random_state = 1)
```

Implementing Logistic Regression algorithm

In [41]:
```
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression()
reg.fit(X_train_train, y_train_train)
y_pred = reg.predict(X_train_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [42]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_train_test, y_pred)
print(cm)
```

```
[[3330  146]
 [ 774  250]]
```

accuracy and confusion matrix are calculated

In [43]:
```
accuracy_score(y_train_test, y_pred)
```

Out[43]:
```
0.7955555555555556
```

Implementing Decision tree classifier and calculating the accuracy of it

In [44]:
```
#Fitting Decision Tree classifier to the training set
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)

classifier.fit(X_train_train, y_train_train)
#Predicting the test set result
y_pred_d= classifier.predict(X_train_test)
```

In [45]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm_d = confusion_matrix(y_train_test, y_pred_d)
print(cm_d)
```

```
[[2836  640]
 [ 623  401]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [46]:
```
accuracy_score(y_train_test, y_pred_d)
```

Out[46]:
```
0.7193333333333334
```

Implementing Random Forest Classifier and calculating the accuracy of it

In [47]:
```
#random forest classifier
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=500,criterion='entropy',rando
m_state=0)
model.fit(X_train_train,y_train_train)
#Predicting the test set result
y_pred_r= model.predict(X_train_test)
```

In [48]:
```
cm_r = confusion_matrix(y_train_test, y_pred_r)
print(cm_r)
```

```
[[3297  179]
 [ 654  370]]
```

In [49]:
```
accuracy_score(y_train_test, y_pred_r)
```

Out[49]:
```
0.8148888888888889
```

predicting the target value by giving input variables to predict function..Since, we got highest accuracy in random forest classifier, we use random forest model to predict the value

In [56]:
```
model.predict([['300000','1','3','2','31','0','0','80928','89176','400
0','3500']])
```

Out[56]:
```
array([0], dtype=int64)
```

In [57]:
```
df_test.head()
```

Out[57]:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X14	X15	X16	X17	X18	X19
0	30000	1	2	2	25	0	0	0	0	0	...	11581	12580	13716	14828	1500	200
1	150000	2	1	2	26	0	0	0	0	0	...	116684	101581	77741	77264	4486	423
2	70000	2	3	1	32	0	0	0	0	0	...	68530	69753	70111	70212	2431	311
3	130000	1	3	2	49	0	0	0	0	0	...	16172	16898	11236	6944	1610	180
4	50000	2	2	2	36	0	0	0	0	0	...	42361	19574	20295	19439	2000	150

5 rows × 23 columns

In [58]:
```
df_train.head()
```

Out[58]:

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23	y
0	300000	1	3	2	31	0	0	80928	89176	4000	3500	0
1	20000	1	2	2	24	0	0	15730	14074	1313	174	0
2	90000	2	2	2	35	1	1	2667	0	2667	0	0
3	300000	2	2	1	40	1	2	0	0	0	0	0
4	70000	2	2	2	36	0	0	68028	29443	3340	1297	1

Now we need to find the target variable for test dataset ie finding whether the clients are credible or not...We have trained the train dataset and implemented to calculate the credibility in test dataset..

In [59]:
```
df_test.drop(['X19','X20','X21','X22','X8','X9','X10','X11','X13','X14'
,'X15','X16'],axis=1,inplace=True)
```

In [60]:
```
df_test.head()
```

Out[60]:

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	X1	X2	X3	X4	X5	X6	X7	X12	X17	X18	X23
0	30000	1	2	2	25	0	0	8864	14828	1500	2000
1	150000	2	1	2	26	0	0	136736	77264	4486	2669
2	70000	2	3	1	32	0	0	70122	70212	2431	2554
3	130000	1	3	2	49	0	0	20678	6944	1610	4408
4	50000	2	2	2	36	0	0	94228	19439	2000	1000

In [61]:
```
model.predict([['30000','1','2','2','25','0','0','8864','14828','1500',
'2000']])
```

Out[61]:
```
array([0], dtype=int64)
```

In [63]:
```
model.predict([['70000','2','3','1','32','0','1','20678','6944','1610',
'4408']])
```

Out[63]:
```
array([0], dtype=int64)
```

So, we have analysed the model and implemented it with various algorithms in which the highest accuracy is found in random forest classifier...So we could found the clients are credible or not!!!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD