Importing the libraries

In [1]:

```
#Importing the required libraries
import pandas as pd
import numpy as np
from collections import Counter as c
import matplotlib.pyplot as plt
from sklearn import preprocessing
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
!pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in c:\users\nagay
elamarthi\anaconda3\lib\site-packages (1.0.116)
Requirement already satisfied: requests in c:\users\nagayelamarthi\anaconda3
\lib\site-packages (from ibm_watson_machine_learning) (2.25.1)
Requirement already satisfied: certifi in c:\users\nagayelamarthi\anaconda3
\lib\site-packages (from ibm_watson_machine_learning) (2020.12.5)
Requirement already satisfied: lomond in c:\users\nagayelamarthi\anaconda3\l
ib\site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: ibm-cos-sdk==2.7.* in c:\users\nagayelamarthi
\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2.7.0)
Requirement already satisfied: packaging in c:\users\nagayelamarthi\anaconda
3\lib\site-packages (from ibm_watson_machine_learning) (20.9)
Requirement already satisfied: pandas<1.3.0,>=0.24.2 in c:\users\nagayelamar
thi\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (1.2.4)
Requirement already satisfied: urllib3 in c:\users\nagayelamarthi\anaconda3
\lib\site-packages (from ibm_watson_machine_learning) (1.26.4)
Requirement already satisfied: tabulate in c:\users\nagayelamarthi\anaconda3
\lib\site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\nagayelama
rthi\anaconda3\lib\site-packages (from ibm-cos-sdk==2.7.*->ibm_watson_machin
e_learning) (0.10.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.7.0 in c:\users\nag
ayelamarthi\anaconda3\lib\site-packages (from ibm-cos-sdk==2.7.*->ibm_watson
_machine_learning) (2.7.0)
Requirement already satisfied: ibm-cos-sdk-core==2.7.0 in c:\users\nagayelam
arthi\anaconda3\lib\site-packages (from ibm-cos-sdk==2.7.*->ibm_watson_machi
ne_learning) (2.7.0)
Requirement already satisfied: docutils<0.16,>=0.10 in c:\users\nagayelamart
hi\anaconda3\lib\site-packages (from ibm-cos-sdk-core==2.7.0->ibm-cos-sdk==
2.7.*->ibm_watson_machine_learning) (0.15.2)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\nagay
elamarthi\anaconda3\lib\site-packages (from ibm-cos-sdk-core==2.7.0->ibm-cos
-sdk==2.7.*->ibm_watson_machine_learning) (2.8.1)
Requirement already satisfied: numpy>=1.16.5 in c:\users\nagayelamarthi\anac
onda3\lib\site-packages (from pandas<1.3.0,>=0.24.2->ibm_watson_machine_lear
ning) (1.19.5)
Requirement already satisfied: pytz>=2017.3 in c:\users\nagayelamarthi\anaco
nda3\lib\site-packages (from pandas<1.3.0,>=0.24.2->ibm_watson_machine_learn
ing) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\nagayelamarthi\anaconda3
\lib\site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.7.
0->ibm-cos-sdk==2.7.*->ibm_watson_machine_learning) (1.15.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\nagayelamarthi\anaco
nda3\lib\site-packages (from requests->ibm_watson_machine_learning) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\nagayelamarthi
\anaconda3\lib\site-packages (from requests->ibm_watson_machine_learning)
(4.0.0)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\nagayelamarthi\a
naconda3\lib\site-packages (from packaging->ibm_watson_machine_learning) (2.
4.7)
```

In [ ]:

```

```

Loading the dataset

In [5]:

```
dataset = pd.read_csv('credit_train.csv')
dataset.head()
```

Out[5]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Home Ownership	Purp
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	Fully Paid	445412.0	Short Term	709.0	1167493.0	8 years	Home Mortgage	Ho Improveme
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	Fully Paid	262328.0	Short Term	NaN	NaN	10+ years	Home Mortgage	D Consolida
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	Fully Paid	99999999.0	Short Term	741.0	2231892.0	8 years	Own Home	D Consolida

In [6]:

```
#finding the number of rows and columns
dataset.shape
```

Out[6]:

```
(100514, 19)
```

In [7]:

```
#lists out the names of the columns
dataset.columns
```

Out[7]:

```
Index(['Loan ID', 'Customer ID', 'Loan Status', 'Current Loan Amount', 'Ter
m',
       'Credit Score', 'Annual Income', 'Years in current job',
       'Home Ownership', 'Purpose', 'Monthly Debt', 'Years of Credit Histor
y',
       'Months since last delinquent', 'Number of Open Accounts',
       'Number of Credit Problems', 'Current Credit Balance',
       'Maximum Open Credit', 'Bankruptcies', 'Tax Liens'],
      dtype='object')
```

In [8]:

```
# It will display the first five rows of the dataset
dataset.head()
```

Out[8]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Hor Ownersh
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	Fully Paid	445412.0	Short Term	709.0	1167493.0	8 years	Hor Mortga
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	Fully Paid	262328.0	Short Term	NaN	NaN	10+ years	Hor Mortga
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	Fully Paid	99999999.0	Short Term	741.0	2231892.0	8 years	Own Hor
3	77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	Fully Paid	347666.0	Long Term	721.0	806949.0	3 years	Own Hor
4	d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	Fully Paid	176220.0	Short Term	NaN	NaN	5 years	Re

Null Values

In [9]:

```
#lists the  null values in every column of the dataset
dataset.isnull().any()
```

Out[9]:

```
Loan ID                         True
Customer ID                     True
Loan Status                     True
Current Loan Amount             True
Term                            True
Credit Score                    True
Annual Income                   True
Years in current job            True
Home Ownership                  True
Purpose                         True
Monthly Debt                    True
Years of Credit History         True
Months since last delinquent    True
Number of Open Accounts         True
Number of Credit Problems       True
Current Credit Balance          True
Maximum Open Credit             True
Bankruptcies                    True
Tax Liens                       True
dtype: bool
```

In [10]:

```
#Finding the  sum of null values in every column of the dataset
dataset.isnull().sum()
```

Out[10]:

```
Loan ID                           514
Customer ID                       514
Loan Status                       514
Current Loan Amount               514
Term                              514
Credit Score                    19668
Annual Income                   19668
Years in current job             4736
Home Ownership                    514
Purpose                           514
Monthly Debt                      514
Years of Credit History           514
Months since last delinquent    53655
Number of Open Accounts           514
Number of Credit Problems         514
Current Credit Balance            514
Maximum Open Credit               516
Bankruptcies                      718
Tax Liens                         524
dtype: int64
```

Categorical Columns

In [11]:

```
#lists the columns with categorical data
object_train_df=dataset.select_dtypes(include=['object'])
object_train_df.columns
```

Out[11]:

```
Index(['Loan ID', 'Customer ID', 'Loan Status', 'Term', 'Years in current jo
b',
       'Home Ownership', 'Purpose'],
      dtype='object')
```

Numerical Columns

In [12]:

```
#lists the columns with numerical data
num_train_df=dataset.select_dtypes(include=['int','float'])
num_train_df.columns
```

Out[12]:

```
Index(['Current Loan Amount', 'Credit Score', 'Annual Income', 'Monthly Deb
t',
       'Years of Credit History', 'Months since last delinquent',
       'Number of Open Accounts', 'Number of Credit Problems',
       'Current Credit Balance', 'Maximum Open Credit', 'Bankruptcies',
       'Tax Liens'],
      dtype='object')
```

Dropping Loan Status Null Values and Labeling it

In [13]:

```
dataset.dropna(subset=['Loan Status'], inplace=True)
```

In [14]:

```
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
dataset['Loan Status'] = le.fit_transform(dataset['Loan Status'])
```

In [15]:

```
dataset
```

Out[15]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Owr
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	1	445412.0	Short Term	709.0	1167493.0	8 years	M
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	1	262328.0	Short Term	NaN	NaN	10+ years	M
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	1	99999999.0	Short Term	741.0	2231892.0	8 years	Owr
3	77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	1	347666.0	Long Term	721.0	806949.0	3 years	Owr
4	d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	1	176220.0	Short Term	NaN	NaN	5 years	
...	...	...	...	...	...	...	...	...	
99995	3f94c18c-ba8f-45d0-8610-88a684a410a9	2da51983-cfef-4b8f-a733-5dfaf69e9281	1	147070.0	Short Term	725.0	475437.0	7 years	Owr
99996	06eba04f-58fc-424a-b666-ed72aa008900	77f2252a-b7d1-4b07-a746-1202a8304290	1	99999999.0	Short Term	732.0	1289416.0	1 year	
99997	e1cb4050-eff5-4bdb-a1b0-aabd3f7eaac7	2ced5f10-bd60-4a11-9134-cadce4e7b0a3	1	103136.0	Short Term	742.0	1150545.0	6 years	
99998	81ab928b-d1a5-4523-9a3c-271ebb01b4fb	3e45ffda-99fd-4cfc-b8b8-446f4a505f36	1	530332.0	Short Term	746.0	1717524.0	9 years	
99999	c63916c6-6d46-47a9-949a-51d09af4414f	1b3014be-5c07-4d41-abe7-44573c375886	1	99999999.0	Short Term	743.0	935180.0	NaN	Owr

100000 rows × 19 columns

Term column Labeling

In [16]:

```
dataset['Term'].replace(('Short Term','Long Term'),(0,1),inplace=True)
dataset.head()
```

Out[16]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Home Ownership	Purp
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	1	445412.0	0	709.0	1167493.0	8 years	Home Mortgage	Hc Improveme
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	1	262328.0	0	NaN	NaN	10+ years	Home Mortgage	[Consolida
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	1	99999999.0	0	741.0	2231892.0	8 years	Own Home	[Consolida

Scaling Credit Score Column

In [17]:

```
#Applying lamda function
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: (val /10) if val>850 el
```

Handling Null values of Credit Score Column

In [18]:

```
do_nothing = lambda: None
cscoredf = dataset[dataset['Term']==0]
stermAVG = cscoredf['Credit Score'].mean()
lscoredf = dataset[dataset['Term']==1]
ltermAVG = lscoredf['Credit Score'].mean()
dataset.loc[(dataset.Term==0) & (dataset['Credit Score'].isnull()),'Credit Score'] = stermA
dataset.loc[(dataset.Term==1) & (dataset['Credit Score'].isnull()),'Credit Score'] = ltermA
```

In [19]:

```
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: "Poor" if np.isreal(val
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: "Average" if np.isreal(
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: "Good" if np.isreal(val
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: "Very Good" if np.isrea
dataset['Credit Score'] = dataset['Credit Score'].apply(lambda val: "Exceptional" if np.isr
```

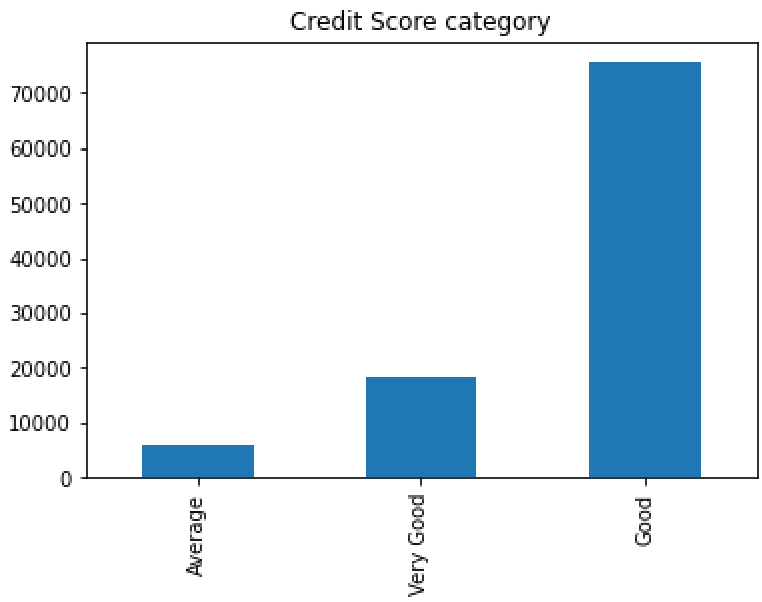
Analying the data

In [20]:

```
## The graph lists out the counts in an ascending way
dataset['Credit Score'].value_counts().sort_values(ascending = True).plot(kind='bar', title
```

Out[20]:

```
<AxesSubplot:title={'center':'Credit Score category'}>
```



Annual income column

In [21]:

```
# Prints the sum of missing values in the column Annual Income.
print("There are",dataset['Annual Income'].isna().sum(), "Missing Annual Income Values.")
```

```
There are 19154 Missing Annual Income Values.
```

In [22]:

```
#by using fillna function we are filling the null values with the mean method inplace where
dataset['Annual Income'].fillna(dataset['Annual Income'].mean(), inplace=True)
```

In [23]:

```
#finding the data shape
dataset.shape
```

Out[23]:

```
(100000, 19)
```

In [26]:

```
#By using the counter function we are to get the count of Good, Very Good and Average.
from collections import Counter as c
print(c(dataset['Credit Score']))  #returns the class count values
```

```
Counter({1: 75506, 2: 18479, 0: 6015})
```

In [27]:

```
##applying label encoder
dataset['Credit Score'] = le.fit_transform(dataset['Credit Score'])
c(dataset['Credit Score'])
```

Out[27]:

```
Counter({1: 75506, 2: 18479, 0: 6015})
```

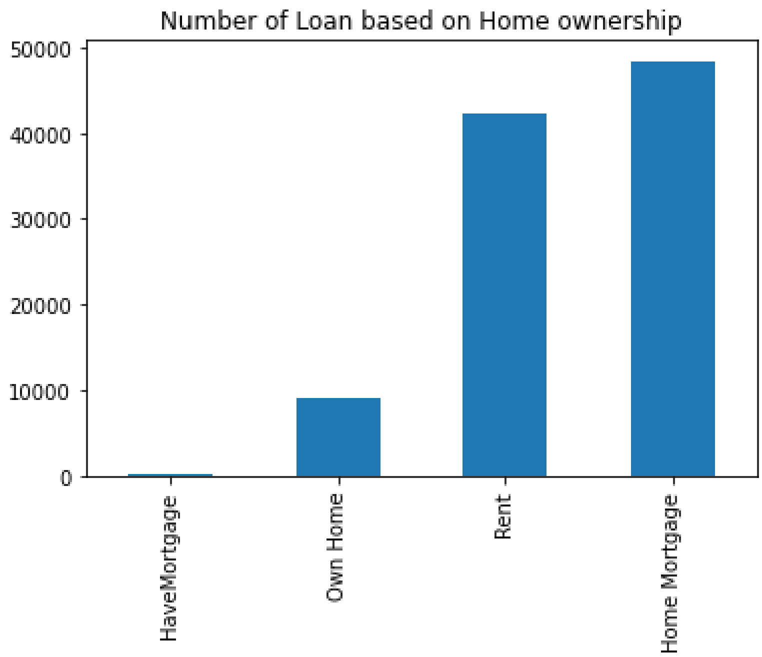
Home Ownership Column

In [28]:

```
#Home Ownership Column   we are sorting the elements with values in ascending order.
dataset['Home Ownership'].value_counts().sort_values(ascending = True).plot(kind='bar', tit
```

Out[28]:

```
<AxesSubplot:title={'center':'Number of Loan based on Home ownership'}>
```



In [29]:

```
print(c(dataset['Home Ownership']))
dataset['Home Ownership'] =  le.fit_transform(dataset['Home Ownership'])
print(c(dataset['Home Ownership']))
```

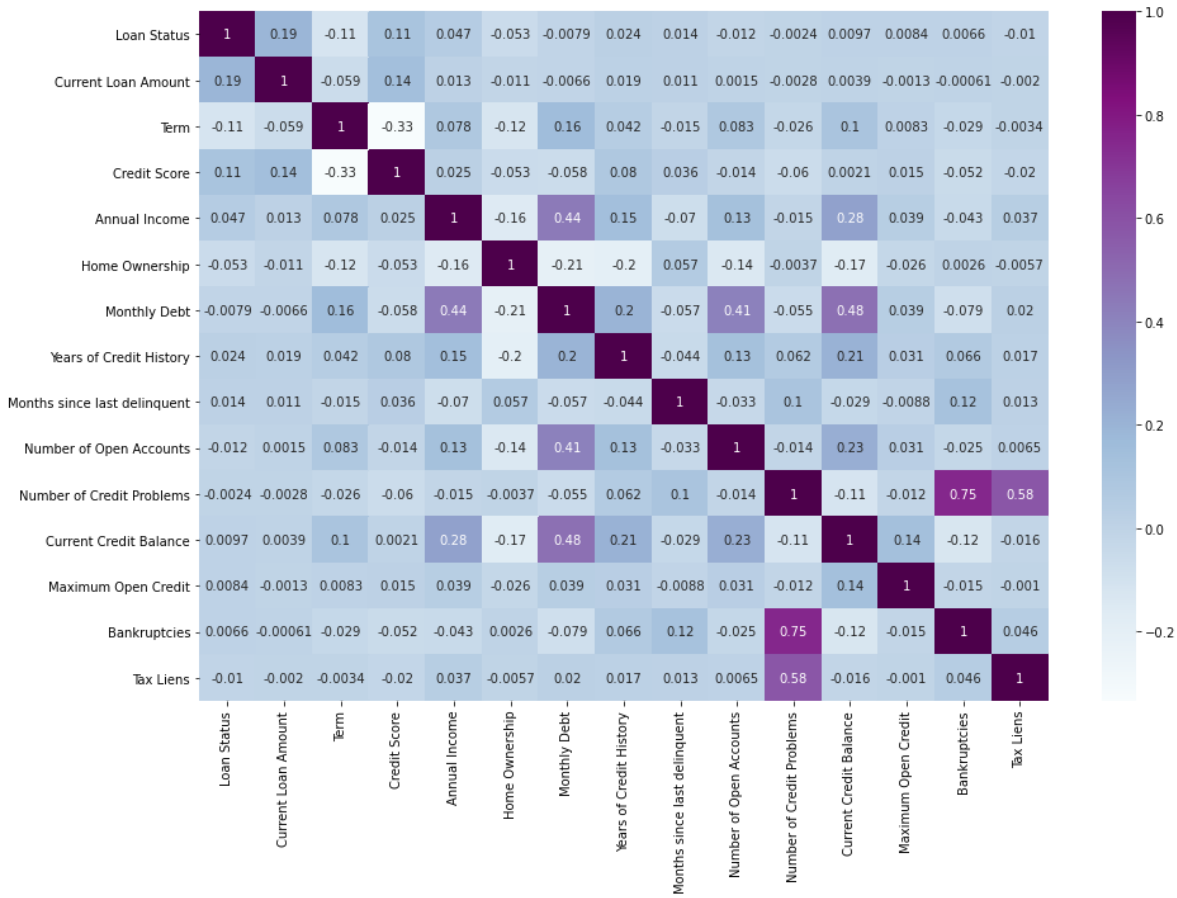
```
Counter({'Home Mortgage': 48410, 'Rent': 42194, 'Own Home': 9182, 'HaveMortg
age': 214})
Counter({1: 48410, 3: 42194, 2: 9182, 0: 214})
```

In [37]:

```
corr = dataset.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr,annot = True, cmap="BuPu")
```

Out[37]:

```
<AxesSubplot:>
```



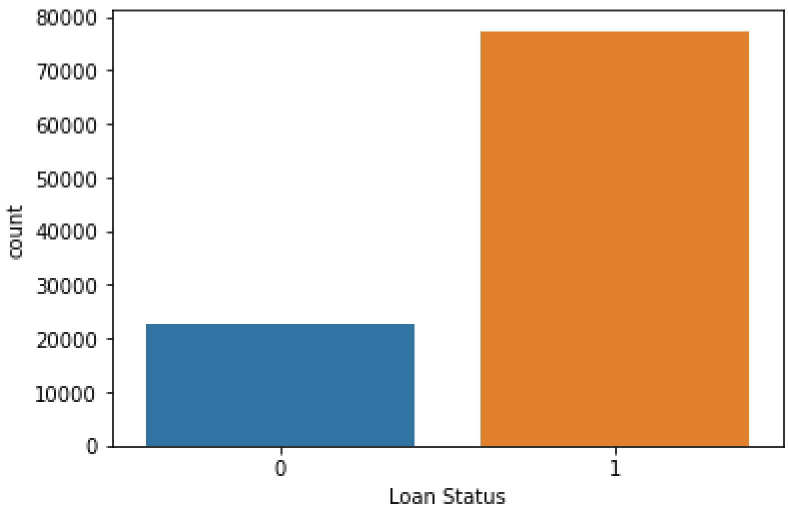
In [34]:

```
sns.countplot(dataset['Loan Status'])
```

```
C:\Users\Nagayelamarthi\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```

Out[34]:

```
<AxesSubplot:xlabel='Loan Status', ylabel='count'>
```



In [ ]:

```

```

Years in current job

In [34]:

```
dataset['Years in current job']=dataset['Years in current job'].str.extract(r"(\d+)")
dataset['Years in current job'] = dataset['Years in current job'].astype(float)
```

In [35]:

```
expmean = dataset['Years in current job'].mean()
```

In [36]:

```
dataset['Years in current job'].fillna(expmean, inplace=True)
dataset['Years in current job'].fillna(expmean, inplace=True)
```

In [37]:

```
dataset
```

Out[37]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	1	445412.0	0	1	1.167493e+06	8.000000
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	1	262328.0	0	1	1.378277e+06	10.000000
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	1	99999999.0	0	2	2.231892e+06	8.000000
3	77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	1	347666.0	1	1	8.069490e+05	3.000000
4	d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	1	176220.0	0	1	1.378277e+06	5.000000
...	...	...	...	...	...	...	...	...
99995	3f94c18c-ba8f-45d0-8610-88a684a410a9	2da51983-cfef-4b8f-a733-5dfaf69e9281	1	147070.0	0	1	4.754370e+05	7.000000
99996	06eba04f-58fc-424a-b666-ed72aa008900	77f2252a-b7d1-4b07-a746-1202a8304290	1	99999999.0	0	1	1.289416e+06	1.000000
99997	e1cb4050-eff5-4bdb-a1b0-aabd3f7eaac7	2ced5f10-bd60-4a11-9134-cadce4e7b0a3	1	103136.0	0	2	1.150545e+06	6.000000
99998	81ab928b-d1a5-4523-9a3c-271ebb01b4fb	3e45ffda-99fd-4cfc-b8b8-446f4a505f36	1	530332.0	0	2	1.717524e+06	9.000000
99999	c63916c6-6d46-47a9-949a-51d09af4414f	1b3014be-5c07-4d41-abe7-44573c375886	1	99999999.0	0	2	9.351800e+05	5.977594

100000 rows × 19 columns

Dropping unwanted columns

In [38]:

```
dataset = dataset.drop(['Loan ID','Customer ID','Purpose'], axis=1)
```

Credit Problems

In [39]:

```
#Normalizing
dataset['Credit Problems'] = dataset['Number of Credit Problems'].apply(lambda x: "No Credi
```

In [40]:

```
print(c(dataset['Credit Problems']))
dataset['Credit Problems'] = le.fit_transform(dataset['Credit Problems'])
print(c(dataset['Credit Problems']))
```

```
Counter({'No Credit Problem': 86035, 'Some Credit promblem': 13879, 'Major C
redit Problems': 86})
Counter({1: 86035, 2: 13879, 0: 86})
```

Credit Age

In [41]:

```
dataset['Credit Age'] = dataset['Years of Credit History'].apply(lambda x: "Short Credit Ag
                                 else ("Good Credit Age" if x>5 and x<17 else "Exceptional C
```

In [42]:

```
print(c(dataset['Credit Age']))
dataset['Credit Age'] = le.fit_transform(dataset['Credit Age'])
print(c(dataset['Credit Age']))
```

```
Counter({'Exceptional Credit Age': 49958, 'Good Credit Age': 49848, 'Short C
redit Age': 194})
Counter({0: 49958, 1: 49848, 2: 194})
```

In [43]:

```
dataset = dataset.drop(['Months since last delinquent','Number of Open Accounts',
                  'Maximum Open Credit','Current Credit Balance','Monthly Debt'],axis=1)
```

Tax Liens

In [44]:

```
dataset['Tax Liens'] = dataset['Tax Liens'].apply(lambda x: "No Tax Lien" if x==0 else ("So
```

In [45]:

```
print(c(dataset['Tax Liens']))
dataset['Tax Liens'] = le.fit_transform(dataset['Tax Liens'])
print(c(dataset['Tax Liens']))
```

```
Counter({'No Tax Lien': 98062, 'Some Tax Liens': 1717, 'Many Tax Liens': 22
1})
Counter({1: 98062, 2: 1717, 0: 221})
```

Bankruptcies

In [46]:

```
dataset['Bankruptcies'] = dataset['Bankruptcies'].apply(lambda x: "No bankruptcies" if x==0
```

In [47]:

```
print(c(dataset['Bankruptcies']))
dataset['Bankruptcies'] = le.fit_transform(dataset['Bankruptcies'])
print(c(dataset['Bankruptcies']))
```

```
Counter({'No bankruptcies': 88774, 'Some Bankruptcies': 10892, 'Many Bankrup
tcies': 334})
Counter({1: 88774, 2: 10892, 0: 334})
```

Annual Income

In [48]:

```
meanxoutlier = dataset[dataset['Annual Income'] < 99999999.00 ]['Annual Income'].mean()
stddevxoutlier = dataset[dataset['Annual Income'] < 99999999.00 ]['Annual Income'].std()
poorline = meanxoutlier -  stddevxoutlier
richline = meanxoutlier + stddevxoutlier
```

In [49]:

```
dataset['Annual Income'] = dataset['Annual Income'].apply(lambda x: "Low Income" if x<=poor
```

In [50]:

```
print(c(dataset['Annual Income']))
dataset['Annual Income'] = le.fit_transform(dataset['Annual Income'])
print(c(dataset['Annual Income']))
```

```
Counter({'Average Income': 86004, 'High Income': 9145, 'Low Income': 4851})
Counter({0: 86004, 1: 9145, 2: 4851})
```

Current Loan Amount

In [51]:

```
lmeanxoutlier = dataset[dataset['Current Loan Amount'] < 99999999.00 ]['Current Loan Amount
lstddevxoutlier = dataset[dataset['Current Loan Amount'] < 99999999.00 ]['Current Loan Amou
lowrange = lmeanxoutlier - lstddevxoutlier
highrange = lmeanxoutlier + lstddevxoutlier
print(lowrange, highrange)
```

```
126051.43019084723 498575.76557037106
```

In [52]:

```
dataset['Current Loan Amount'] = dataset['Current Loan Amount'].apply(lambda x: "Small Loan
```

In [53]:

```
print(c(dataset['Current Loan Amount']))
dataset['Current Loan Amount'] = le.fit_transform(dataset['Current Loan Amount'])
print(c(dataset['Current Loan Amount']))
```

```
Counter({'Medium Loan': 60112, 'Big Loan': 26506, 'Small Loan': 13382})
Counter({1: 60112, 0: 26506, 2: 13382})
```

In [54]:

```
dataset.shape
```

Out[54]:

```
(100000, 13)
```

Seperating Dependent and Independent Columns

In [55]:

```
y = dataset['Loan Status']
X = dataset.drop(['Loan Status'],axis=1)
```

In [56]:

```
dataset.head()
```

Out[56]:

	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Home Ownership	Years of Credit History	Number of Credit Problems	Bankruptcies
0	1	1	0	1	0	8.0	1	17.2	1.0	[illegible]
1	1	1	0	1	0	10.0	1	21.1	0.0	[illegible]
2	1	0	0	2	1	8.0	2	14.9	1.0	[illegible]
3	1	1	1	1	0	3.0	2	12.0	0.0	[illegible]
4	1	1	0	1	0	5.0	3	6.1	0.0	[illegible]

Performing Train and test split

In [57]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [58]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [59]:

```
#By using DecisionTree we are fitting the model
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[59]:

```
DecisionTreeClassifier()
```

In [60]:

```
X_train.shape
```

Out[60]:

```
(67000, 12)
```

In [61]:

```
y_pred_dt =dt.predict(X_test)  #prediction
c(y_pred_dt)
```

Out[61]:

```
Counter({0: 6716, 1: 26284})
```

In [62]:

```
X_train
```

Out[62]:

```
array([[ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497,  0.98973021],
       [-1.40134783, -0.62204006,  1.82489603, ..., -0.10969543,
        -0.39894497,  0.98973021],
       [-1.40134783, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497, -0.99423114],
       ...,
       [ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497, -0.99423114],
       [ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
         2.47147096,  0.98973021],
       [ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497, -0.99423114]])
```

In [66]:

```
X_test
```

Out[66]:

```
array([[ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497,  0.98973021],
       [ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
         2.47147096,  0.98973021],
       [ 0.21538779,  1.60761349, -2.34480128, ..., -0.10969543,
        -0.39894497,  0.98973021],
       ...,
       [ 1.83212341, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497,  0.98973021],
       [ 1.83212341, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497,  0.98973021],
       [ 0.21538779, -0.62204006, -0.25995262, ..., -0.10969543,
        -0.39894497, -0.99423114]])
```

In [67]:

```
from sklearn.metrics import accuracy_score
```

In [69]:

```
accuracy_score(y_pred_dt,y_test)
```

Out[69]:

```
0.6917575757575758
```

Creating a pickle file dumping the model in it

In [70]:

```
 #importing the pickle file
import pickle
#Dumping the model into the pickle file
pickle.dump(dt,open('loan.pkl','wb'))
```

In [ ]:

```
```