ML M2

AVANI NARVEKAR J042

IMPORTS

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

LOADING DATA

In [2]:
```
cur_path = os.getcwd()
train_data = pd.read_csv(cur_path+"\\train.csv")
test_data= pd.read_csv(cur_path+"\\test.csv")
```

CHECKING AND CLEANING

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

Out[3]:

	id	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	
0	86603	168005	7040268	47	19	single	rented	yes	Softwa
1	232884	150874	8437491	22	2	single	rented	no	Indus
2	72597	26384	1729641	38	12	single	rented	no	H
3	71625	100138	3424035	24	2	single	rented	no	
4	43561	183275	6712876	51	17	single	rented	yes	

In [4]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201600 entries, 0 to 201599
Data columns (total 14 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   id                 201600 non-null  int64 
 1   Id                 201600 non-null  int64 
 2   Income             201600 non-null  int64 
 3   Age                201600 non-null  int64 
 4   Experience         201600 non-null  int64 
 5   Married/Single     201600 non-null  object
 6   House_Ownership    201600 non-null  object
 7   Car_Ownership      201600 non-null  object
 8   Profession         201600 non-null  object
 9   CITY               201600 non-null  object
 10  STATE              201600 non-null  object
 11  CURRENT_JOB_YRS    201600 non-null  int64 
 12  CURRENT_HOUSE_YRS  201600 non-null  int64 
 13  Risk_Flag          201600 non-null  int64 
```

```
dtypes: int64(8), object(6)
memory usage: 21.5+ MB
```

In [5]:
```
train_data.isnull().sum()
```

Out[5]:
```
id                 0
Id                 0
Income             0
Age                0
Experience         0
Married/Single     0
House_Ownership    0
Car_Ownership      0
Profession         0
CITY               0
STATE              0
CURRENT_JOB_YRS    0
CURRENT_HOUSE_YRS  0
Risk_Flag          0
dtype: int64
```

This dataset has no null values.

EDA

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

Out[6]:

	id	Id	Income	Age	Experience	CURRENT_JOB_YRS
count	201600.000000	201600.000000	2.016000e+05	201600.000000	201600.000000	201600.000000
mean	126067.093641	126056.352609	5.001826e+06	49.931473	10.087361	6.333914
std	72749.062863	72796.938606	2.879258e+06	17.057638	6.001094	3.642897
min	0.000000	1.000000	1.031000e+04	21.000000	0.000000	0.000000
25%	62998.500000	63017.500000	2.504515e+06	35.000000	5.000000	3.000000
50%	126142.500000	126082.000000	5.004938e+06	50.000000	10.000000	6.000000
75%	189025.250000	189227.250000	7.488205e+06	65.000000	15.000000	9.000000
max	251999.000000	252000.000000	9.999400e+06	79.000000	20.000000	14.000000

In [7]:
```
train_data.shape
```

Out[7]:
```
(201600, 14)
```

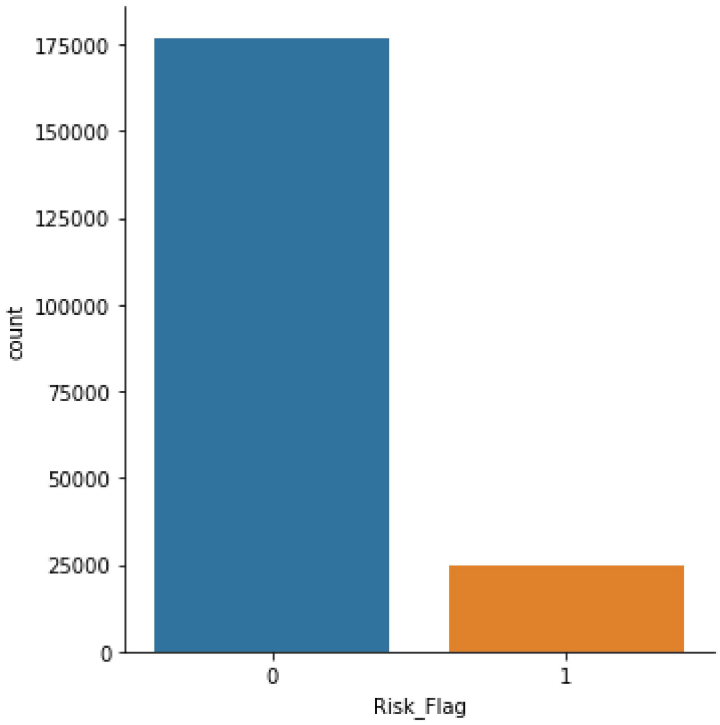
In [8]:
```
train_data.Risk_Flag.value_counts()
```

Out[8]:
```
0    176847
1     24753
Name: Risk_Flag, dtype: int64
```

In [9]:
```
sns.catplot('Risk_Flag', data=train_data, kind='count')
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will result in a
n error or misinterpretation.
  warnings.warn(
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x29abfab1a00>



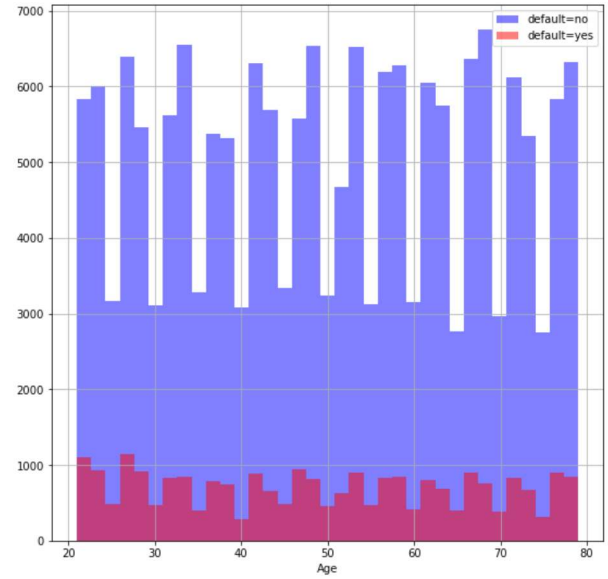
In [10]:
```
numerical_cols = ['Age','Experience','CURRENT_JOB_YRS','CURRENT_HOUSE_YRS']
```

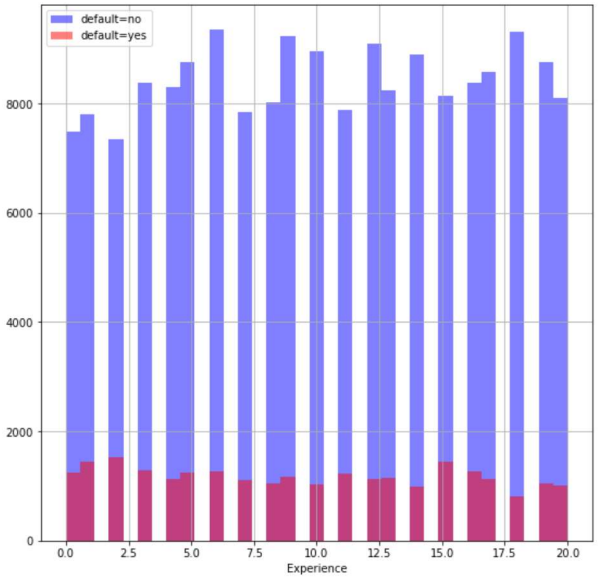
In [11]:
```
train_data.columns
```

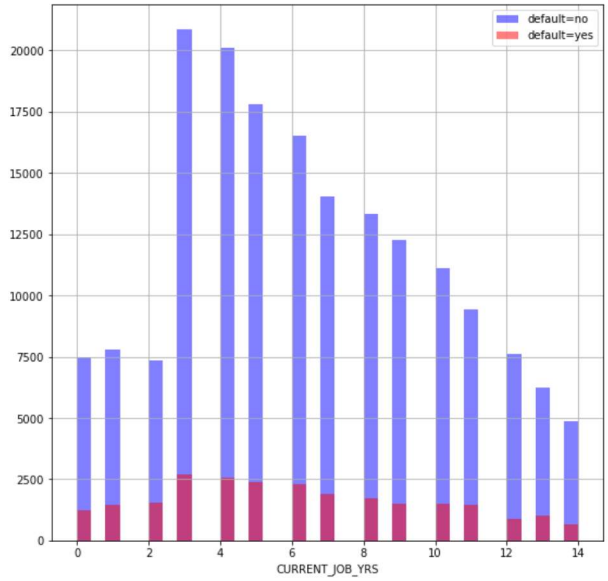
Out[11]:
```
Index(['id', 'Id', 'Income', 'Age', 'Experience', 'Married/Single',
       'House_Ownership', 'Car_Ownership', 'Profession', 'CITY', 'STATE',
       'CURRENT_JOB_YRS', 'CURRENT_HOUSE_YRS', 'Risk_Flag'],
      dtype='object')
```

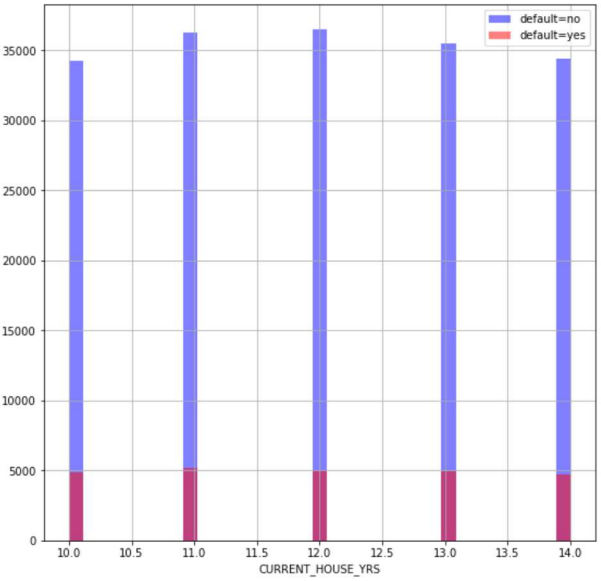
In [12]:
```
# Visulazing the distibution of the data for every feature
plt.figure(figsize=(20, 20))

for i, column in enumerate(numerical_cols, 1):
    plt.subplot(2,2, i)
    train_data[train_data["Risk_Flag"] == 0][column].hist(bins=35, color='blue', label=
    train_data[train_data["Risk_Flag"] == 1][column].hist(bins=35, color='red', label='
    plt.legend()
    plt.xlabel(column)
```









ENCODING: we need to encode the categorical data.

In [13]:
```
from sklearn.preprocessing import LabelEncoder
labelEncoder = LabelEncoder()
```

In [14]:
```
# Accommodate data into dataVariables
data = train_data

# Encode the object data to type int
for e in data.columns:
    if data[e].dtype == 'object':
        labelEncoder.fit(list(data[e].values))
        data[e] = labelEncoder.transform(data[e].values)

        # Accommodate the data that has been changed
        train_data = data
```

In [15]:
```
data = test_data
```

```
# Encode the object data to type int
for e in data.columns:
    if data[e].dtype == 'object':
        labelEncoder.fit(list(data[e].values))
        data[e] = labelEncoder.transform(data[e].values)

        # Accommodate the data that has been changed
        test_data = data
```

In [16]:
```
train_data.head()
```

Out[16]:

	id	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profes
0	86603	168005	7040268	47	19	1	2	1	
1	232884	150874	8437491	22	2	1	2	0	
2	72597	26384	1729641	38	12	1	2	0	
3	71625	100138	3424035	24	2	1	2	0	
4	43561	183275	6712876	51	17	1	2	1	

In [17]:
```
test_data.head()
```

Out[17]:

	id	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession
0	5	248505	8124352	42	20	1	2	0	38
1	7	123419	2417756	64	18	0	2	0	17
2	14	60805	8953239	65	13	1	2	0	14
3	26	251548	6208078	59	13	1	2	0	17
4	31	111835	5863246	38	9	1	2	0	27

CHECKING CORR

In [18]:
```
corr = train_data.corr()
corr
```

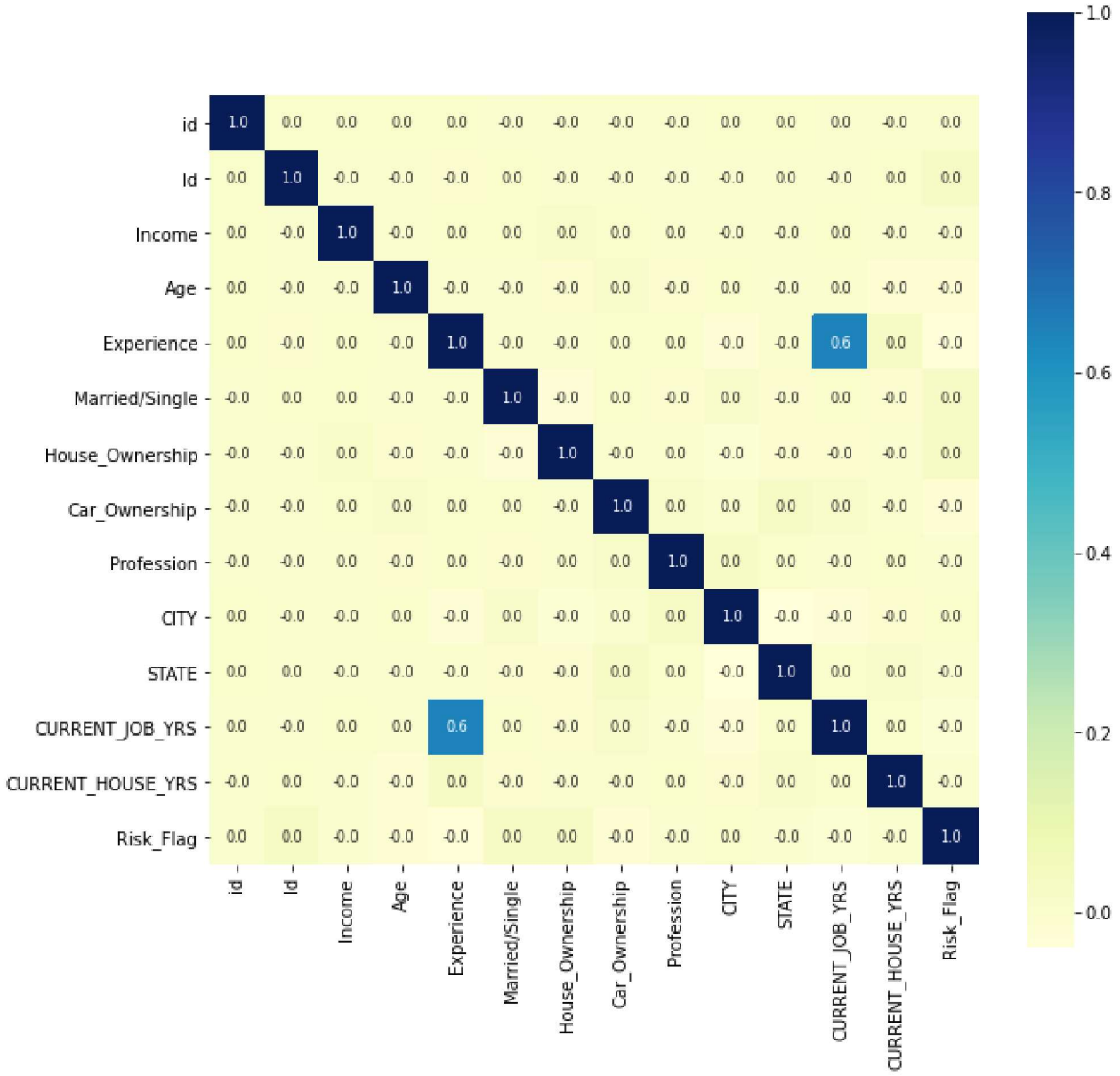
Out[18]:

	id	Id	Income	Age	Experience	Married/Single	House_Ow
id	1.000000	0.002214	0.001732	0.005617	0.006538	-0.003098	-(
Id	0.002214	1.000000	-0.000842	-0.000511	-0.008042	0.002311	-(
Income	0.001732	-0.000842	1.000000	-0.001089	0.005846	0.002510	(
Age	0.005617	-0.000511	-0.001089	1.000000	-0.002271	-0.005676	-(
Experience	0.006538	-0.008042	0.005846	-0.002271	1.000000	-0.000673	-(
Married/Single	-0.003098	0.002311	0.002510	-0.005676	-0.000673	1.000000	-(

	id	Id	Income	Age	Experience	Married/Single	House_Ow
House_Ownership	-0.001947	-0.001879	0.015870	-0.010334	-0.005589	-0.023968	1
Car_Ownership	-0.002715	-0.002160	0.004682	0.010570	0.008323	0.003281	-(
Profession	-0.001100	-0.004873	0.000237	-0.009944	0.001473	-0.007242	(
CITY	0.002130	-0.000037	-0.002659	0.001268	-0.024718	0.012905	-(
STATE	0.007574	0.001110	-0.001360	-0.004775	-0.001119	-0.008380	-(
CURRENT_JOB_YRS	0.000520	-0.005194	0.006792	0.000227	0.645558	0.005592	-(
CURRENT_HOUSE_YRS	-0.000836	0.001514	-0.003091	-0.019816	0.019764	-0.007356	-(
Risk_Flag	0.000758	0.033321	-0.002981	-0.022061	-0.034390	0.021256	(

In [19]:
```
# Constructing a heatmap to understand the correlation
plt.figure(figsize=(10, 10))
sns.heatmap(corr, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 8},
plt.plot()
```

Out[19]: []



In [20]:
```
corr["Risk_Flag"].sort_values(ascending=False)
```

Out[20]:
```
Risk_Flag            1.000000
Id                   0.033321
House_Ownership      0.024007
Married/Single       0.021256
CITY                 0.004116
id                   0.000758
Income              -0.002981
CURRENT_HOUSE_YRS   -0.002999
STATE               -0.003839
Profession          -0.004271
CURRENT_JOB_YRS     -0.017203
Age                 -0.022061
Car_Ownership       -0.022240
Experience          -0.034390
Name: Risk_Flag, dtype: float64
```

SPLIT

In [21]:
```
from sklearn.model_selection import train_test_split
```

```
X = train_data.drop(['id','Id','Risk_Flag'], axis=1)
y= train_data['Risk_Flag']
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state =
```

In [23]:
```
testdataX =  test_data.drop(['id','Id'], axis=1)
```

TRAIN AND PREDICT

random forest

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=500, random_state=12, min_samples_leaf=2, c
model.fit(X_train, y_train)
y_predict = model.predict_proba(X_test)[:,1]
```

In [25]:
```
from sklearn.metrics import roc_auc_score
print(roc_auc_score(y_test, y_predict))
```

```
0.9374842078812401
```

APPLYING THE MODEL ON TEST DATA

In [26]:
```
y_pred = model.predict_proba(testdataX)[:,1]
```

In [27]:
```
test_copy = test_data.copy()
```

In [28]:
```
test_copy['Risk_Flag'] = y_pred
test_copy.head()
```

Out[28]:

	id	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession
0	5	248505	8124352	42	20	1	2	0	38
1	7	123419	2417756	64	18	0	2	0	17
2	14	60805	8953239	65	13	1	2	0	14
3	26	251548	6208078	59	13	1	2	0	17
4	31	111835	5863246	38	9	1	2	0	27

In [29]:
```
sub1 = test_copy[['id','Risk_Flag']]
```

In [30]:
```
sub1 = sub1.to_csv('submission8.csv',index = False)
```