```
 !pip install sweetviz
#uncomment the above if you need to install the library
!pip install auto-sklearn
#uncomment the above if you need to install the library
```

```
  Building wheel for smac (setup.py) ... done
  Created wheel for smac: filename=smac-1.2-py3-none-any.whl size=215916 sha256=5ac2dafa7e8934733924a6a6367c75ba8a652bad08b24a9cd2bff65f964
  Stored in directory: /root/.cache/pip/wheels/a7/3d/a9/7039d2989e5f6b803942a48ec440ab72530234d8809c01cb0e
  Building wheel for liac-arff (setup.py) ... done
  Created wheel for liac-arff: filename=liac_arff-2.5.0-py3-none-any.whl size=11732 sha256=36b07fc7c7e5dfcaf019a064d26314b9a351e06ef0cc9c20
  Stored in directory: /root/.cache/pip/wheels/08/82/8b/5c514221984e88c059b94e36a71d4722e590acaae04deab22e
Successfully built auto-sklearn pynisher smac liac-arff
Installing collected packages: pyrfr, pynisher, liac-arff, emcee, distro, scikit-learn, ConfigSpace, smac, auto-sklearn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of
yellowbrick 1.5 requires scikit-learn>=1.0.0, but you have scikit-learn 0.24.2 which is incompatible.
imbalanced-learn 0.10.1 requires scikit-learn>=1.0.2, but you have scikit-learn 0.24.2 which is incompatible.
Successfully installed ConfigSpace-0.4.21 auto-sklearn-0.15.0 distro-1.8.0 emcee-3.1.4 liac-arff-2.5.0 pynisher-0.6.4 pyrfr-0.8.3 scikit-le
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sweetviz
import autosklearn.classification
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
```

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

```
train.head(5)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	

```
test.head(5)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	

```
# we concat for easy analysis
n = train.shape[0] # we set this to be able to separate the
df = pd.concat([train, test], axis=0)
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	[illegible]
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	[illegible]
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	[illegible]

```
autoEDA = sweetviz.analyze(train)
autoEDA.show_notebook()
```

Done! Use 'show' commands to display/save. [100%] 00:01 -> (00:00 left)

Sweetviz
2.1.4
Get updates, docs & report issues here
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

DataFrame NO COMPARISON TARGET

614	ROWS
0	DUPLICATES
324.2 kb	RAM
13	FEATURES
9	CATEGORICAL
3	NUMERICAL
1	TEXT

ASSOCIATIONS

DataFrame

Part One

1. **An Overview of the data(Hint: Provide the number of records, fields and their data Types.Do for both).**

1 <1% LP002314

```
train.shape
```

    (614, 13)

```
test.shape
```

    (367, 12)

```
train.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 981 entries, 0 to 366
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            981 non-null    object 
 1   Gender             957 non-null    object 
 2   Married            978 non-null    object 
 3   Dependents         956 non-null    object 
 4   Education          981 non-null    object 
 5   Self_Employed      926 non-null    object 
 6   ApplicantIncome    981 non-null    int64  
 7   CoapplicantIncome  981 non-null    float64
 8   LoanAmount         954 non-null    float64
 9   Loan_Amount_Term   961 non-null    float64
```

```
 10  Credit_History     902 non-null    float64
 11  Property_Area      981 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 107.3+ KB
```

The data Consists of 981 observation and 13 variable,with 8 of them being categorical and 5 numerical.614 of the observations are designated for training, while the remaining are for testing.

2. What data Quality issues exist in both train and test?(Hint: Comment any missing values and duplicates **

```
train.isna().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
test.isna().sum()
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
```

```
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

There are missing values in both training and testing data sets, which will be filled before using in machine learning models. There are no duplicate values in the data

3.**How do the loan statuses compare? i.e what is the distribution of each?**

```
sns.countplot(data=train,x='Loan_Status')
plt.show()
```

When we look at the credits status,we see that the weight of the people with a yes credit status in the higher. if the ratio is too high, it is can be a problem for machine learning algorithmns. Fortunately, there is an acceptable situation.

350

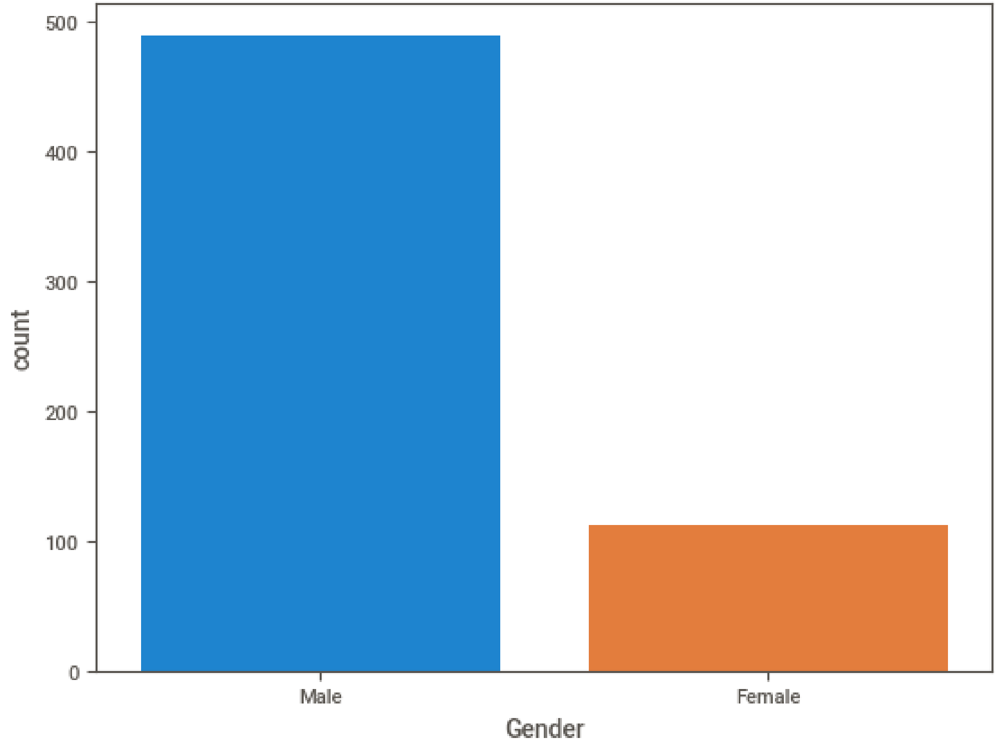
4. How do women and men compare when it comes to defaulting on loans in the historical dataset?

```
default_data=train.query("Loan_Status=='N'")
default_data
```

rried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Pro
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	
Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	
No	0	Graduate	No	1853	2840.0	114.0	360.0	1.0	
No	0	Graduate	No	3510	0.0	76.0	360.0	0.0	
...	...	...	...	...	...	...	...	...	
Yes	2	Not Graduate	Yes	6383	1000.0	187.0	360.0	1.0	
No	NaN	Graduate	No	2987	0.0	88.0	360.0	0.0	
No	3+	Graduate	NaN	416	41667.0	350.0	180.0	NaN	
Yes	0	Not Graduate	No	2400	3800.0	NaN	180.0	1.0	
No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	

```
print(default_data.Gender.value_counts())
sns.countplot(x="Gender",data=train)
```

```
Male      150
Female     37
Name: Gender, dtype: int64
<Axes: xlabel='Gender', ylabel='count'>
```



In the data set, it is seen that men are common among the observation with defaulting on loans

5. How many of the loan Application have dependents based on the histrical data set?

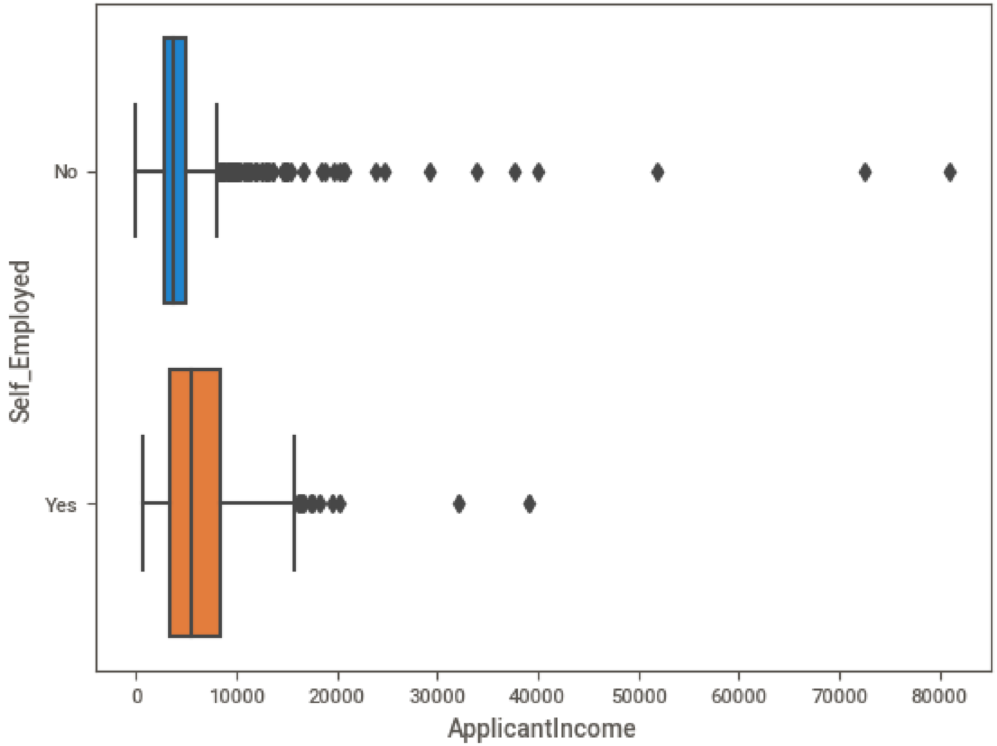
```
df[df['Dependents'] != '0'].shape[0]
```

```
436
```

436 of the loan application are obliged to work after someone.

6. How do the incomes of those who are employed compare to those who self employed based on the histrocial dataset?

```
sns.boxplot(data=df,x="ApplicantIncome",y='Self_Employed')
plt.show()
```



It is seen that the incomes ofthose who are employed are higher on a average.However, Statistical tests will be done to make a clear inference.

```
df.groupby('Self_Employed')['ApplicantIncome'].mean()
```

```
Self_Employed
No     4892.030979
Yes    6912.579832
Name: ApplicantIncome, dtype: float64
```

Let's Check if the daily is normally distributed to decide which statistical test to use

```
from scipy import stats
```

```
p1=stats.shapiro(df.query("Self_Employed=='Yes'")['ApplicantIncome'])
p2=stats.shapiro(df.query("Self_Employed=='No'")['ApplicantIncome'])
print(f"{p1[1]},{p2[1]}")
```

```
6.469167649486574e-13,7.006492321624085e-45
```

It is seen that both are normally distributed(p<0.05).The mean Mann Whitey u test, which is an non -parametric test,will bw used.

```
p3=stats.mannwhitneyu(df.query("Self_Employed=='Yes'")['ApplicantIncome'],df.query("Self_Employed=='No'")['ApplicantIncome'])

if p3[1] < 0.05:
  print("Statistical tests do not reject that those who are employed have higher incomes on average")
else:
    print("Statistical tests reject the at those who are employed have higher incomes on average")
```

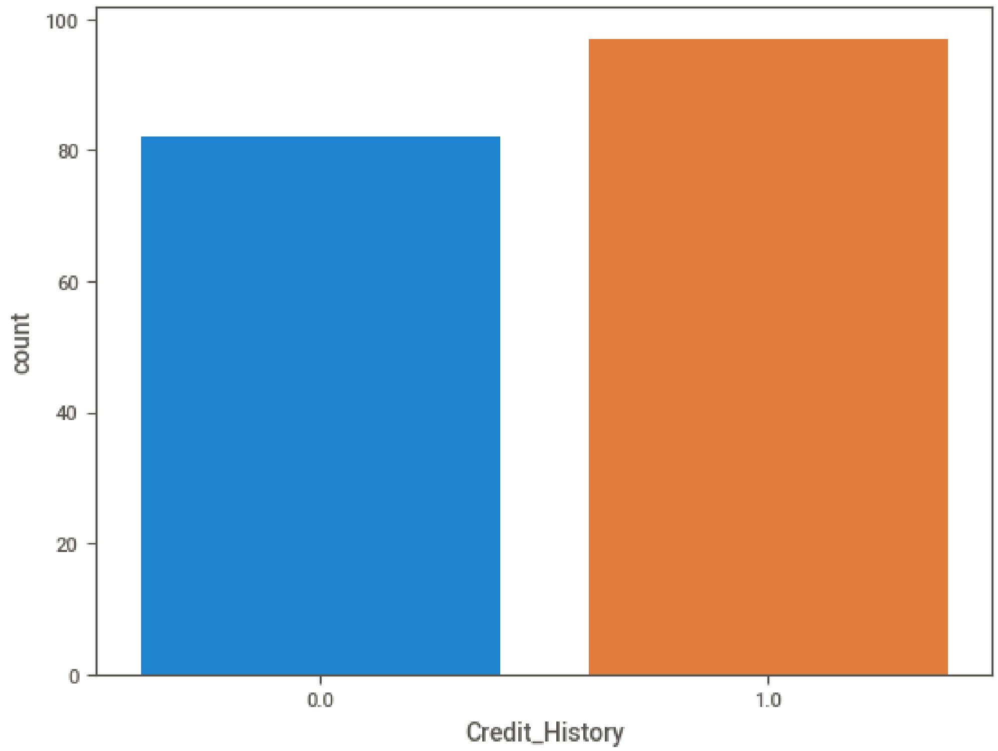
```
Statistical tests do not reject that those who are employed have higher incomes on average
```

Statistical tests do not reject that those who are employed have higher incomes on average.

7. Are applicants with a credit histroy more likely to default than those who do not have one?

a statistacal test will be made between those who have credit histroy and are in default, and those who do not have a credit histrory and are in default.

```
sns.countplot(data=train[train['Loan_Status']=='N'], x="Credit_History")
plt.show()
```



It has been observed that the number of defaults is higher for those with a credit histroy.Now, a statistical test will be conducted to investigate whether like likehood of defaults is higher for those with a credit history compared to these without one.

```
# The number of defaults
train[train['Loan_Status']=='N']['Credit_History'].value_counts()
```

```
1.0    97
0.0    82
Name: Credit_History, dtype: int64
```

```
num_def=np.array([97,82])
```

```
# Observation count
train.Credit_History.value_counts()
```

```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

```
from statsmodels.stats.proportion import proportions_ztest
ob_count=np.array([475,89])
proportions_ztest(count=num_def,nobs=ob_count)
```

```
(-13.339117169122137, 1.3707318825450035e-40)
```

According to the result of the statistical test($p < 0.05$) , the likelihood of default is higher for those who have a credit history compared to those who do not have one

8. Is there a correlation between the applicant's income and the loan amount they applied for?

```
cor_data=df[['ApplicantIncome','LoanAmount']]
sns.heatmap(cor_data.corr(),annot=True)
plt.show()
```

This 0.55 correlation shows that there is not a strong relationship between the income of the person applying for credit and the amount of credit appiled for.

Part Two

Machine learning

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

```
df.isnull().sum()
```

```
Loan_ID               0
Gender               24
Married               3
Dependents           25
Education             0
Self_Employed        55
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           27
Loan_Amount_Term     20
Credit_History       79
Property_Area         0
Loan_Status         367
dtype: int64
```

```
# Filling in Missing values

df.Loan_Status = np.where(df.Loan_Status.isna(), 'Test', df.Loan_Status)
df.Gender = df.Gender.fillna(df.Gender.mode()[0])
df.Married = df.Married.fillna(df.Married.mode()[0])
df.Dependents = df.Dependents.fillna(df.Dependents.mode()[0])
df.Self_Employed = df.Self_Employed.fillna(df.Self_Employed.mode()[0])
df.LoanAmount = df.LoanAmount.fillna(df.groupby('Education')['LoanAmount'].transform('median'))
df.Loan_Amount_Term = df.Loan_Amount_Term.fillna(df.groupby('Education')['Loan_Amount_Term'].transform('median'))
df.Credit_History = df.Credit_History.fillna(df['Credit_History'].median())
df.isna().sum()
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

```
df.drop('Loan_ID', axis = 1, inplace = True)
train_set = df[df['Loan_Status'] != 'Test']
test_set = df[df['Loan_Status'] == 'Test']
```

```
df
```

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Pro
No	0	Graduate	No	5849	0.0	130.5	360.0	1.0	
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	
...	...	...	...	...	...	...	...	...	
Yes	3+	Not Graduate	Yes	4009	1777.0	113.0	360.0	1.0	
Yes	0	Graduate	No	4158	709.0	115.0	360.0	1.0	
No	0	Graduate	No	3250	1993.0	126.0	360.0	1.0	
Yes	0	Graduate	No	5000	2393.0	158.0	360.0	1.0	
No	0	Graduate	Yes	9200	0.0	98.0	180.0	1.0	

ıns

```
# Develop a machine learning model

Y = train_set.Loan_Status
Y = np.where(Y == 'Y', 1, 0)
X = pd.get_dummies(train_set.drop('Loan_Status', axis = 1), drop_first = True)
X
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Yes	Dependents_1	D
0	5849	0.0	130.5	360.0	1.0	1	0	0	
1	4583	1508.0	128.0	360.0	1.0	1	1	1	
2	3000	0.0	66.0	360.0	1.0	1	1	0	
3	2583	2358.0	120.0	360.0	1.0	1	1	0	
4	6000	0.0	141.0	360.0	1.0	1	0	0	
...	...	...	...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	0	0	0	
610	4106	0.0	40.0	180.0	1.0	1	1	0	
611	8072	240.0	253.0	360.0	1.0	1	1	1	
612	7583	0.0	187.0	360.0	1.0	1	1	0	
613	4583	0.0	133.0	360.0	0.0	0	0	0	

614 rows × 14 columns

```
sc=StandardScaler()
X=sc.fit_transform(X)
X=pd.DataFrame(X)
X
```

	0	1	2	3	4	5	6	7	8	9	10	11	
0	0.072991	-0.554487	-0.180823	0.273231	0.411733	0.472343	-1.372089	-0.446339	-0.443713	-0.300975	-0.528362	-0.392601	-0
1	-0.134412	-0.038732	-0.210564	0.273231	0.411733	0.472343	0.728816	2.240448	-0.443713	-0.300975	-0.528362	-0.392601	-0
2	-0.393747	-0.554487	-0.948154	0.273231	0.411733	0.472343	0.728816	-0.446339	-0.443713	-0.300975	-0.528362	2.547117	-0
3	-0.462062	0.251980	-0.305737	0.273231	0.411733	0.472343	0.728816	-0.446339	-0.443713	-0.300975	1.892641	-0.392601	-0
4	0.097728	-0.554487	-0.055908	0.273231	0.411733	0.472343	-1.372089	-0.446339	-0.443713	-0.300975	-0.528362	-0.392601	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	
609	-0.410130	-0.554487	-0.888671	0.273231	0.411733	-2.117107	-1.372089	-0.446339	-0.443713	-0.300975	-0.528362	-0.392601	-0
610	-0.212557	-0.554487	-1.257466	-2.522836	0.411733	0.472343	0.728816	-0.446339	-0.443713	3.322532	-0.528362	-0.392601	-0

```
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import train_test_split, HalvingGridSearchCV, StratifiedKFold, cross_val_score
```

```
# Nested Cross Validation

outer_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
inner_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)

models = []
cv_score = []
fold = 0
for train_index, test_index in outer_cv.split(X, Y):
        X_train, X_test = X.iloc[train_index,:], X.iloc[test_index,:]
        y_train, y_test = Y[train_index], Y[test_index]
        params = {'C': np.logspace(-4, 4, 20),
                   'solver':['liblinear'],
                   'penalty': ['l1', 'l2']}
        model = LogisticRegression()
        grid_search = HalvingGridSearchCV(model, param_grid = params, cv =  inner_cv,  scoring='accuracy')
        grid_result = grid_search.fit(X_train, y_train)
        best_params = grid_result.best_params_
        models.append(model.set_params(**best_params))
        score = grid_search.score(X_test, y_test)
        cv_score.append(score)
        fold = fold + 1
```

```
        print(f'Fold {fold}: train score => {score}')
print(f'CV score = > {np.mean(cv_score)}')
```

```
Fold 1: train score => 0.8048780487804879
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iter
  warnings.warn("Liblinear failed to converge, increase "
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iter
  warnings.warn("Liblinear failed to converge, increase "
Fold 2: train score => 0.7967479674796748
Fold 3: train score => 0.8130081300813008
Fold 4: train score => 0.8048780487804879
Fold 5: train score => 0.8360655737704918
CV score = > 0.8111155537784887
```

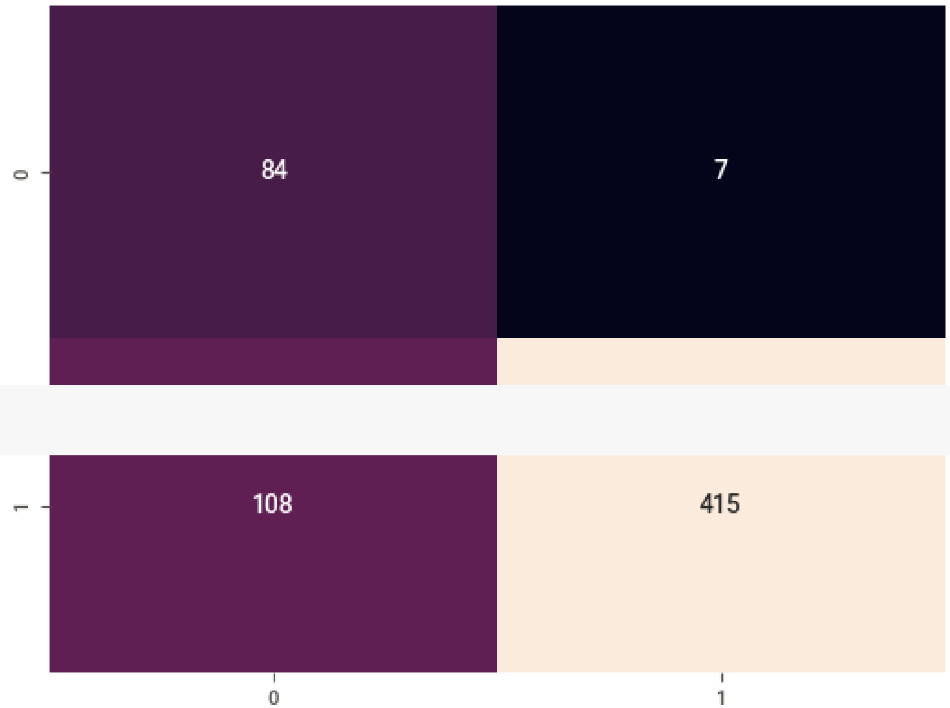
```
best_model = LogisticRegression()
best_model.set_params(**(models[4].get_params()))
best_model
```

```
LogisticRegression(C=0.615848211066026, solver='liblinear')
```

```
best_model.fit(X, Y)
best_model.score(X, Y)
```

```
0.8127035830618893
```

```
y_pred = best_model.predict(X)
cf = confusion_matrix(y_pred, Y)
sns.heatmap(cf, annot = True, fmt = '.4g', cbar=False);
```



0s completed at 7:57 PM