Load the data from “college.csv” that has attributes collected about private and public colleges for a particular year. Predict the private/public status of the colleges from other attributes. Use LabelEncoder to encode the target variable to numerical form. Split the data such that 20% of the data is set aside for testing. Fit a linear SVM from scikit-learn and observe the accuracy. [Hint: Use Linear SVC] Preprocess the data using StandardScalar and fit the same model again. Observe the change in accuracy. Use scikit-learn’s gridsearch to select the best hyper-parameter for a non-linear SVM. Identify the model with the best score and its parameters. [Hint: Refer to model_selection module of Scikit learn]

Objective: Employ SVM from scikit-learn for binary classification and measure the impact of preprocessing data and hyper-parameter search using grid search.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv("College.csv")
df.head()
```

Out[2]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terr
0	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	
1	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	
2	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	
3	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	
4	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	

In [3]:
```
#data exploration
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Private      777 non-null    object 
 1   Apps         777 non-null    int64  
 2   Accept       777 non-null    int64  
 3   Enroll       777 non-null    int64  
 4   Top10perc    777 non-null    int64  
 5   Top25perc    777 non-null    int64  
 6   F.Undergrad  777 non-null    int64  
 7   P.Undergrad  777 non-null    int64  
 8   Outstate     777 non-null    int64  
 9   Room.Board   777 non-null    int64  
 10  Books        777 non-null    int64  
 11  Personal     777 non-null    int64  
 12  PhD          777 non-null    int64  
 13  Terminal     777 non-null    int64  
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64  
 16  Expend       777 non-null    int64  
 17  Grad.Rate    777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 109.4+ KB
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Private', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc',
       'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board', 'Books',
       'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expend',
       'Grad.Rate'],
      dtype='object')
```

In [5]:
```
df.describe()
```

Out[5]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.298584	10440.669241	4357.526384	549.
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.431887	4023.016484	1096.696416	165.
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.000000	2340.000000	1780.000000	96.
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.000000	7320.000000	3597.000000	470.
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.000000	9990.000000	4200.000000	500.
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.000000	12925.000000	5050.000000	600.
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.000000	21700.000000	8124.000000	2340.

In [6]:
```
#checking for missing values
df.isnull().sum()
```

Out[6]:
```
Private        0
Apps           0
Accept         0
Enroll         0
Top10perc      0
Top25perc      0
F.Undergrad    0
P.Undergrad    0
Outstate       0
Room.Board     0
Books          0
Personal       0
PhD            0
Terminal       0
S.F.Ratio      0
perc.alumni    0
Expend         0
Grad.Rate      0
dtype: int64
```

In [7]:
```
#Shape function to record the total number of labels
#Record total number of private
#Similarly record the total number of non-private

print("shape of data:",df.shape)
print("total number of labels: {}".format(df.shape[0]))
print("Number of Private:{}".format(df[df.Private =='Yes'].shape[0]))
print("Number of Public:{}".format(df[df.Private =='No'].shape[0]))
```

```
shape of data: (777, 18)
total number of labels: 777
Number of Private:565
Number of Public:212
```

In [17]:
```
#Split the data such that 20% of the data is set aside for testing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
X,y = df.iloc[:,1:].values, df.iloc[:,0].values
#private > 1
#public > 0
target_encoder = LabelEncoder()
y = target_encoder.fit_transform(y)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
print(X_train.shape)
```

```
(621, 17)
```

In [21]:
```
#Support vector machine classifier. SCV model from sklearn

from sklearn.svm import LinearSVC,SVC
classifier = LinearSVC()

classifier.fit(X_train,y_train)
y_predict = classifier.predict(X_test)
classifier.score(X_test,y_test)
```

Out[21]:
```
0.9230769230769231
```

In [22]:
```
#Performance Matrix

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_predict,y_test))
```

```
[[ 28   1]
 [ 11 116]]
```

In [23]:
```
#fit the SVC Classifier

classifier = SVC()
classifier.fit(X_train,y_train)
classifier.score(X_test,y_test)
```

Out[23]: 0.9230769230769231

In [26]:
```
#Preprocesing the data

from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X,y = df.iloc[:,1:].values, df.iloc[:, 0].values
X = scaler.fit_transform(X)
target_encoder = LabelEncoder()
y = target_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=1)

print(X_train.shape)
```

```
(621, 17)
```

In [27]:
```
#Refiting the SVC Model

classifier = SVC()
classifier.fit(X_train,y_train)
classifier.score(X_test,y_test)
```

Out[27]: 0.9423076923076923

In [31]:
```
#fitting Grid Search

import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
C_range = np.logspace(-2,10,13)
gamma_range =np.logspace(-9,3,13)
param_grid = dict( gamma=gamma_range,C=C_range)
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
```

Out[31]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': array([1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05,
       1.e+06, 1.e+07, 1.e+08, 1.e+09, 1.e+10]),
                         'gamma': array([1.e-09, 1.e-08, 1.e-07, 1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02,
       1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03])})
```

In [32]:
```
#getting the best Hyperparameter

print("The best parameters are %s with a score of %0.2f"
     %(grid.best_params_, grid.best_score_))
```

```
The best parameters are {'C': 1000000.0, 'gamma': 1e-07} with a score of 0.94
```

In [ ]:
```

```