Team2-FinalProject

June 15, 2023

[1]:
```
import pandas as pd
import io
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt #to allow subplot creation
%matplotlib inline

import plotly.graph_objs as go
from plotly.offline import iplot
import plotly.express as px

import plotly.figure_factory as ff
import plotly.graph_objects as go
from tabulate import tabulate

from sklearn import tree # Import Tree Classifiers
from sklearn.ensemble import RandomForestClassifier # Import Random Forest␣
 ↪Classifiers
from sklearn.model_selection import train_test_split,  RandomizedSearchCV #␣
 ↪Import train_test_split function
from sklearn import metrics # Import scikit-learn metrics module for accuracy␣
 ↪calculation
from sklearn.preprocessing import LabelEncoder
from scipy.stats import randint # Generate random numbers
```

[2]:
```
import pandas as pd
from dataprep.eda import *
from dataprep.datasets import load_dataset
from dataprep.eda import plot, plot_correlation, plot_missing, plot_diff,␣
 ↪create_report #
```

[3]:
```
import opendatasets as od
od.download("https://www.kaggle.com/datasets/rikdifos/
 ↪credit-card-approval-prediction?select=application_record.csv")
```

```
Skipping, found downloaded files in ".\credit-card-approval-prediction" (use
force=True to force download)
```

[4]:
```
od.download("https://www.kaggle.com/datasets/rikdifos/
↪credit-card-approval-prediction?select=credit_record.csv")
```

```
Skipping, found downloaded files in ".\credit-card-approval-prediction" (use
force=True to force download)
```

[5]:
```
df1 = pd.read_csv("credit-card-approval-prediction/application_record.csv")
df2 = pd.read_csv("credit-card-approval-prediction/credit_record.csv")
```

[6]:
```
df1.head()
```

[6]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	-21474	-1134	1	0	0	0	Security staff	2.0
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0

[7]:
```
df2.head(10)
```

[7]:

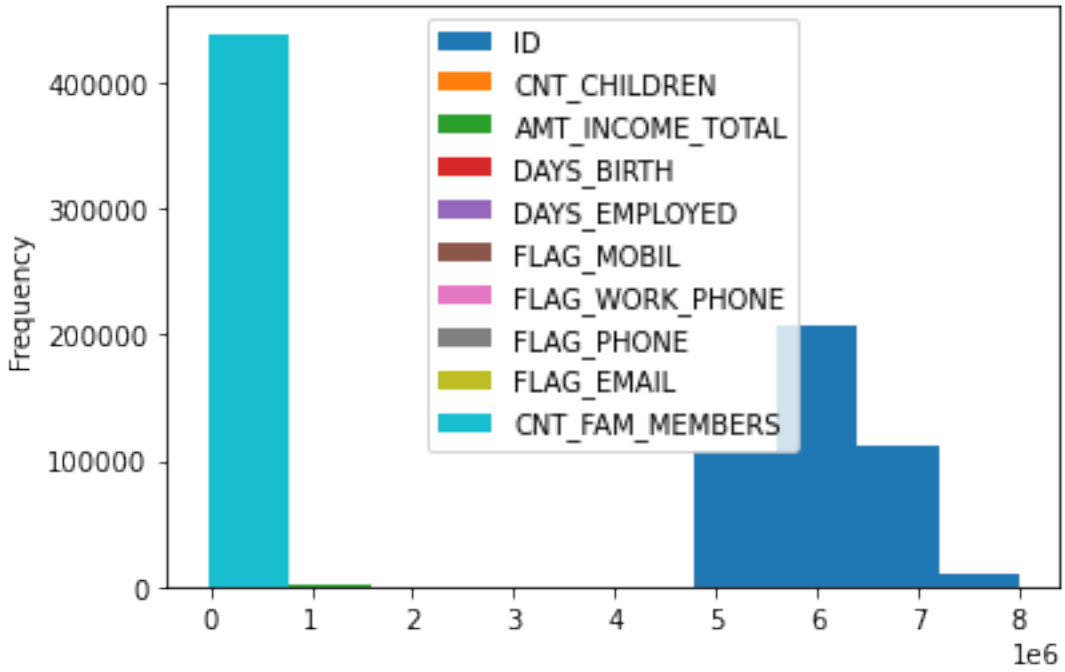
	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C
5	5001712	-1	C
6	5001712	-2	C
7	5001712	-3	C
8	5001712	-4	C
9	5001712	-5	C

[8]:
```
df1.count()
```

[8]:
```
ID                   438557
CODE_GENDER          438557
FLAG_OWN_CAR         438557
FLAG_OWN_REALTY      438557
CNT_CHILDREN         438557
AMT_INCOME_TOTAL     438557
NAME_INCOME_TYPE     438557
NAME_EDUCATION_TYPE  438557
NAME_FAMILY_STATUS   438557
NAME_HOUSING_TYPE    438557
DAYS_BIRTH           438557
DAYS_EMPLOYED        438557
FLAG_MOBIL           438557
FLAG_WORK_PHONE      438557
FLAG_PHONE           438557
FLAG_EMAIL           438557
OCCUPATION_TYPE      304354
CNT_FAM_MEMBERS      438557
dtype: int64
```

[9]:
```
df1.plot.hist()
```

[9]:
```
<AxesSubplot:ylabel='Frequency'>
```



[10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   438557 non-null  int64
 1   CODE_GENDER          438557 non-null  object
 2   FLAG_OWN_CAR         438557 non-null  object
 3   FLAG_OWN_REALTY      438557 non-null  object
 4   CNT_CHILDREN         438557 non-null  int64
 5   AMT_INCOME_TOTAL     438557 non-null  float64
 6   NAME_INCOME_TYPE     438557 non-null  object
 7   NAME_EDUCATION_TYPE  438557 non-null  object
 8   NAME_FAMILY_STATUS   438557 non-null  object
 9   NAME_HOUSING_TYPE    438557 non-null  object
 10  DAYS_BIRTH           438557 non-null  int64
 11  DAYS_EMPLOYED        438557 non-null  int64
 12  FLAG_MOBIL           438557 non-null  int64
 13  FLAG_WORK_PHONE      438557 non-null  int64
 14  FLAG_PHONE           438557 non-null  int64
 15  FLAG_EMAIL           438557 non-null  int64
 16  OCCUPATION_TYPE      304354 non-null  object
 17  CNT_FAM_MEMBERS      438557 non-null  float64
```

```
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

[11]:
```
null_count = df1.isnull().sum().reset_index(name = "null count")
unique_valuecount = df1.nunique().reset_index(name = "unique value count")
datatypes = df1.dtypes.reset_index(name="types")
pd.set_option('display.max_rows', 500)
pd.concat([null_count, unique_valuecount , datatypes], axis=1).T.
  ↪drop_duplicates().T
```

[11]:

	index	null count	unique value count	types
0	ID	0	438510	int64
1	CODE_GENDER	0	2	object
2	FLAG_OWN_CAR	0	2	object
3	FLAG_OWN_REALTY	0	2	object
4	CNT_CHILDREN	0	12	int64
5	AMT_INCOME_TOTAL	0	866	float64
6	NAME_INCOME_TYPE	0	5	object
7	NAME_EDUCATION_TYPE	0	5	object
8	NAME_FAMILY_STATUS	0	5	object
9	NAME_HOUSING_TYPE	0	6	object
10	DAYS_BIRTH	0	16379	int64
11	DAYS_EMPLOYED	0	9406	int64
12	FLAG_MOBIL	0	1	int64
13	FLAG_WORK_PHONE	0	2	int64
14	FLAG_PHONE	0	2	int64
15	FLAG_EMAIL	0	2	int64
16	OCCUPATION_TYPE	134203	18	object
17	CNT_FAM_MEMBERS	0	13	float64

[12]:
```
plt.rcParams['figure.facecolor'] = 'white'
```

[13]:
```
apprecprocess=df1.copy()
credrecprocess=df2.copy()
```

[14]:
```
apprecprocess['ID'].nunique()
```

[14]:
```
438510
```

the total rows are 438,557. This means it has duplicates

[15]:
```
apprecprocess = apprecprocess.drop_duplicates('ID', keep='last')
# we identified that there are some duplicates in this dataset
# we will be deleting those duplicates and will keep the last entry of the ID␣
  ↪if its repeated.
```

[16]:
```
credrecprocess.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   ID              1048575 non-null  int64
 1   MONTHS_BALANCE  1048575 non-null  int64
 2   STATUS          1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

[17]:
```
credrecprocess['ID'].nunique()
```

[17]:
```
45985
```

this has around 46,000 unique rows as there are repeating entries for different monthly values and status.

[18]:
```
# checking to see how many records match in two datasets
len(set(credrecprocess['ID']).intersection(set(apprecprocess['ID'])))
```

[18]:
```
36457
```

[19]:
```
# find all users' account open month.
begin_month=pd.DataFrame(credrecprocess.groupby(["ID"])["MONTHS_BALANCE"].
 ↪agg(min))
begin_month=begin_month.rename(columns={'MONTHS_BALANCE':'begin_month'})
appcredmergedata=pd.merge(apprecprocess,begin_month,how="left",on="ID") #merge␣
 ↪to record data
appcredmergedata.head(3)
```

[19]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	\
0	5008804	M	Y	Y	0	
1	5008805	M	Y	Y	0	
2	5008806	M	Y	Y	0	

	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	\
0	427500.0	Working	Higher education	
1	427500.0	Working	Higher education	
2	112500.0	Working	Secondary / secondary special	

	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	\
0	Civil marriage	Rented apartment	-12005	-4542	
1	Civil marriage	Rented apartment	-12005	-4542	
2	Married	House / apartment	-21474	-1134	

	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	\
0	1	1	0	0	NaN	
1	1	1	0	0	NaN	

```
2           1                0           0           0  Security staff

   CNT_FAM_MEMBERS  begin_month
0              2.0        -15.0
1              2.0        -14.0
2              2.0        -29.0
```

[20]:
```
credrecprocess['STATUS'].nunique()
```

[20]:
```
8
```

As per the data dictionary, following values are expected in status columns : 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month

[21]:
```
#Creating a new column and considering all candidates with overdue as more than␣
 ↪90 days as possible risk
credrecprocess['targetrisk'] = None
credrecprocess['targetrisk'][credrecprocess['STATUS'] =='3']='Yes'
credrecprocess['targetrisk'][credrecprocess['STATUS'] =='4']='Yes'
credrecprocess['targetrisk'][credrecprocess['STATUS'] =='5']='Yes'
credrecprocess.head()
```

[21]:
```
         ID  MONTHS_BALANCE STATUS targetrisk
0  5001711               0      X       None
1  5001711              -1      0       None
2  5001711              -2      0       None
3  5001711              -3      0       None
4  5001712               0      C       None
```

[22]:
```
credrecprocess1=credrecprocess.groupby('ID').count()
credrecprocess1['targetrisk'][credrecprocess1['targetrisk'] > 0]='Yes'
credrecprocess1['targetrisk'][credrecprocess1['targetrisk'] == 0]='No'
credrecprocess1 = credrecprocess1[['targetrisk']]
credrecprocess1.head(3)
```

[22]:
```
         targetrisk
ID
5001711         No
5001712         No
5001713         No
```

[23]:
```
# Merge  status with the main record dataframe where targetrisk = yes will get␣
 ↪converted to 1 and No as 0 to avoid label encoding in future.
appcredmergedata=pd.merge(appcredmergedata,credrecprocess1,how='inner',on='ID')
appcredmergedata['targetrisk']=appcredmergedata['targetrisk']
appcredmergedata.loc[appcredmergedata['targetrisk']=='Yes','targetrisk']=1
```

```
appcredmergedata.loc[appcredmergedata['targetrisk']=='No','targetrisk']=0
```

[24]:
```
appcredmergedata.head()
```

[24]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS	begin_month	targetrisk
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-15.0	0
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-14.0	0
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	-21474	-1134	1	0	0	0	Security staff	2.0	-29.0	0
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0	-4.0	0
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0	-26.0	0

[25]:
```
print(appcredmergedata['targetrisk'].value_counts())
appcredmergedata['targetrisk'].value_counts(normalize=True)
```

```
0    36155
1      302
Name: targetrisk, dtype: int64
```

[25]:
```
0    0.991716
1    0.008284
```

```
Name: targetrisk, dtype: float64
```

only records are at risk , which is low though it is also important to understand the factors affecting it and this can be associated with high numbers in terms of amount as well

Data Report

0.1 Feature Engineering

[26]:
```
dfprocess=appcredmergedata.copy()
```

[27]:
```
dfprocess.rename(columns={'CODE_GENDER':'Gender','FLAG_OWN_CAR':
 ↪'OwnCar','FLAG_OWN_REALTY':'OwnReality',
                          'CNT_CHILDREN':'ChildCount','AMT_INCOME_TOTAL':
 ↪'AnnualIncome',
                          'NAME_EDUCATION_TYPE':
 ↪'EducationType','NAME_FAMILY_STATUS':'FamilyStatus',
                         'NAME_HOUSING_TYPE':'HousingType','FLAG_EMAIL':
 ↪'EmailFlag',
                          'NAME_INCOME_TYPE':'IncomeType','FLAG_WORK_PHONE':
 ↪'WorkPhoneFlag',
                          'FLAG_PHONE':'PhoneFlag','CNT_FAM_MEMBERS':
 ↪'FamilySize',
                         'OCCUPATION_TYPE':'OccupationType',
                           'DAYS_BIRTH':'DaysBirth' , 'DAYS_EMPLOYED':
 ↪'DaysEmployed' , 'FLAG_MOBIL': 'MobileFlag'
                         },inplace=True)
```

[28]:
```
dfprocess.head(2)
```

[28]:

	ID	Gender	OwnCar	OwnReality	ChildCount	AnnualIncome	IncomeType	EducationType	FamilyStatus	HousingType	DaysBirth	DaysEmployed	MobileFlag	WorkPhoneFlag	PhoneFlag	EmailFlag	OccupationType	FamilySize	begin_month	targetrisk
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-15.0	0
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-14.0	0

Gender

[29]:
```
dfprocess['Gender'].unique()
```

[29]:
```
array(['M', 'F'], dtype=object)
```

Here Gender has only 2 values hence easy to convert this to numeric value for model training and also, we will be converting this to type integer

[30]:
```
dfprocess['Gender'] = dfprocess['Gender'].replace(['F','M'],[0,1])
```

[31]:
```
dfprocess['Gender']=dfprocess['Gender'].astype('int64')
```

Having a car or not

[32]:
```
dfprocess['OwnCar'].unique()
```

[32]:
```
array(['Y', 'N'], dtype=object)
```

[33]:
```
dfprocess['OwnCar'] = dfprocess['OwnCar'].replace(['N','Y'],[0,1])
dfprocess['OwnCar']=dfprocess['OwnCar'].astype('int64')
```

Here we have encoded N and Y as 0 and 1 respectively and converted the column into Integer type

Similar to owncar column, OwnReality column will be updated

[34]:
```
dfprocess['OwnReality'] = dfprocess['OwnReality'].replace(['N','Y'],[0,1])
dfprocess['OwnReality']=dfprocess['OwnReality'].astype('int64')
```

[35]:
```
print(dfprocess['PhoneFlag'].value_counts())
dfprocess['PhoneFlag'].value_counts(normalize=True)
```

```
0    25709
1    10748
Name: PhoneFlag, dtype: int64
```

[35]:
```
0    0.705187
1    0.294813
Name: PhoneFlag, dtype: float64
```

[36]:
```
print(dfprocess['EmailFlag'].value_counts())
dfprocess['EmailFlag'].value_counts(normalize=True)
```

```
0    33186
1     3271
Name: EmailFlag, dtype: int64
```

[36]:
```
0    0.910278
1    0.089722
Name: EmailFlag, dtype: float64
```

[37]:
```
print(dfprocess['WorkPhoneFlag'].value_counts())
dfprocess['WorkPhoneFlag'].value_counts(normalize=True)
```

```
0    28235
1     8222
Name: WorkPhoneFlag, dtype: int64
```

[37]:
```
0    0.774474
1    0.225526
Name: WorkPhoneFlag, dtype: float64
```

[38]:
```
print(dfprocess['MobileFlag'].value_counts())
dfprocess['MobileFlag'].value_counts(normalize=True)
```

```
1    36457
Name: MobileFlag, dtype: int64
```

[38]:
```
1    1.0
Name: MobileFlag, dtype: float64
```

0.1.1 Continuous Variables

[39]:
```
dfprocess['ChildCount'].value_counts(normalize=True)
```

[39]:
```
0     0.691253
1     0.205502
2     0.089311
3     0.011493
4     0.001728
5     0.000549
14    0.000082
7     0.000055
19    0.000027
Name: ChildCount, dtype: float64
```

Here the percentage of child count more than 2 is pretty low hence we will club them together

[40]:
```
dfprocess.loc[dfprocess['ChildCount'] >= 2,'ChildCount']='2More'
dfprocess['ChildCount'].value_counts()
```

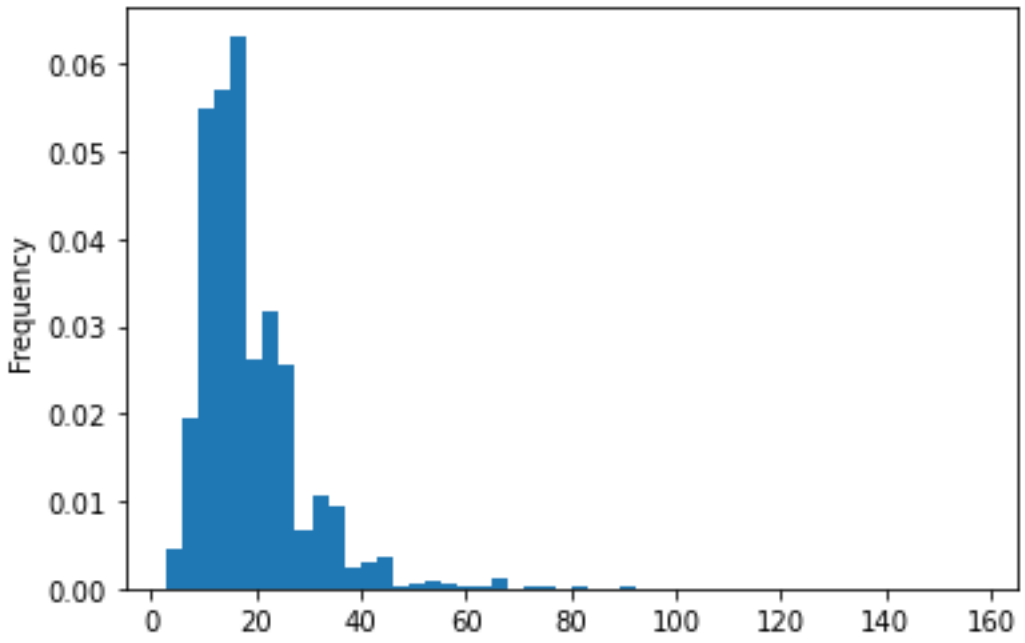
[40]:
```
0        25201
1         7492
2More     3764
Name: ChildCount, dtype: int64
```

[41]:
```
# Categorizing annual incode in bins
#dfprocess['AnnualIncome']=dfprocess['AnnualIncome'].astype(object)
dfprocess['AnnualIncome'] = dfprocess['AnnualIncome']/10000
print(dfprocess['AnnualIncome'].value_counts(bins=10,sort=False))
dfprocess['AnnualIncome'].plot(kind='hist',bins=50,density=True)
```

```
(2.544, 18.18]      22460
(18.18, 33.66]      11380
```

```
(33.66, 49.14]       2099
(49.14, 64.62]        274
(64.62, 80.1]         165
(80.1, 95.58]          58
(95.58, 111.06]         4
(111.06, 126.54]        3
(126.54, 142.02]        6
(142.02, 157.5]         8
Name: AnnualIncome, dtype: int64
```

[41]: <AxesSubplot:ylabel='Frequency'>



[42]:
```
dfprocess1=dfprocess.copy()
```

Reference: https://stackoverflow.com/questions/30211923/what-is-the-difference-between-pandas-qcut-and-pandas-cut

[43]:
```
def createbins(dataframe, column_name, label_texts, option):
    # Create a new column to store the bin labels
    bin_column_name = column_name + '_bin'

    if option == 'quantile':
        dataframe[bin_column_name] = pd.qcut(dataframe[column_name],␣
↪q=len(label_texts), labels=label_texts)
    elif option == 'equal-length':
```

```
        dataframe[bin_column_name] = pd.cut(dataframe[column_name],␣
↪bins=len(label_texts), labels=label_texts)
    else:
        raise ValueError("Invalid option. Please choose either 'quantile' or␣
↪'equal-length'.")

    return dataframe
```

[44]:
```
#option = 'quantile'  # or 'equal-length'
dfprocess1 = createbins(dfprocess1,'AnnualIncome', ["low","medium", "high"],␣
↪'quantile')
```

[45]:
```
dfprocess1['AnnualIncome_bin'].value_counts()
```

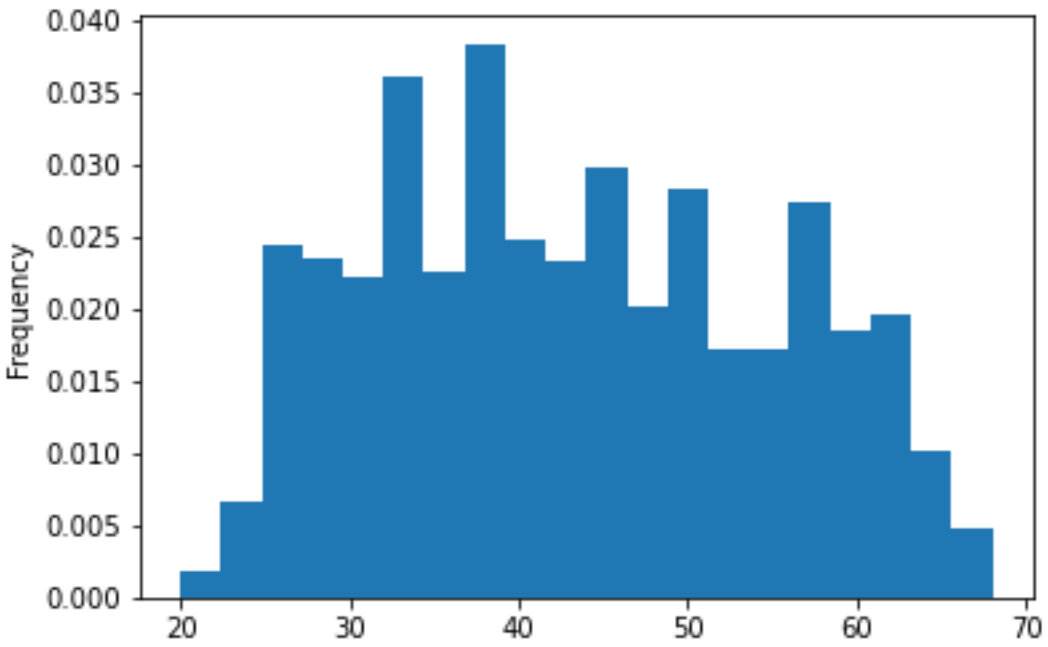
[45]:
```
low       14473
high      11282
medium    10702
Name: AnnualIncome_bin, dtype: int64
```

Days_Birth / Age

[46]:
```
dfprocess1['Age']=-(dfprocess1['DaysBirth'])//365
print(dfprocess1['Age'].value_counts(bins=10,normalize=True,sort=False))
dfprocess1['Age'].plot(kind='hist',bins=20,density=True)
```

```
(19.951, 24.8]    0.020243
(24.8, 29.6]      0.114930
(29.6, 34.4]      0.139836
(34.4, 39.2]      0.146419
(39.2, 44.0]      0.140796
(44.0, 48.8]      0.094166
(48.8, 53.6]      0.109444
(53.6, 58.4]      0.106948
(58.4, 63.2]      0.091286
(63.2, 68.0]      0.035933
Name: Age, dtype: float64
```

[46]:
```
<AxesSubplot:ylabel='Frequency'>
```



[47]:
```
dfprocess2=dfprocess1.copy()
```

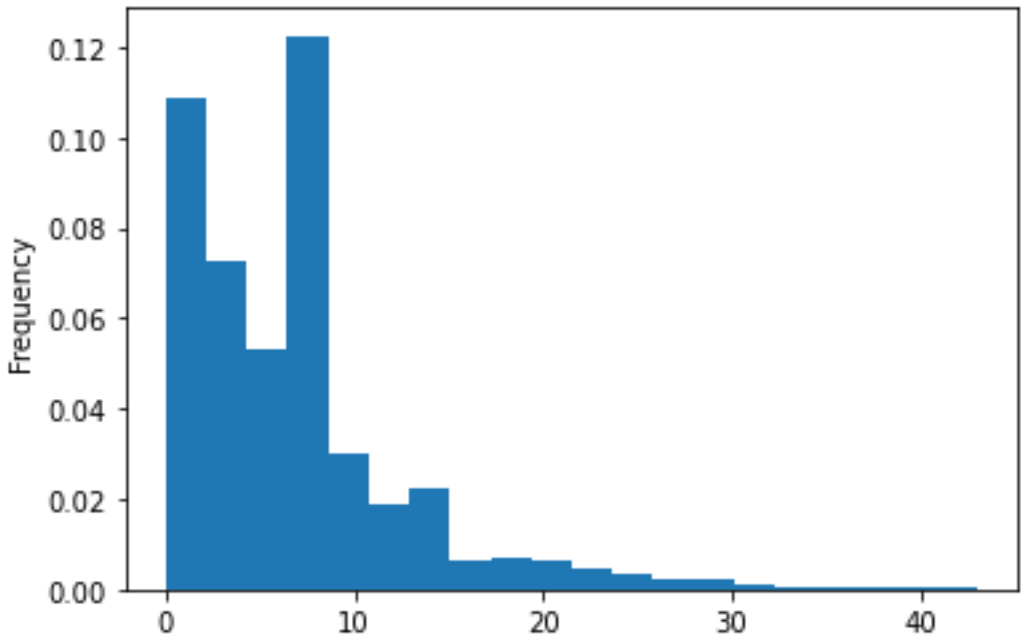
[48]:
```
#option = 'quantile'  # or 'equal-length'
dfprocess2 = createbins(dfprocess2,'Age', ["0-20","20-40", "40-60","60-80"],␣
 ↪'equal-length')
```

[49]:
```
dfprocess2['Age_bin'].value_counts()
```

[49]:
```
20-40    12582
40-60     9737
0-20      7915
60-80     6223
Name: Age_bin, dtype: int64
```

[50]:
```
dfprocess2['YearsEmployed']=-(dfprocess2['DaysEmployed'])//365
dfprocess2[dfprocess2['YearsEmployed']<0] = np.nan # replace by na
dfprocess2['YearsEmployed'].fillna(dfprocess2['YearsEmployed'].
 ↪mean(),inplace=True) #replace na by mean
dfprocess2['YearsEmployed'].plot(kind='hist',bins=20,density=True)
```

[50]:
```
<AxesSubplot:ylabel='Frequency'>
```



[51]:
```
#option = 'quantile'  # or 'equal-length'
dfprocess2 = createbins(dfprocess2,'YearsEmployed', ["0-15","15-30",␣
 ↪"30-45","45-60"], 'equal-length')
```

[52]:
```
dfprocess2['YearsEmployed_bin'].value_counts()
```

[52]:
```
0-15     30419
15-30     4775
30-45     1087
45-60      176
Name: YearsEmployed_bin, dtype: int64
```

Family Size

[53]:
```
dfprocess2['FamilySize'].value_counts(sort=False)
```

[53]:
```
2.0     15389
1.0      5151
5.0       387
3.0      6255
4.0      3057
6.0        58
15.0        3
7.0        19
20.0        1
9.0         2
```

```
Name: FamilySize, dtype: int64
```

[54]:
```
dfprocess2['FamilySizeGroup']=dfprocess2['FamilySize']
dfprocess2.loc[dfprocess2['FamilySizeGroup']>=3,'FamilySizeGroup']='3more'
dfprocess2['FamilySizeGroup'].value_counts(sort=False)
```

[54]:
```
2.0      15389
1.0       5151
3more     9782
Name: FamilySizeGroup, dtype: int64
```

[55]:
```
dfprocess2['FamilySizeGroup']=dfprocess2['FamilySizeGroup'].astype(object)
```

0.1.2 Categorial Data

[56]:
```
dfprocess3=dfprocess2.copy()
```

[57]:
```
dfprocess3['IncomeType'].value_counts(sort=False)
```

[57]:
```
Working                 18819
Commercial associate     8490
State servant            2985
Student                    11
Pensioner                  17
Name: IncomeType, dtype: int64
```

[58]:
```
dfprocess3.loc[dfprocess3['IncomeType']=='Pensioner','IncomeType']='State␣
↪servant'
dfprocess3.loc[dfprocess3['IncomeType']=='Student','IncomeType']='State servant'
dfprocess3['IncomeType'].value_counts(sort=False)
```

[58]:
```
Working                 18819
Commercial associate     8490
State servant            3013
Name: IncomeType, dtype: int64
```

[59]:
```
dfprocess3['OccupationType'].value_counts(sort=False)
```

[59]:
```
Security staff           592
Sales staff             3485
Accountants             1241
Laborers                6211
Managers                3012
Drivers                 2138
Core staff              3591
High skill tech staff   1383
Cleaning staff           551
Private service staff    344
```

```
Cooking staff             655
Low-skill Laborers        175
Medicine staff           1207
Secretaries               151
Waiters/barmen staff      174
HR staff                   85
Realty agents              79
IT staff                   60
Name: OccupationType, dtype: int64
```

there is a wide variation in this field , hence will keep it as it is for now. may group it at later point

[60]:
```
dfprocess3['HousingType'].value_counts(sort=False)
```

[60]:
```
Rented apartment         545
House / apartment      26653
Municipal apartment      951
With parents            1771
Co-op apartment          160
Office apartment         242
Name: HousingType, dtype: int64
```

[61]:
```
dfprocess3['EducationType'].value_counts(sort=False)
```

[61]:
```
Higher education                 8858
Secondary / secondary special   19867
Incomplete higher                1352
Lower secondary                   214
Academic degree                    31
Name: EducationType, dtype: int64
```

[62]:
```
dfprocess3['FamilyStatus'].value_counts(sort=False)
```

[62]:
```
Civil marriage          2575
Married                21137
Single / not married    4148
Separated               1758
Widow                    704
Name: FamilyStatus, dtype: int64
```

[63]:
```
ot = pd.DataFrame(dfprocess3.dtypes =='object').reset_index()
object_type = ot[ot[0] == True]['index']
object_type
```

[63]:
```
4          ChildCount
6          IncomeType
7       EducationType
8        FamilyStatus
```

```
9          HousingType
16      OccupationType
19          targetrisk
25     FamilySizeGroup
Name: index, dtype: object
```

[64]:
```
num_type = pd.DataFrame(dfprocess3.dtypes != 'object').reset_index().
 ↪rename(columns =  {0:'yes/no'})
num_type = num_type[num_type['yes/no'] ==True]['index']
num_type
```

[64]:
```
0                    ID
1                Gender
2                OwnCar
3            OwnReality
5          AnnualIncome
10            DaysBirth
11         DaysEmployed
12           MobileFlag
13        WorkPhoneFlag
14            PhoneFlag
15            EmailFlag
17           FamilySize
18          begin_month
20     AnnualIncome_bin
21                  Age
22              Age_bin
23        YearsEmployed
24    YearsEmployed_bin
Name: index, dtype: object
```

[65]:
```
dfvisual=dfprocess3.copy()
```

[66]:
```
# Create Dataprep EDA report
report = create_report(dfvisual)

# Show the report
report.show()
```

```
  0%|          | 0/3192 [00:00<?, ?it/s]
C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
```

```
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
```

```
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

C:\Users\shari\anaconda3\lib\site-
packages\dataprep\eda\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
```

```
<IPython.core.display.HTML object>
```

[67]:
```
dfprocessM1=dfprocess3.copy()
```

[68]:
```
dfprocessM1['targetrisk'].head()
```

[68]:
```
0    0
1    0
2    0
3    0
4    0
Name: targetrisk, dtype: object
```

[69]:
```
y=dfprocessM1.targetrisk
X=dfprocessM1.drop(columns=['targetrisk'])
```

[70]:
```
#Converting categorial variables into Dummy / indicator variable
X = pd.get_dummies(X)
X.shape
```

[70]:
```
(36457, 69)
```

[71]:
```
X=X.dropna().reset_index(drop=True)
```

[72]:
```
y.shape
```

[72]:
```
(36457,)
```

[73]:
```
#y=dfprocessM1.targetrisk
#X=dfprocessM1.drop(columns=['targetrisk'])

#@title Splitting dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=1) # 70% training and 30% test
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [73], in <cell line: 5>()
      1 #y=dfprocessM1.targetrisk
      2 #X=dfprocessM1.drop(columns=['targetrisk'])
      3
      4 #@title Splitting dataset into training set and test set
----> 5 X_train, X_test, y_train, y_test =␣
 ↪train_test_split(X, y, test_size=0.3, random_state=1)

File ~\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2417, in␣
 ↪train_test_split(test_size, train_size, random_state, shuffle, stratify,␣
 ↪*arrays)
```

```
   2414 if n_arrays == 0:
   2415     raise ValueError("At least one array required as input")
-> 2417 arrays = indexable(*arrays)
   2419 n_samples = _num_samples(arrays[0])
   2420 n_train, n_test = _validate_shuffle_split(
   2421     n_samples, test_size, train_size, default_test_size=0.25
   2422 )

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:378, in␣
 ↪indexable(*iterables)
    359 """Make arrays indexable for cross-validation.
    360
    361 Checks consistent length, passes through None, and ensures that␣
 ↪everything
   (…)
    374     sparse matrix, or dataframe) or `None`.
    375 """
    377 result = [_make_indexable(X) for X in iterables]
--> 378 check_consistent_length(*result)
    379 return result

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:332, in␣
 ↪check_consistent_length(*arrays)
    330 uniques = np.unique(lengths)
    331 if len(uniques) > 1:
--> 332     raise ValueError(
    333         "Found input variables with inconsistent numbers of samples: %r
    334         % [int(l) for l in lengths]
    335     )

ValueError: Found input variables with inconsistent numbers of samples: [30322,
 ↪36457]
```

[ ]:
```
dfprocess3.head(2)
```

[ ]:
```
dfprocess3.info()
```