MileStonePredictiveAnalysis

July 30, 2022

1 Felipe Castillo

2 MileStone

3 Predictive Analytics

4 07/29/2022

[1]:
```
import pandas as pd
import numpy as np
import os
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score␣
↪,accuracy_score,mean_squared_error,recall_score,f1_score,precision_score,confusion_matrix,r
from sklearn.model_selection import train_test_split
```

Will I be able to answer the questions I want to answer with the data I have?

What visualizations are especially useful for explaining my data?

Do I need to adjust the data and/or driving questions?

Do I need to adjust my model/evaluation choices?

Are my original expectations still reasonable?

Please submit Milestone 3 in Blackboard under the group submission link.

This should be submitted through the group assignment submission regardless if it is an independent project or multi-person group.

[2]:
```
# Change directory to work with data set
os.chdir("C:\DataScience_DSC_630\Week2")
```

[3]:
```
#load data set
bank_loan_dt = pd.read_csv("Training Data.csv")
```

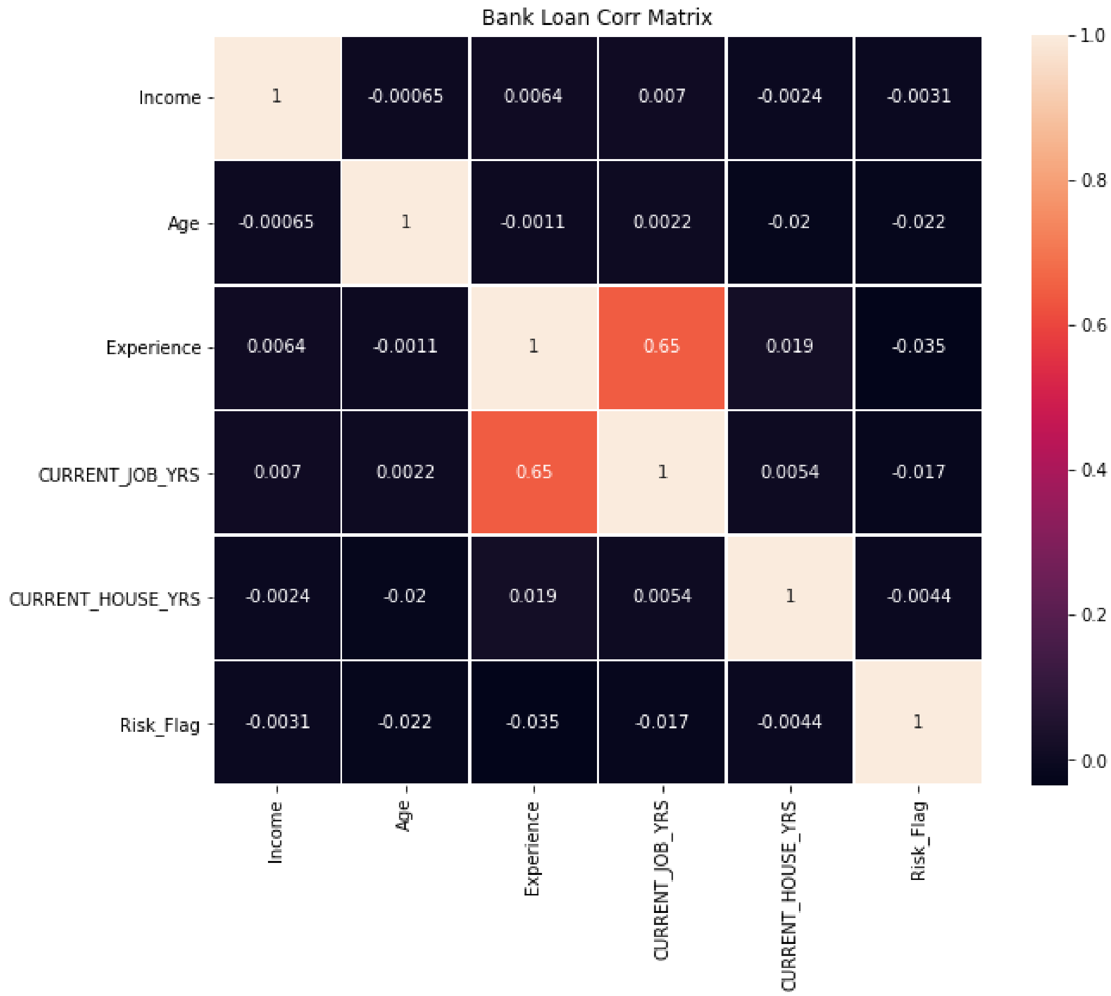
[4]:
```
bank_loan_dt.head(3)
```

[4]:
```
   Id   Income  Age  Experience Married/Single House_Ownership Car_Ownership  \
0   1  1303834   23           3         single          rented            no
1   2  7574516   40          10         single          rented            no
2   3  3991815   66           4        married          rented            no

            Profession       CITY           STATE  CURRENT_JOB_YRS  \
0  Mechanical_engineer       Rewa  Madhya_Pradesh                3
1   Software_Developer   Parbhani     Maharashtra                9
2     Technical_writer  Alappuzha          Kerala                4

   CURRENT_HOUSE_YRS  Risk_Flag
0                 13          0
1                 13          0
2                 10          0
```

[5]:
```
#droping Id does not hold any useful information
bank_loan_dt = bank_loan_dt.drop(['Id'] , axis= 1)
```

[6]:
```
corrMatrix = bank_loan_dt.corr()
```

[7]:
```
fig, ax = plt.subplots(figsize=(10,8))
sns.heatmap(corrMatrix, annot= True,linewidths=.5, ax=ax)
plt.title("Bank Loan Corr Matrix")
plt.show()
```



[8]:
```
list(enumerate(bank_loan_dt))
```

[8]:
```
[(0, 'Income'),
 (1, 'Age'),
 (2, 'Experience'),
 (3, 'Married/Single'),
 (4, 'House_Ownership'),
 (5, 'Car_Ownership'),
 (6, 'Profession'),
 (7, 'CITY'),
 (8, 'STATE'),
 (9, 'CURRENT_JOB_YRS'),
 (10, 'CURRENT_HOUSE_YRS'),
 (11, 'Risk_Flag')]
```

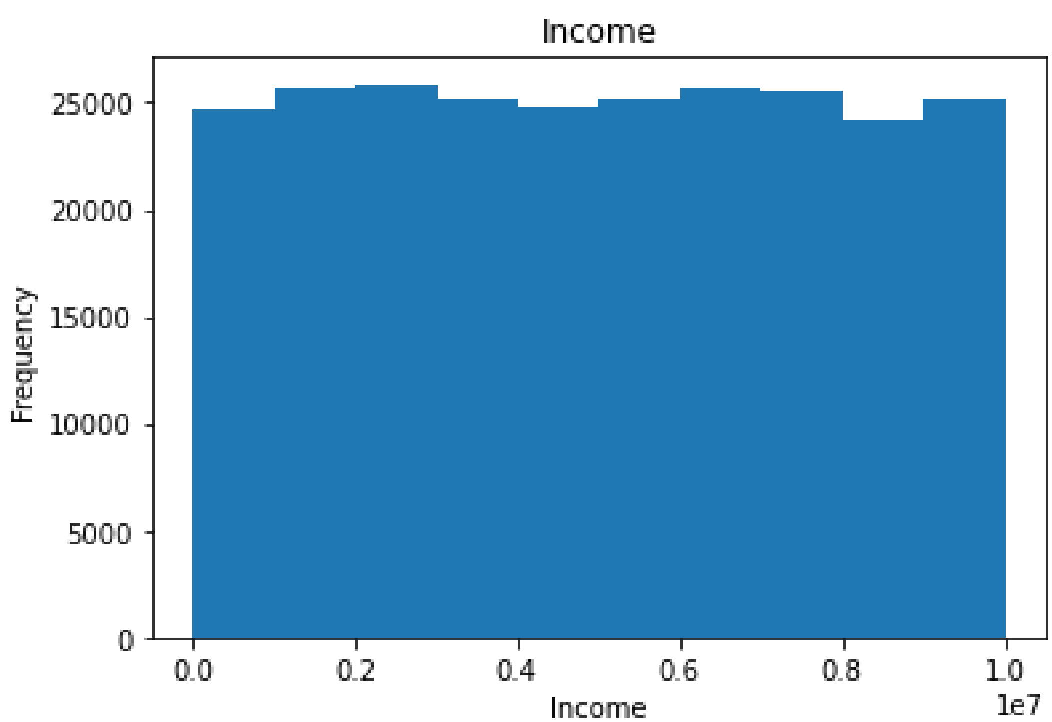
[9]:
```
#Will wold all numerical values
bank_loan_numerical =  bank_loan_dt.drop(['Profession' ,'CITY','STATE','Married/
↪Single','House_Ownership','Car_Ownership','Risk_Flag'] , axis= 1).copy()
```

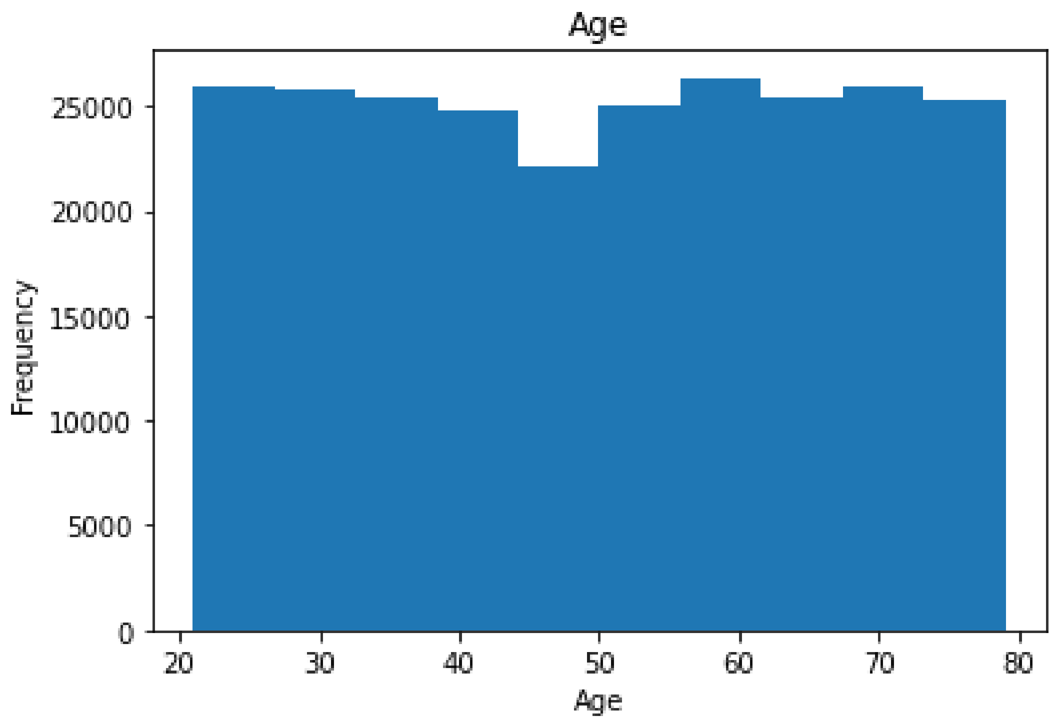
[10]:
```
#Getting a basic understanding with the numerical data
bank_loan_numerical.head(2)
```

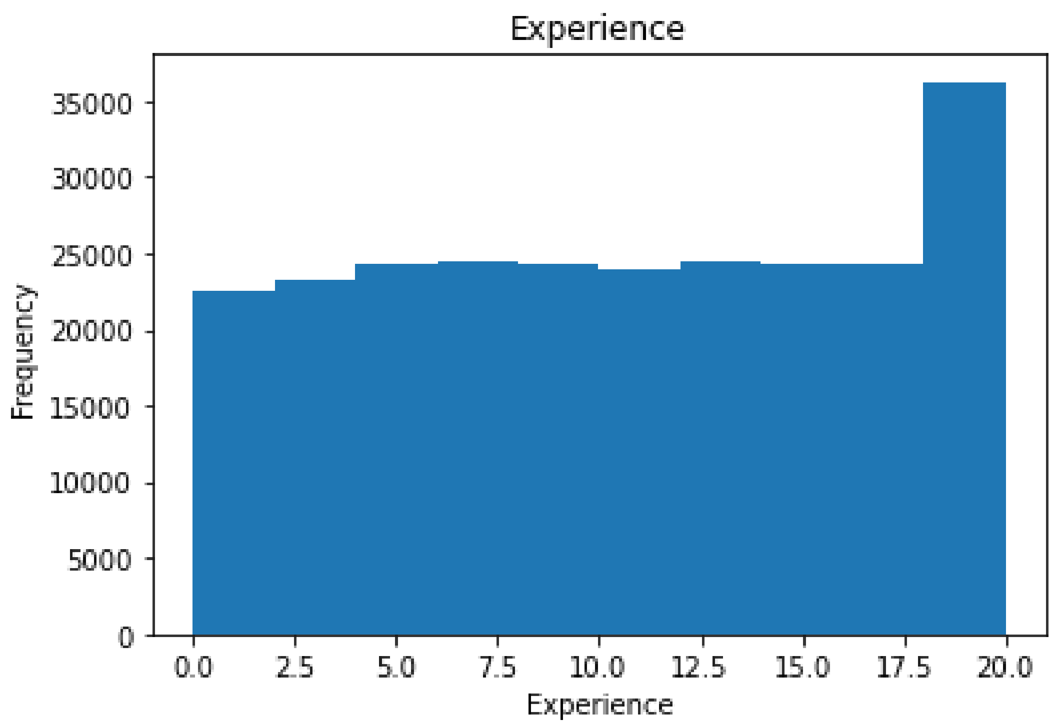
[10]:

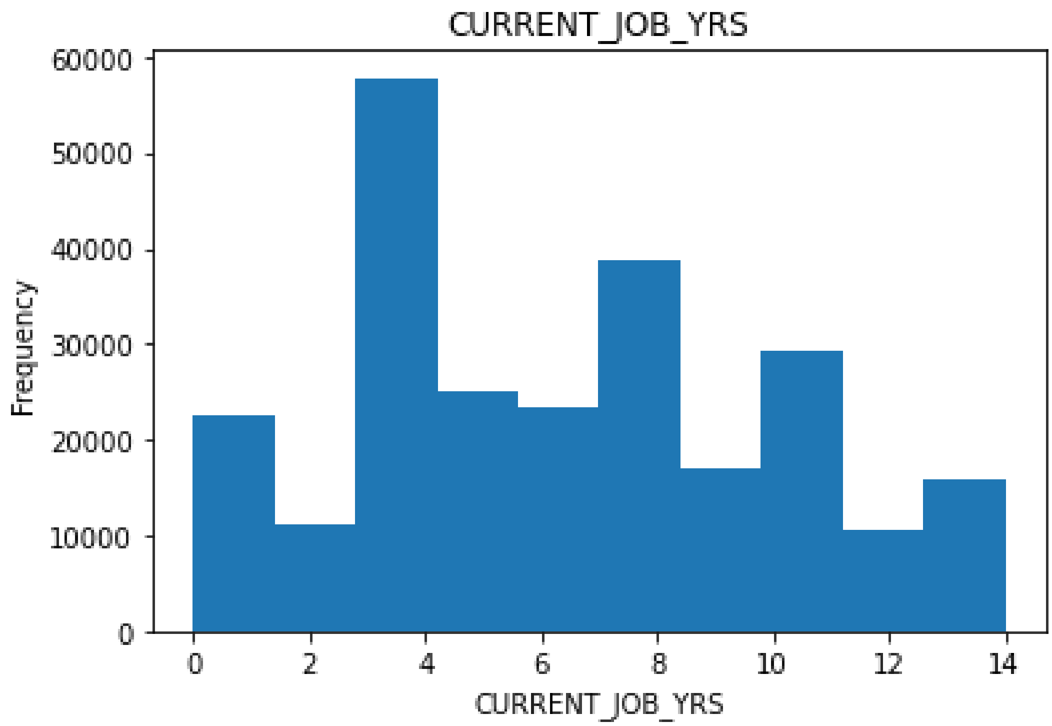
	Income	Age	Experience	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS
0	1303834	23	3	3	13
1	7574516	40	10	9	13

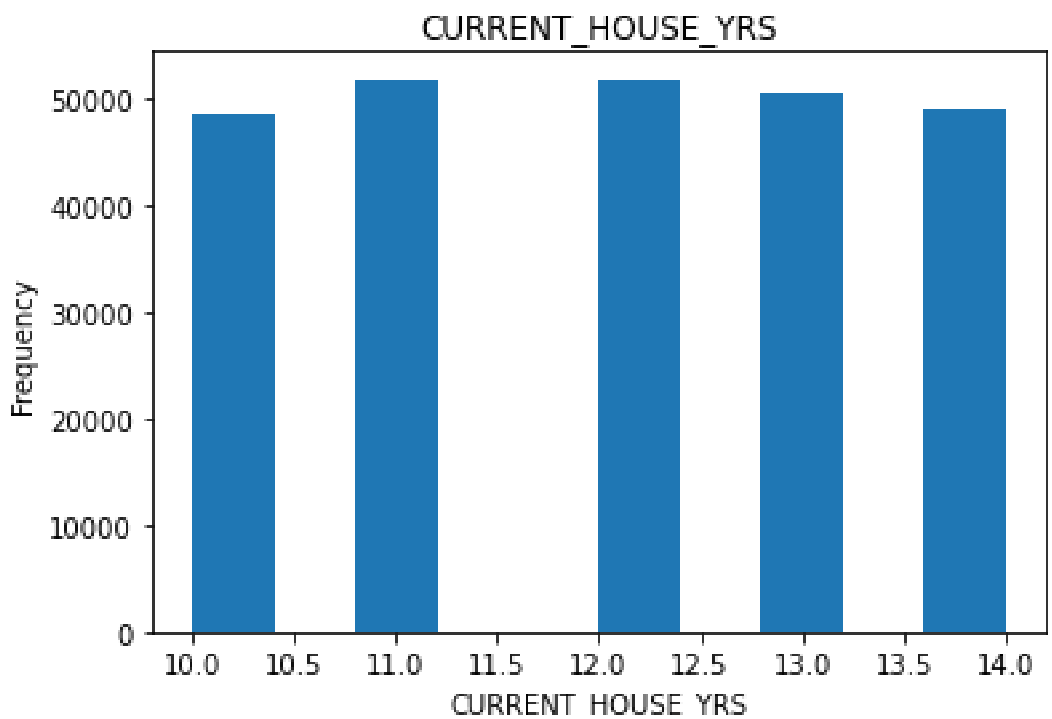
[12]:
```
#builing a tuple struct that is numbered, and call directly by index
for i in list(enumerate(bank_loan_numerical)):
    plt.title(i[1])
    plt.hist(bank_loan_dt[i[1]])
    plt.ylabel("Frequency")
    plt.xlabel(i[1])
    plt.show()
```




Age
Frequency
25000
20000
15000
10000
5000
0
20
30
40
50
60
70
80
Age

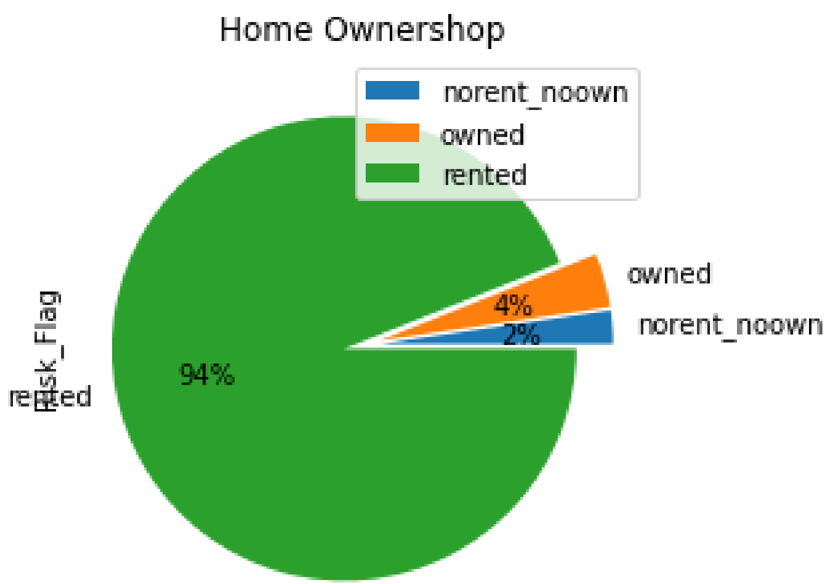

Experience
Frequency
35000
30000
25000
20000
15000
10000
5000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Experience


CURRENT_JOB_YRS
60000
50000
40000
30000
20000
10000
0
Frequency
0
2
4
6
8
10
12
14
CURRENT_JOB_YRS


CURRENT_HOUSE_YRS
50000
40000
30000
20000
10000
0
Frequency
10.0
10.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0
CURRENT_HOUSE_YRS

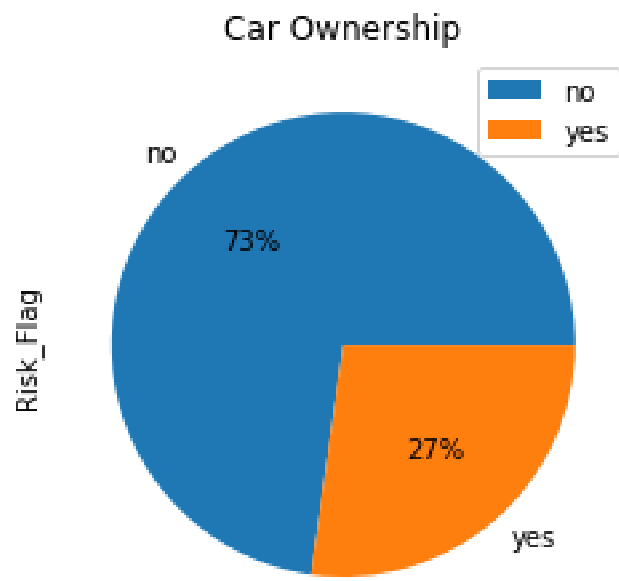
[16]:
```
#Seeing the flagged risk by home ownership
bank_loan_dt.groupby(['House_Ownership']).sum().plot(kind='pie', y='Risk_Flag',␣
↪autopct='%1.0f%%', explode = (0.08, 0.08, 0.08))
plt.title("Home Ownershop")
```

[16]: Text(0.5, 1.0, 'Home Ownershop')



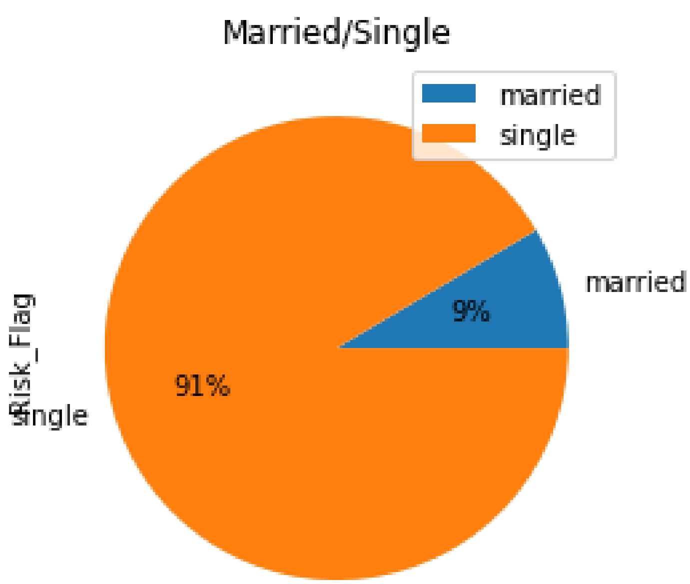
[17]:
```
#Seeing the flagged risk by car ownership
bank_loan_dt.groupby(['Car_Ownership']).sum().plot(kind='pie', y='Risk_Flag',␣
↪autopct='%1.0f%%')
plt.title("Car Ownership")
```

[17]: Text(0.5, 1.0, 'Car Ownership')




```
[14]: #Seeing the flagged risk by realtionship status
bank_loan_dt.groupby(['Married/Single']).sum().plot(kind='pie', y='Risk_Flag',␣
 ↪autopct='%1.0f%%')
plt.title("Married/Single")
```

```
[14]: Text(0.5, 1.0, 'Married/Single')
```



[15]:
```
#Identifying risk by profession
profession_table = pd.pivot_table(data= bank_loan_dt, index='Profession',␣
↪values='Risk_Flag')
profession_table.head(5)
```

[15]:
```
                        Risk_Flag
Profession
Air_traffic_controller   0.135391
Analyst                  0.121465
Architect                0.131200
Army_officer             0.152113
Artist                   0.122609
```

[16]:
```
#making a dictionary mapped to flagged percentage
#profession is key risk is value
risk_factors_dic = {}

#Index is profession
#For each profession by index
for i in range(len(profession_table.index)):
    #mapped dictionary by profession and by the risk profession score
    risk_factors_dic[profession_table.index[i]] =␣
↪profession_table['Risk_Flag'][i]
print(risk_factors_dic)
```

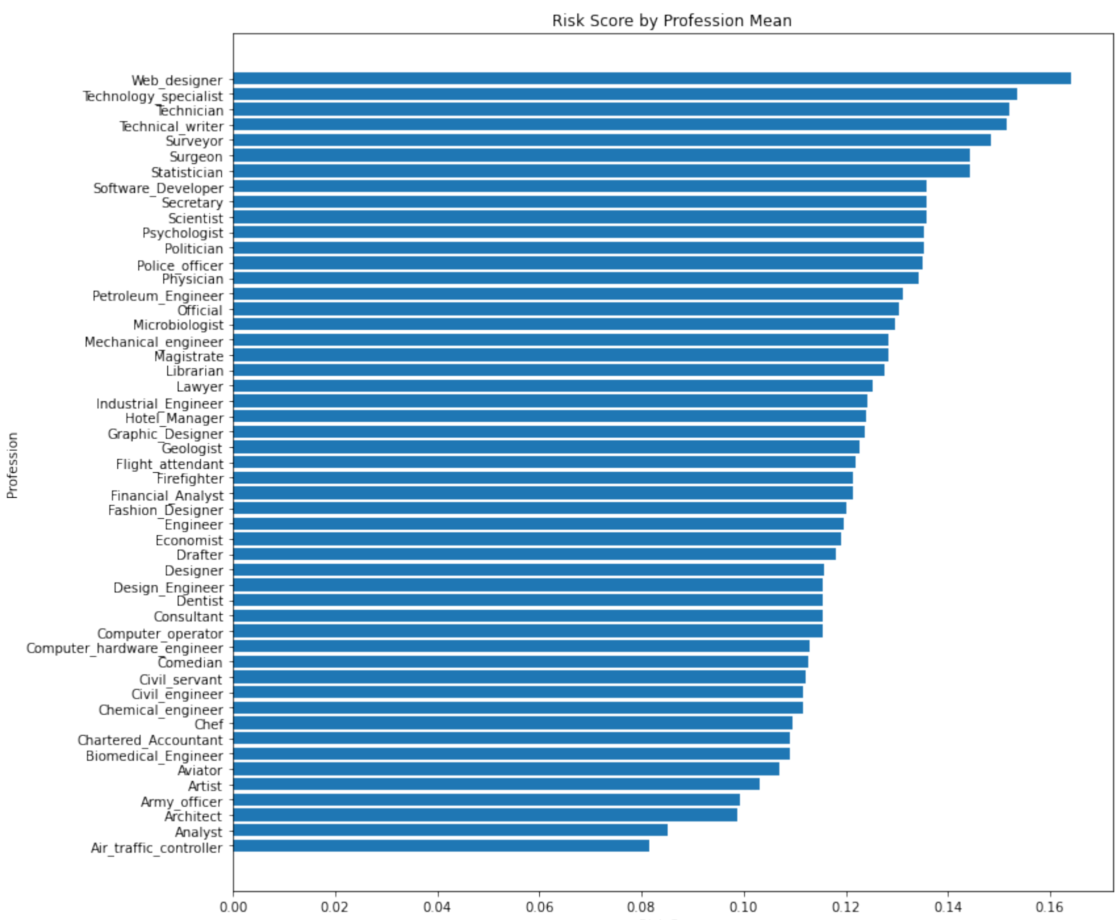
```
{'Air_traffic_controller': 0.1353910244271918, 'Analyst': 0.12146529562982006,
'Architect': 0.13120034356882113, 'Army_officer': 0.15211328041192876, 'Artist':
0.1226085167660975, 'Aviator': 0.13493064312736444, 'Biomedical_Engineer':
0.12755997659449972, 'Chartered_Accountant': 0.15357222345871355, 'Chef':
0.12146709816612729, 'Chemical_engineer': 0.11162343900096061, 'Civil_engineer':
0.1358318890814558, 'Civil_servant': 0.11579424427826875, 'Comedian':
0.11960448754516068, 'Computer_hardware_engineer': 0.12844378257632166,
'Computer_operator': 0.12404809619238477, 'Consultant': 0.1252079866888519,
'Dentist': 0.109577582601422, 'Design_Engineer': 0.1069993656164094, 'Designer':
0.10917790343627665, 'Drafter': 0.1128941966784848, 'Economist':
0.09927837305926088, 'Engineer': 0.11808300395256917, 'Fashion_Designer':
0.11538461538461539, 'Financial_Analyst': 0.10315463518482679, 'Firefighter':
0.13578877301974707, 'Flight_attendant': 0.12363494539781592, 'Geologist':
0.144263698630137, 'Graphic_Designer': 0.11536972512582269, 'Hotel_Manager':
0.13538045577443028, 'Industrial_Engineer': 0.09866666666666667, 'Lawyer':
0.1295143212951432, 'Librarian': 0.11257562662057044, 'Magistrate':
0.12002986746313235, 'Mechanical_engineer': 0.11155836687751582,
'Microbiologist': 0.12435976234378202, 'Official': 0.1357964276975777,
```

```
'Petroleum_Engineer': 0.08510216226939099, 'Physician': 0.11918751049185831,
'Police_officer': 0.16405163853028798, 'Politician': 0.11225728155339806,
'Psychologist': 0.12189239332096476, 'Scientist': 0.14432127170048106,
'Secretary': 0.13040901007705988, 'Software_Developer': 0.1484266772214526,
'Statistician': 0.11557009989665863, 'Surgeon': 0.11546521374685666, 'Surveyor':
0.15146372507424694, 'Technical_writer': 0.134167468719923, 'Technician':
0.12828947368421054, 'Technology_specialist': 0.08148617268313278,
'Web_designer': 0.10913470446544377}
```

[17]:
```python
#Easier to work with data frame , mapping dictionary to dataframe
risk_factor_df = pd.DataFrame(list(risk_factors_dic.items()),␣
↪columns=['Careers','Risk_Score'])
```

[18]:
```python
plt.figure(figsize=(12,12))
plt.title("Risk Score by Profession Mean")
plt.ylabel("Profession")
plt.xlabel("Risk Score")
plt.barh( risk_factor_df['Careers'],sorted(risk_factor_df['Risk_Score']))
```

[18]:
```
<BarContainer object of 51 artists>
```



[19]:
```
bank_loan_dt.head(3)
```

[19]:
```
    Income  Age  Experience Married/Single House_Ownership Car_Ownership  \
0  1303834   23           3         single          rented            no
1  7574516   40          10         single          rented            no
2  3991815   66           4        married          rented            no

            Profession       CITY           STATE  CURRENT_JOB_YRS  \
0  Mechanical_engineer       Rewa  Madhya_Pradesh                3
1   Software_Developer   Parbhani     Maharashtra                9
2     Technical_writer  Alappuzha          Kerala                4

   CURRENT_HOUSE_YRS  Risk_Flag
0                 13          0
1                 13          0
2                 10          0
```

[19]:
```
age_group  = bank_loan_dt.groupby(['Age'])
```

[20]:
```
#Identifying risk by age and seeing if there is a trend.
#Risk will be determined by levels of salary.
age_group.first()
```

[20]:
```
      Income  Experience Married/Single House_Ownership Car_Ownership  \
Age
21   4128828          10         single          rented            no
22   6623263           4         single          rented            no
23   1303834           3         single          rented            no
24   7566849          17         single          rented           yes
25   6868118          16         single          rented            no
26   5023035          10         single          rented           yes
27   4260004           5         single    norent_noown            no
28   9120988           9         single          rented            no
29   1240330          18        married          rented           yes
30   8390825          11         single          rented            no
31   4386333          16         single          rented            no
32   7433875          12         single          rented           yes
33   1706172           2         single          rented            no
34   2217063           3         single          rented            no
35   5083653          14         single          rented           yes
36   9236505          19         single          rented            no
37   6501716           5         single          rented            no
38   6063428           6        married          rented            no
39   5694236           2        married          rented           yes
40   7574516          10         single          rented            no
```

41	6256451	2	single	rented	yes
42	9585696	13	single	rented	yes
43	9603186	5	single	rented	no
44	7992060	15	single	rented	no
45	7537675	4	single	rented	no
46	1885923	16	single	rented	no
47	5768871	11	single	rented	no
48	9420838	6	single	rented	no
49	4047079	7	single	rented	yes
50	6506739	4	single	rented	no
51	9984878	18	single	rented	yes
52	3939397	19	single	rented	yes
53	3970273	14	single	rented	no
54	9225468	14	single	rented	no
55	9086933	7	single	rented	no
56	3666346	12	single	rented	no
57	8043880	12	single	rented	no
58	3954973	14	married	rented	no
59	6944134	5	single	owned	no
60	6227811	14	single	owned	no
61	5165629	0	single	rented	no
62	3159260	4	single	rented	no
63	9760667	17	single	rented	no
64	6915937	0	single	rented	no
65	6245331	6	single	rented	no
66	3991815	4	married	rented	no
67	1213131	8	single	rented	no
68	9311486	9	single	rented	no
69	4432483	6	single	rented	no
70	4269729	8	single	rented	yes
71	7315840	8	married	rented	no
72	9157379	13	single	rented	yes
73	2471915	18	single	rented	no
74	3634814	4	single	rented	no
75	8996641	12	single	rented	no
76	1797876	20	single	norent_noown	no
77	9625415	15	married	rented	no
78	4634680	7	single	rented	no
79	9576258	18	single	rented	yes

	Profession	CITY	STATE \
Age			
21	Computer_hardware_engineer	Khammam	Telangana
22	Designer	Adoni	Andhra_Pradesh
23	Mechanical_engineer	Rewa	Madhya_Pradesh
24	Flight_attendant	Kota[6]	Rajasthan
25	Secretary	Danapur	Bihar

26	Petroleum_Engineer	Madurai	Tamil_Nadu
27	Police_officer	Sagar	Madhya_Pradesh
28	Physician	Erode[17]	Tamil_Nadu
29	Consultant	Gopalpur	West_Bengal
30	Secretary	Bidhannagar	West_Bengal
31	Physician	Shimoga	Karnataka
32	Fashion_Designer	Chennai	Tamil_Nadu
33	Economist	Jamnagar	Gujarat
34	Computer_hardware_engineer	Chinsurah	West_Bengal
35	Statistician	Ambattur	Tamil_Nadu
36	Chemical_engineer	Ongole	Andhra_Pradesh
37	Civil_servant	Nagpur	Maharashtra
38	Dentist	Ambattur	Tamil_Nadu
39	Economist	Anantapuram[24]	Andhra_Pradesh
40	Software_Developer	Parbhani	Maharashtra
41	Software_Developer	Bhubaneswar	Odisha
42	Official	Anantapuram[24]	Andhra_Pradesh
43	Microbiologist	Munger	Bihar
44	Fashion_Designer	Nagaon	Assam
45	Graphic_Designer	Gopalpur	West_Bengal
46	Magistrate	Thanjavur	Tamil_Nadu
47	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu
48	Technical_writer	Madurai	Tamil_Nadu
49	Dentist	Indore	Madhya_Pradesh
50	Politician	Ahmedabad	Gujarat
51	Comedian	Jammu[16]	Jammu_and_Kashmir
52	Flight_attendant	Chennai	Tamil_Nadu
53	Air_traffic_controller	Satna	Madhya_Pradesh
54	Surveyor	Secunderabad	Telangana
55	Air_traffic_controller	Saharanpur	Uttar_Pradesh
56	Politician	Bhusawal	Maharashtra
57	Financial_Analyst	Kollam	Kerala
58	Librarian	Tiruppur	Tamil_Nadu
59	Graphic_Designer	Gulbarga	Karnataka
60	Economist	Sirsa	Haryana
61	Aviator	North_Dumdum	West_Bengal
62	Petroleum_Engineer	Panihati	West_Bengal
63	Chartered_Accountant	Khandwa	Madhya_Pradesh
64	Civil_servant	Jalgaon	Maharashtra
65	Financial_Analyst	Eluru[25]	Andhra_Pradesh
66	Technical_writer	Alappuzha	Kerala
67	Psychologist	Agartala	Tripura
68	Lawyer	Panchkula	Haryana
69	Hotel_Manager	Kulti	West_Bengal
70	Fashion_Designer	Unnao	Uttar_Pradesh
71	Air_traffic_controller	Kamarhati	West_Bengal
72	Design_Engineer	Ajmer	Rajasthan

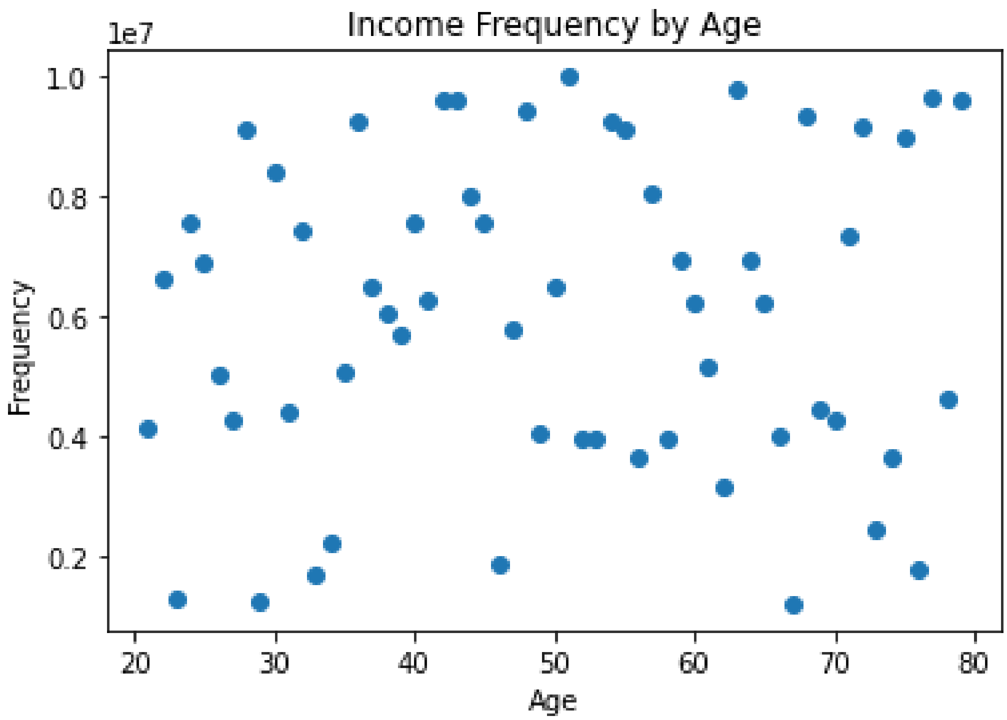
73	Chemical_engineer	Purnia[26]	Bihar
74	Fashion_Designer	Durgapur	West_Bengal
75	Microbiologist	Patiala	Punjab
76	Mechanical_engineer	Erode[17]	Tamil_Nadu
77	Secretary	Amravati	Maharashtra
78	Flight_attendant	Hajipur[31]	Bihar
79	Air_traffic_controller	Jamshedpur	Jharkhand

	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
Age			
21	10	12	0
22	4	14	0
23	3	13	0
24	11	11	0
25	13	13	1
26	9	13	0
27	5	13	0
28	9	12	0
29	12	14	0
30	7	10	0
31	3	12	0
32	11	10	1
33	2	14	0
34	3	11	0
35	12	11	0
36	6	14	0
37	5	13	0
38	6	13	0
39	2	10	0
40	9	13	0
41	2	12	1
42	3	12	0
43	5	13	1
44	4	11	0
45	4	14	0
46	8	14	1
47	3	14	1
48	6	10	1
49	5	14	0
50	4	11	0
51	8	12	0
52	3	10	0
53	4	12	0
54	8	10	0
55	7	13	0
56	12	11	1
57	8	10	0

58	8	12	0
59	5	11	0
60	12	12	0
61	0	12	1
62	4	10	0
63	13	12	1
64	0	12	0
65	6	12	0
66	4	10	0
67	8	11	0
68	9	12	0
69	6	10	0
70	8	13	0
71	8	14	0
72	9	10	0
73	14	13	0
74	4	11	0
75	12	13	0
76	11	14	0
77	9	10	0
78	7	12	0
79	6	11	0

[22]:
```
#Getting and idea on distribution
#Example of earning between ages.
# Trying to establish a connection with income, doesnt look like age is a␣
↪factor to earning.
plt.scatter(age_group['Age'].first(), age_group['Income'].first())
plt.title("Income Frequency by Age")
plt.ylabel("Frequency")
plt.xlabel("Age")
```

[22]: Text(0.5, 0, 'Age')



[23]:
```python
bank_loan_dt.head(3)
```

[23]:
```
     Income  Age  Experience Married/Single House_Ownership Car_Ownership  \
0  1303834   23           3         single           rented            no
1  7574516   40          10         single           rented            no
2  3991815   66           4        married           rented            no

            Profession       CITY           STATE  CURRENT_JOB_YRS  \
0  Mechanical_engineer       Rewa  Madhya_Pradesh                3
1   Software_Developer   Parbhani     Maharashtra                9
2     Technical_writer  Alappuzha          Kerala                4

   CURRENT_HOUSE_YRS  Risk_Flag
0                 13          0
1                 13          0
2                 10          0
```

[24]:
```python
#Identifying risk by state
state_risk_count = bank_loan_dt.groupby(['STATE']).sum()
state_risk_count
```

[24]:
```
                          Income      Age  Experience  CURRENT_JOB_YRS  \
STATE
```

Andhra_Pradesh	128774135101	1258351	254191	157165
Assam	34191471977	363308	69798	43373
Bihar	100011519383	984349	202738	128559
Chandigarh	2770147655	32881	7200	3789
Chhattisgarh	19374650965	189092	38262	23339
Delhi	27715045558	278091	53789	34934
Gujarat	57022857402	572202	117116	73267
Haryana	38497627381	398957	82105	49341
Himachal_Pradesh	3530005560	38338	9596	5524
Jammu_and_Kashmir	8103396262	93279	17504	10054
Jharkhand	45694257276	450482	90256	55219
Karnataka	59182874802	597219	114296	72165
Kerala	29966404885	294204	49499	34272
Madhya_Pradesh	69037948483	700156	148894	92872
Maharashtra	128176832360	1282109	255773	163533
Manipur	5060276687	41418	8070	5166
Mizoram	4290428951	38176	7429	4797
Odisha	23030587907	229298	52459	30087
Puducherry	6083282667	71062	12635	8377
Punjab	22844034394	227629	51450	31644
Rajasthan	47631372378	459631	93607	60161
Sikkim	2794414893	27148	5900	3795
Tamil_Nadu	83418951419	828580	171876	108390
Telangana	37181838612	356291	72135	46969
Tripura	4021181844	40691	7998	5763
Uttar_Pradesh	138728179845	1435202	288993	180259
Uttar_Pradesh[5]	3230631936	34662	7103	4415
Uttarakhand	8786402954	89382	17729	11842
West_Bengal	120122640125	1176238	232877	147066

	CURRENT_HOUSE_YRS	Risk_Flag
STATE		
Andhra_Pradesh	302053	2935
Assam	84306	930
Bihar	237338	2583
Chandigarh	8133	61
Chhattisgarh	46446	511
Delhi	65612	574
Gujarat	136820	1343
Haryana	96069	980
Himachal_Pradesh	10306	111
Jammu_and_Kashmir	21337	283
Jharkhand	107576	1195
Karnataka	141296	1189
Kerala	70054	970
Madhya_Pradesh	169622	2180
Maharashtra	305097	2895

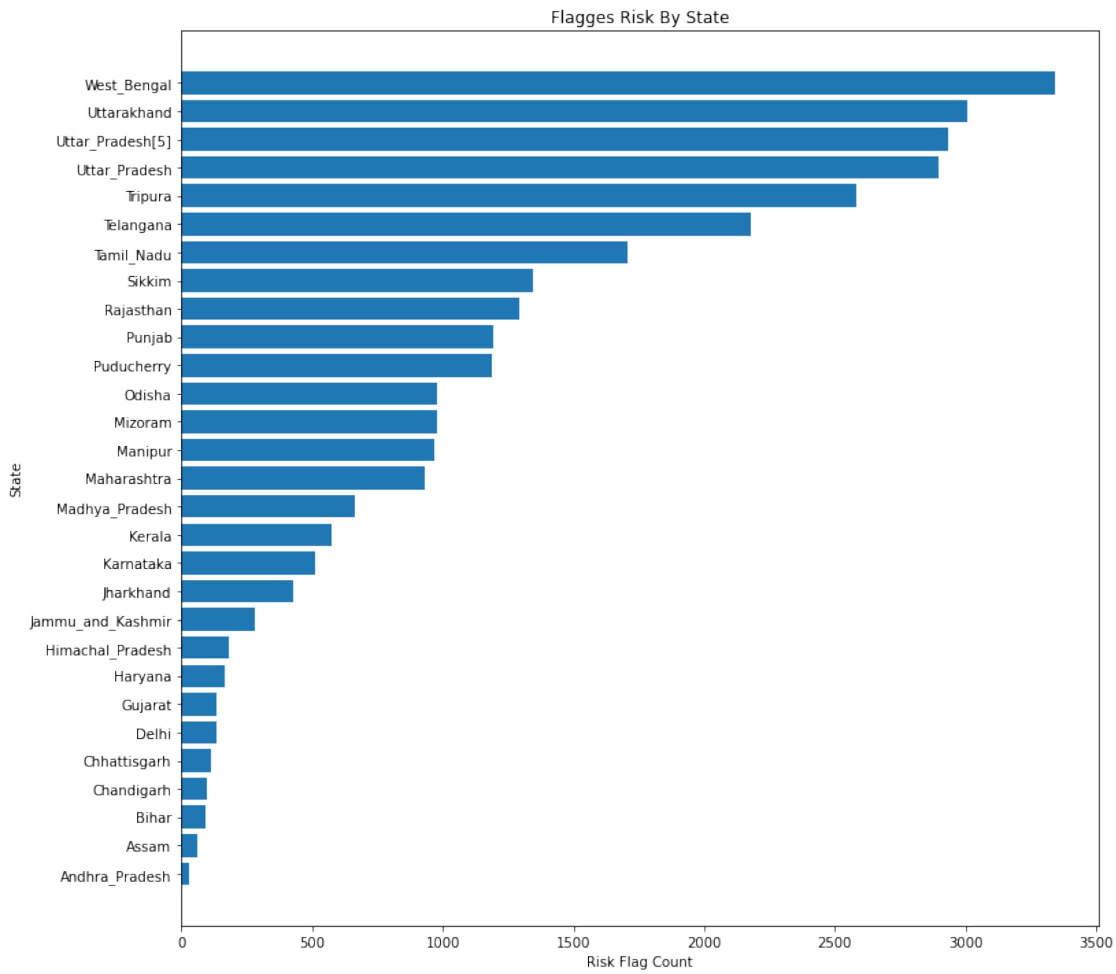
```
Manipur                      10289        183
Mizoram                       9802         94
Odisha                       55955        664
Puducherry                   16763        167
Punjab                       57400        425
Rajasthan                   111145       1292
Sikkim                        7469         28
Tamil_Nadu                  198209       1706
Telangana                    89522        979
Tripura                       9706        136
Uttar_Pradesh               342534       3343
Uttar_Pradesh[5]              8956         97
Uttarakhand                  22416        133
West_Bengal                 281213       3009
```

[25]:

```python
plt.figure(figsize=(12,12))
plt.title("Risk By State")
plt.ylabel("State")
plt.xlabel("Risk Flag Count")
#sorting based off the number of risk
plt.barh( state_risk_count.index,sorted(state_risk_count['Risk_Flag']))
```

[25]:

```
<BarContainer object of 29 artists>
```



[26]:
```
bank_loan_dt.head(1)
```

[26]:
```
    Income  Age  Experience Married/Single House_Ownership Car_Ownership  \
0  1303834   23           3         single          rented            no

          Profession  CITY          STATE  CURRENT_JOB_YRS  \
0  Mechanical_engineer  Rewa  Madhya_Pradesh                3

   CURRENT_HOUSE_YRS  Risk_Flag
0                 13          0
```

5 Adding Classification Before Model

[7]:
```
#young adults(ages 18-35 years; n = 97),
#middle-aged adults (ages 36-55 years, n = 197),
#and older adults (aged older than 55 years, n = 49).
#https://pubmed.ncbi.nlm.nih.gov/11815703/

#Taking the Age value and returning the group it stands in
# the age group is taken from the government article, to define age groups

def age_classification(age):
    #initializing value to none
    age_value = None
    #taking age and returning value
    #using artilce to base classification age
    if(age >= 18 and age <=35):
        age_value = 'young_adults'
    elif(age>=36 and age<=55):
        age_value = 'middle_aged_adults'
    elif(age> 55):
        age_value = 'older_adult'
    else:
        age_value = None
    return age_value
```

[8]:
```
#https://www.investopedia.com/financial-edge/0912/which-income-class-are-you.
↪aspx
#$25,471.00        $84,372.00        $187,094.00 example for maryland

#seems like salary average are close to the united states
#there is a difference seems like high salaries accounting for that in placement
#assumption is that salaries are similar so this will serve as a base line
def salary_classification(salary):
    #initializing value to none
    salary_value = None
    #taking salary and returning value
    #using artilce to base classification salary
    if( salary <=50000):
        salary_value = 'low_income'
    elif(salary >50000 and salary<=120000):
        salary_value = 'middle_income'
    elif(salary >120000):
        salary_value= 'high_income'
    else:
        salary_value = None
```

```
    return salary_value
```

[9]:
```
#To prove this assumption, I am taking the min , median, and max

#seeing if other other values need to have a classification add to it.
#for example can we establish what a low experience at work would be? And does␣
↪it matter.

min_yr_job = bank_loan_dt['CURRENT_JOB_YRS'].min()
mid_yr_job = bank_loan_dt['CURRENT_JOB_YRS'].median()
max_yr_job = bank_loan_dt['CURRENT_JOB_YRS'].max()


min_yr_house = bank_loan_dt['CURRENT_HOUSE_YRS'].min()
mid_yr_house = bank_loan_dt['CURRENT_HOUSE_YRS'].median()
max_yr_house = bank_loan_dt['CURRENT_HOUSE_YRS'].max()


min_ex_job = bank_loan_dt['Experience'].min()
mid_ex_job = bank_loan_dt['Experience'].median()
max_ex_job = bank_loan_dt['Experience'].max()




#At this point i will not use these as a classification and keep these as is.
print("Years in job : Lowest Score {}\nMiddle Score {}\nHighest score {} ".
↪format(min_yr_job,mid_yr_job, max_yr_job), '\n')
print("Years in house : Lowest Score {}\nMiddle Score {}\nHighest score {} ".
↪format(min_yr_house,mid_yr_house, max_yr_house), '\n')
print("Years in job : Lowest Score {}\nMiddle Score {}\nHighest score {} ".
↪format(min_ex_job,mid_ex_job, max_ex_job))
```

```
Years in job : Lowest Score 0
Middle Score 6.0
Highest score 14

Years in house : Lowest Score 10
Middle Score 12.0
Highest score 14

Years in job : Lowest Score 0
Middle Score 10.0
Highest score 20
```

[10]:
```
#New column Age_Group
bank_loan_dt['Age_Group'] = bank_loan_dt['Age'].apply(age_classification)
```

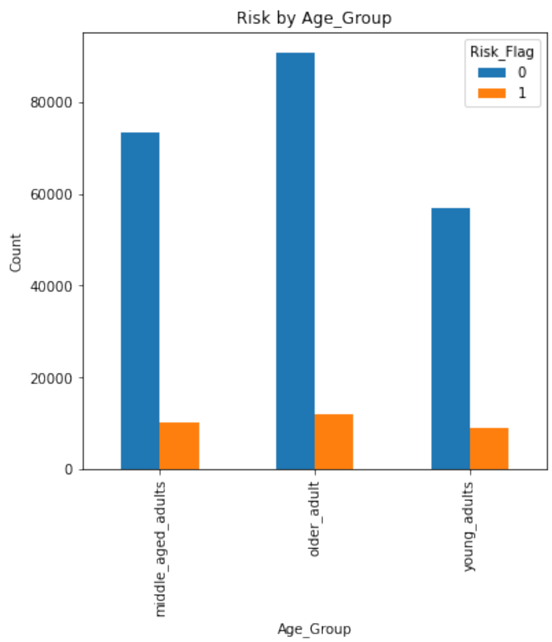
```
bank_loan_dt['Income_Group'] = bank_loan_dt['Income'].
 ↪apply(salary_classification)
```

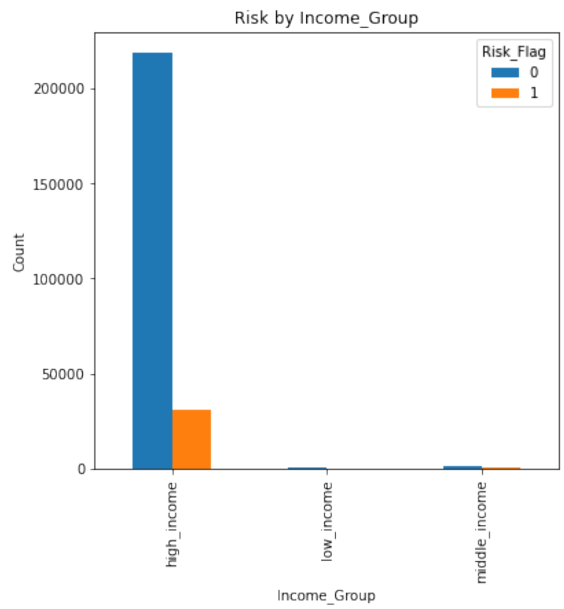
[12]:
```
#adding classifiers to data frame
bank_new_classifiers = bank_loan_dt[['Age_Group','Income_Group']]
```

[13]:
```
bank_new_classifiers.head(3)
```

[13]:
```
             Age_Group Income_Group
0         young_adults  high_income
1  middle_aged_adults  high_income
2          older_adult  high_income
```

[14]:
```
#Setting figure size
fig = plt.figure(figsize=(12,12))
#initializing count to 1
count = 1
#taking each column in bank_new_classifiers
#Should consist of 2
for i in bank_new_classifiers.columns.values:
    #2 plots will show the subplot will be set to 2 and 2
    ax = plt.subplot(2,2,count)

    #for each bank column plot by risk
    pd.crosstab(bank_new_classifiers[i],bank_loan_dt['Risk_Flag']).
 ↪plot(kind='bar',ax=ax)
    plt.tight_layout(pad=3.0)

    #Setting title to instance of first columns
    plt.title("Risk by "+ i)
    plt.xlabel(i)
    plt.ylabel("Count")

    #counter increment for new plot
    count = count + 1
```





6 Data Perpertation

[15]:
```
#splitting exited as main target, this shows if person has left the bank
#splitting all other columns to features

#loading data from trainning set
target = bank_loan_dt[['Risk_Flag']]

#encoding data set
features = pd.get_dummies(bank_loan_dt.drop(['Risk_Flag'],  axis = 1))
```

[16]:
```
target.shape
```

[16]: (252000, 1)

[17]:
```
features.shape
```

[17]: (252000, 416)

[18]:
```
#standarized data
scaler = StandardScaler()
```

[19]:
```
featured_scaled = scaler.fit_transform(features)
```

[20]:
```
#instantiating PCA
#setting to 50 components rather then 416
pca = PCA(n_components= 50)

#Setting to a lower amount of components
features_pca = pca.fit_transform(features)
```

[21]:
```
#Splitting data into training and test
features_train, features_test, target_train, target_test =␣
↪train_test_split(features_pca, target, test_size=.20)
```

7 Main Model Random Forest

[22]:
```
#instantiating class
RandForClass = RandomForestClassifier()
```

[23]:
```
model = RandForClass.fit(features_train, np.ravel(target_train))
```

[24]:
```
predicted = model.predict(features_test)
```

[25]:
```
accuracy_score(target_test,predicted)
```

[25]: 0.9057142857142857

[26]:
```
recall_score(target_test,predicted)
```

[26]: 0.4813023855577047

[27]:
```
precision_score(target_test,predicted)
```

[27]: 0.6606194690265487

[28]:
```
f1_score(target_test,predicted)
```

[28]: 0.556881760537113

8 Test Logestic Regression Model Against Random Forest

[29]:
```
LogReg = LogisticRegression()
```

[30]:
```
l_model = LogReg.fit(features_train, np.ravel(target_train))
```

[33]:
```
l_predicted = l_model.predict(features_test)
```

[34]:
```
accuracy_score(target_test,l_predicted)
```

[34]: 0.5058531746031746

[35]:
```
recall_score(target_test,l_predicted)
```

[35]: 0.5261121856866537

[36]:
```
precision_score(target_test,l_predicted)
```

[36]: 0.1293749256807642

[37]:
```
f1_score(target_test,l_predicted)
```

[37]: 0.20767982693347758

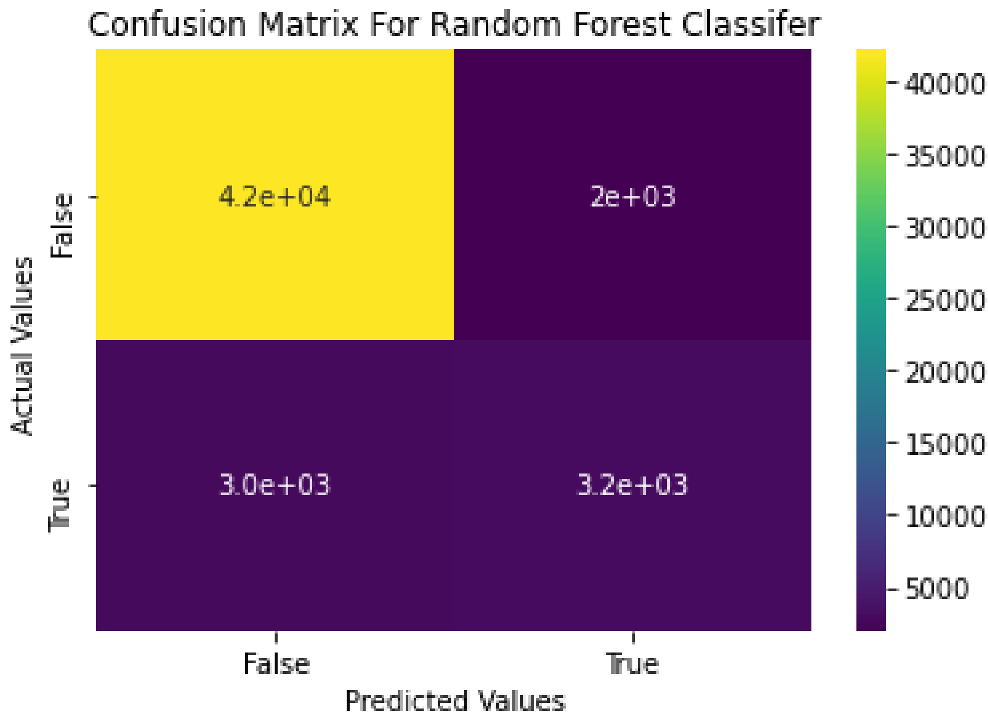
9 Cross Validation

[53]:
```
#random forest is the better model all around.
#it has a higher recall and precision score
#cross validating
results = confusion_matrix(target_test,predicted)
results
```

[53]:
```
array([[42090,  2012],
       [ 3050,  3248]], dtype=int64)
```

[61]:
```
#setting up heat map
cr = sns.heatmap(results, annot= True, cmap='viridis')
cr.set_title('Confusion Matrix For Random Forest Classifer')
cr.set_xlabel('Predicted Values')
cr.set_ylabel('Actual Values')


cr.xaxis.set_ticklabels(['False', 'True'])
cr.yaxis.set_ticklabels(['False', 'True'])
```

[61]: [Text(0, 0.5, 'False'), Text(0, 1.5, 'True')]



10 ROC Curve

[38]:
```
#Setting a random line for our graph prediction
#Setting as a base line to for other model comparison
r_probs = [0 for i in range(len(target_test))]

#Setting Logistic Regression

l_probs = LogReg.predict_proba(features_test)

#Setting random forest by predicting probabilty that the target will be 0 or 1
rf_probs = RandForClass.predict_proba(features_test)
```

[39]:
```
# Getting Postive for random forest and logistic regression
rf_probs = rf_probs[:,1]
l_probs = l_probs[:,1]
```

[40]:
```
#Getting the FPR,TPR and threshold values for each model

#Random prediction line will yeild a straight line.
r_fpr , r_tpr , _ = roc_curve(target_test,r_probs)
```

```
#Random forest FRP , TRP
rf_fpr , rf_tpr , _ = roc_curve(target_test,rf_probs)


#Logistic Regression
l_fpr, l_tpr , _ = roc_curve(target_test,l_probs)
```

[41]:
```
#plotting roc curve
plt.plot(r_fpr,r_tpr, label= "Random Predictioin")
plt.plot(rf_fpr , rf_tpr, label="Random Forrest")
plt.plot(l_fpr , l_tpr, label="Logistic Regression")


plt.title("Roc Plot")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
```

[41]: <matplotlib.legend.Legend at 0x266dfe2bc10>