logistic regerssion

August 6, 2022

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
color = sns.color_palette()
import sklearn.metrics as metrics
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
Default = pd.read_csv("../../datasets/Default_Fin.csv")
```

[3]:
```
Default.head()
```

[3]:

	Index	Employed	Bank Balance	Annual Salary	Defaulted?
0	1	1	8754.36	532339.56	0
1	2	0	9806.16	145273.56	0
2	3	1	12882.60	381205.68	0
3	4	1	6351.00	428453.88	0
4	5	1	9427.92	461562.00	0

[4]:
```
Default.shape
```

[4]:
```
(10000, 5)
```

[5]:
```
Default.rename(columns = {"Defaulted?":"defaulted"}, inplace = True)
```

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

[6]:

	Index	Employed	Bank Balance	Annual Salary	defaulted
0	1	1	8754.36	532339.56	0
1	2	0	9806.16	145273.56	0
2	3	1	12882.60	381205.68	0
3	4	1	6351.00	428453.88	0
4	5	1	9427.92	461562.00	0

[7]:
```
#Checking the numerical columns
Default.describe()
```

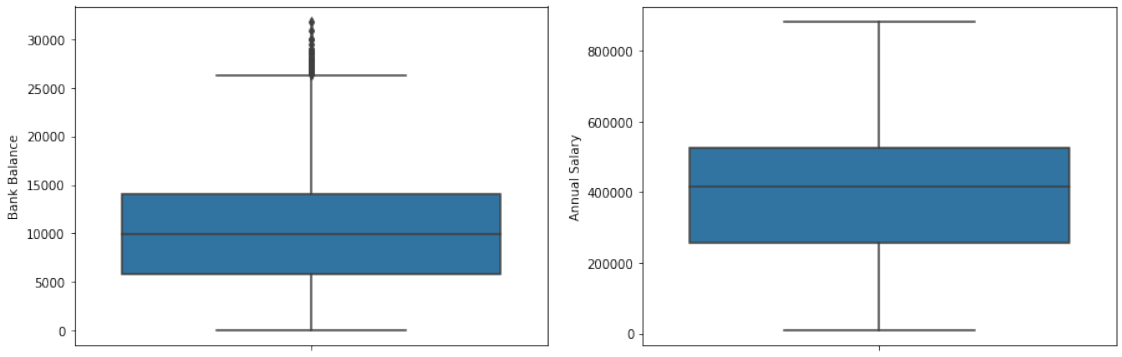
[7]:

	Index	Employed	Bank Balance	Annual Salary	defaulted
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	0.705600	10024.498524	402203.782224	0.033300
std	2886.89568	0.455795	5804.579486	160039.674988	0.179428
min	1.00000	0.000000	0.000000	9263.640000	0.000000
25%	2500.75000	0.000000	5780.790000	256085.520000	0.000000
50%	5000.50000	1.000000	9883.620000	414631.740000	0.000000
75%	7500.25000	1.000000	13995.660000	525692.760000	0.000000
max	10000.00000	1.000000	31851.840000	882650.760000	1.000000

[8]:
```
#lets do "Univariate Analysis"  which will show use how two columns value vary␣
↪with eachother
```

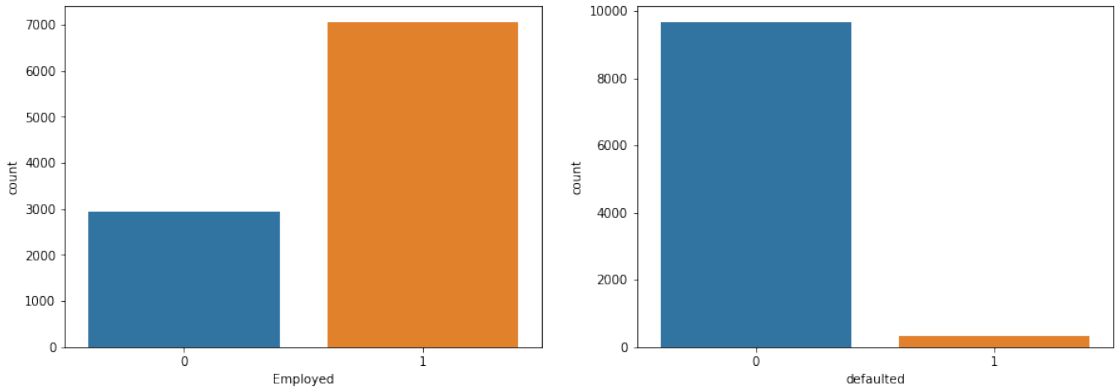
[9]:
```
#boxplot will show us 5 number summary value min.value max.value 25% and 50% 70%
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
sns.boxplot(y = Default['Bank Balance'])

plt.subplot(1,2,2)
sns.boxplot(y = Default['Annual Salary'])
plt.show()
```



[10]:
```
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
sns.countplot(Default['Employed'])

plt.subplot(1,2,2)
sns.countplot(Default['defaulted'])
plt.show()
#we can see that 3000 are not employed while 7000 are employed
#here"0" is for "no" while "1" is for "yes"
# so in the second plot we saw very few employed are default while other can␣
↪repay or not defaulted
```



[11]:
```
#Checking exact value in each figure
```

[12]:
```
#lets see the exact value
Default["Employed"].value_counts()
#here we saw that exactly "7056" are employed while "2944" are "unemployed"
```

[12]:
```
1    7056
0    2944
Name: Employed, dtype: int64
```

[13]:
```
Default["defaulted"].value_counts()
#we see that "9667"employee are not deafaulted while only "333" are defaulted
```

[13]:
```
0    9667
1     333
Name: defaulted, dtype: int64
```

[14]:
```
#'checking Percentage '
```

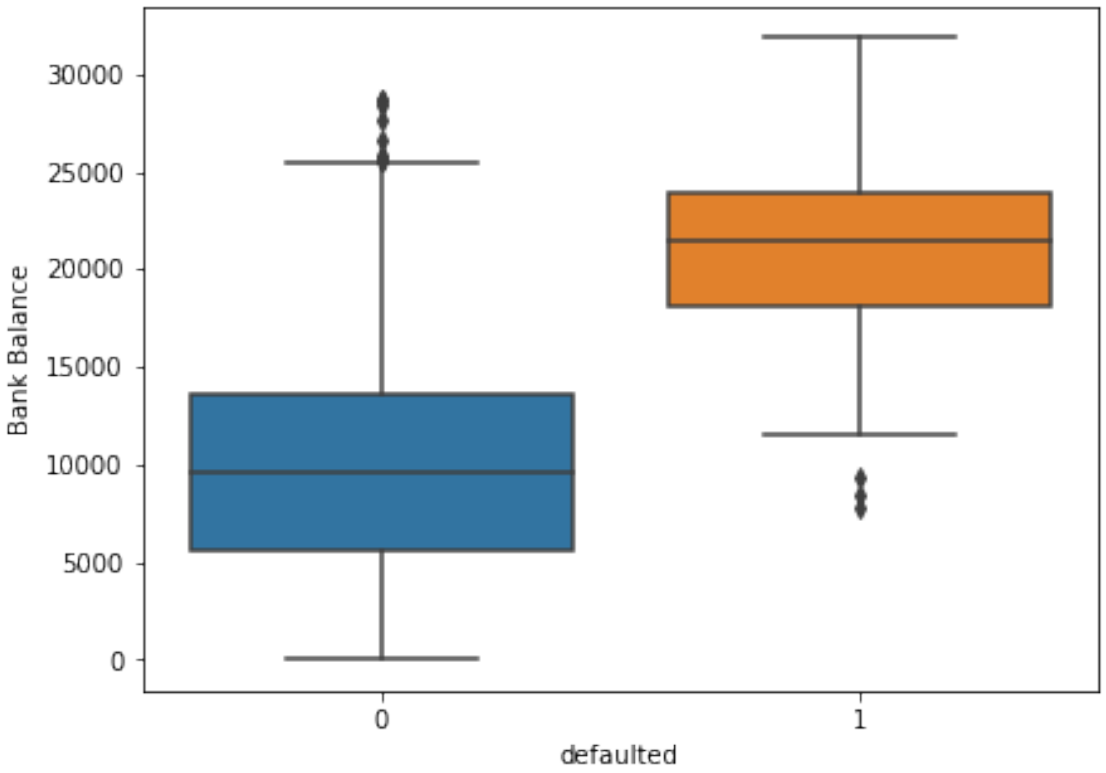
[15]:
```
Default["Employed"].value_counts(normalize = True)
```

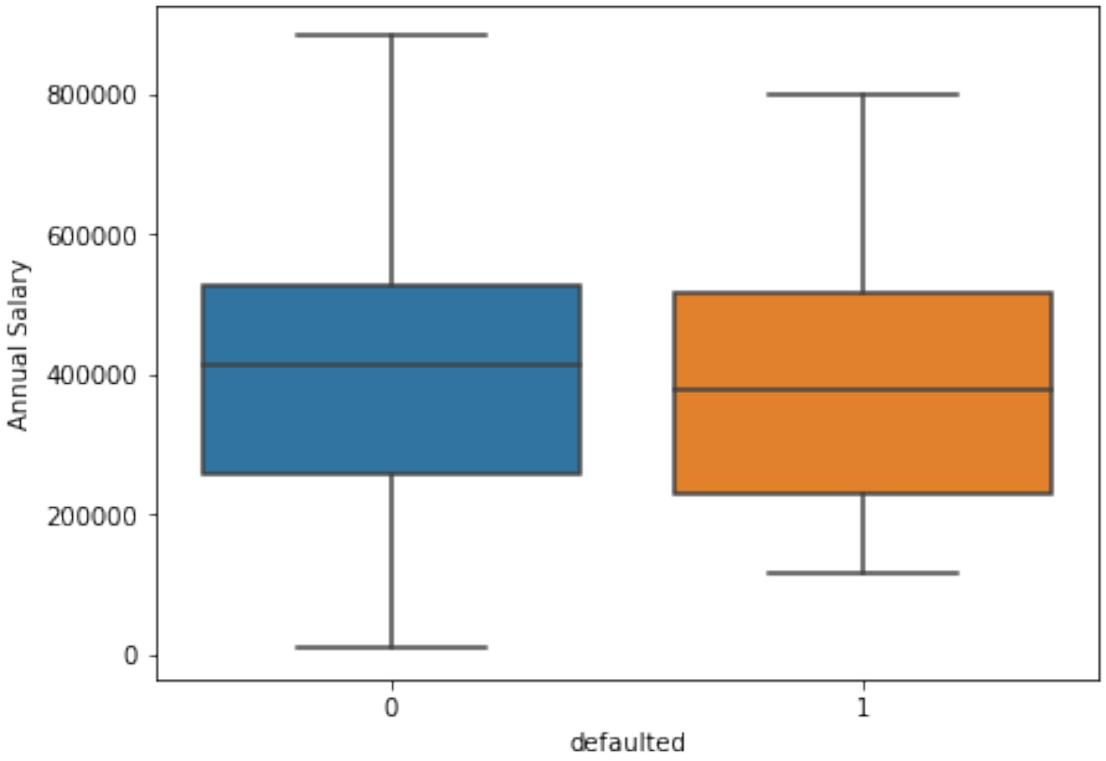
[15]:
```
1    0.7056
0    0.2944
Name: Employed, dtype: float64
```

[16]:
```
#lets do "bivariate Analysis"  which will show use how two columns value vary␣
 ↪with eachother
```

[17]:
```
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
sns.boxplot(Default['defaulted'],Default['Bank Balance'])
#we saw in plot which is not defaulter and their bank balance
```

```
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
sns.boxplot(Default['defaulted'],Default['Annual Salary'])
plt.show()
#we saw in plot which is defaulter and their annual salary
```





[18]:
```
pd.crosstab(Default['Employed'],Default['defaulted'], normalize = 'index').
  ↪round(2)
#here we see the percentage of defaulters vs employee
```

[18]:
```
defaulted     0     1
Employed
0          0.96  0.04
1          0.97  0.03
```

[19]:
```
sns.heatmap(Default[['Bank Balance','Annual Salary']].corr(), annot = True)
```

[19]:
```
<AxesSubplot:>
```

[20]:
```
Default.isnull().sum()
```

[20]:
```
Index            0
Employed         0
Bank Balance     0
Annual Salary    0
defaulted        0
dtype: int64
```

[21]:
```
Q1, Q3 = Default['Bank Balance'].quantile([.25, .75])
IQR = Q3-Q1
LL = Q1 - 1.5+(IQR)
UL = Q3 + 1.5*(IQR)
```

[22]:
```
UL
```

[22]:
```
26317.965
```

[23]:
```
df = Default[Default['Bank Balance'] > UL]
```

[24]:
```
df
```

[24]:

	Index	Employed	Bank Balance	Annual Salary	defaulted
173	174	0	26469.60	171257.88	1
1136	1137	1	29988.24	618051.48	1

1160	1161	0	30032.16	179370.24	1
1359	1360	1	26651.64	488701.20	1
1502	1503	0	27994.56	141242.76	1
1609	1610	0	27239.40	216253.32	1
2096	2097	0	27142.20	240362.04	1
2140	2141	0	27706.68	229323.24	0
2929	2930	0	28647.72	339562.92	1
3162	3163	0	28983.84	209154.00	1
3189	3190	1	26741.64	329260.20	1
3702	3703	0	28445.52	291023.52	0
3855	3856	0	27862.56	255975.72	1
3913	3914	0	28009.44	232030.68	1
3976	3977	0	28658.04	93985.68	0
4060	4061	0	26592.24	250940.40	1
4231	4232	0	27499.44	250046.52	1
4831	4832	0	26595.96	296844.96	0
5461	5462	0	26969.04	215120.64	1
6075	6076	1	28959.84	462486.84	1
6334	6335	1	28125.60	613143.48	1
6882	6883	0	27446.04	224305.68	1
7437	7438	0	29538.12	142542.72	1
7815	7816	0	30941.64	308479.80	1
8264	8265	1	26841.12	445366.56	1
8495	8496	0	31851.84	263164.68	1
8832	8833	0	26491.20	237369.12	1
8992	8993	0	28224.60	288810.60	1
9873	9874	1	28692.12	603634.92	0
9893	9894	1	27460.92	624522.84	1
9978	9979	1	26429.52	567447.12	1

[25]:
```
df['defaulted'].count()
```

[25]: 31

[26]:
```
Default.drop('Index' , axis =1)
```

[26]:

	Employed	Bank Balance	Annual Salary	defaulted
0	1	8754.36	532339.56	0
1	0	9806.16	145273.56	0
2	1	12882.60	381205.68	0
3	1	6351.00	428453.88	0
4	1	9427.92	461562.00	0
…	…	…	…	…
9995	1	8538.72	635908.56	0
9996	1	9095.52	235928.64	0
9997	1	10144.92	703633.92	0
9998	1	18828.12	440029.32	0

```
9999          0       2411.04       202355.40           0

[10000 rows x 4 columns]
```

[27]:
```
df['defaulted'].value_counts(normalize = True)

#now we will see how much percentage of is defaulted in 31 numbers of outlyers
```

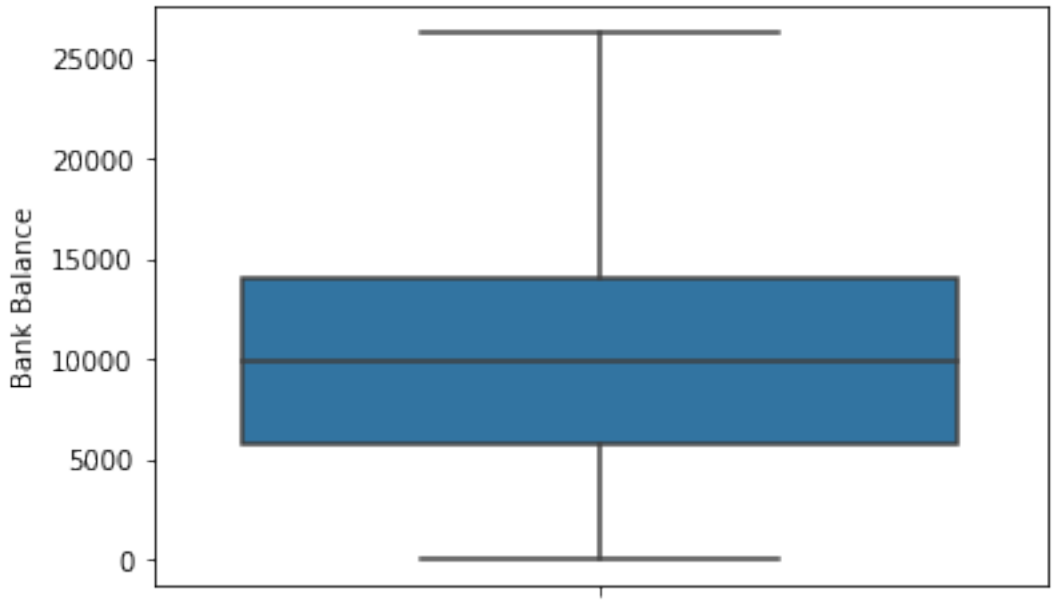
[27]:
```
1    0.83871
0    0.16129
Name: defaulted, dtype: float64
```

[28]:
```
df['defaulted'].value_counts()
```

[28]:
```
1    26
0     5
Name: defaulted, dtype: int64
```

[29]:
```
Default['Bank Balance'] = np.where(Default['Bank Balance']> UL,UL,Default['Bank␣
↪Balance'])
```

[30]:
```
sns.boxplot(y = Default['Bank Balance'])
plt.show()
```



[31]:
```
# As our column of "Employee" is already in numerical digit  so its fine .
#otherwise if its yes and no    we have to change it into 0 and 1 form
```

[32]:
```
Default.drop('Index', axis=1, inplace = True)
```

[33]:
```
Default.head()
```

[33]:

	Employed	Bank Balance	Annual Salary	defaulted
0	1	8754.36	532339.56	0
1	0	9806.16	145273.56	0
2	1	12882.60	381205.68	0
3	1	6351.00	428453.88	0
4	1	9427.92	461562.00	0

[38]:
```
from sklearn.model_selection import train_test_split
```

[39]:
```
X= Default.drop('defaulted' , axis =1)
y = Default['defaulted']
```

[40]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size =0.3,␣
↪random_state = 21, stratify = y)
```

[41]:
```
print(x_train.shape)
print(x_test.shape)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_23936/1480752200.py in <module>
----> 1 print(x_train.shape)
      2 print(x_test.shape)

NameError: name 'x_train' is not defined
```

[42]:
```
from sklearn.linear_model import LogisticRegression
```

[43]:
```
lr = LogisticRegression()
```

[45]:
```
lr.fit(X_train,y_train)
```

[45]:
```
LogisticRegression()
```

[46]:
```
lr.predict(X_test)
```

[46]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[47]:
```
X_test.head()
```

[47]:

	Employed	Bank Balance	Annual Salary
1071	0	11374.68	205133.76

```
9106          1       13545.96       540527.28
501           0       22422.48       245875.20
6475          0        6343.08       235459.56
5943          1       14218.08       414332.28
```

[48]:
```
lr.predict_proba(X_test)
```

[48]:
```
array([[0.85513379, 0.14486621],
       [0.99514363, 0.00485637],
       [0.85610956, 0.14389044],
       …,
       [0.97229472, 0.02770528],
       [0.88550194, 0.11449806],
       [0.98297599, 0.01702401]])
```

[ ]:
```

```