In [2]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import train_test_split, ShuffleSplit, GridSearch
CV
from sklearn.preprocessing import LabelEncoder
```

In [3]:
```
# 1.Load the data from "college.csv" that has attributes collected about priva
te and public colleges
# for a particular year. We will try to predict the private/public status of t
he college from other attributes.
data = pd.read_csv("College.csv")
data.head()
```

Out[3]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Rc
0	Yes	1660	1232	721	23	52	2885	537	7440	
1	Yes	2186	1924	512	16	29	2683	1227	12280	
2	Yes	1428	1097	336	22	50	1036	99	11250	
3	Yes	417	349	137	60	89	510	63	12960	
4	Yes	193	146	55	16	44	249	869	7560	

In [4]:
```
# 2.Use LabelEncoder to encode the target variable in to numerical form and sp
lit the data such that 20% of the data is set aside for testing.
labelencoder = LabelEncoder()
data["Private"] = labelencoder.fit_transform(data["Private"])
data.head()
```

Out[4]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Rc
0	1	1660	1232	721	23	52	2885	537	7440	
1	1	2186	1924	512	16	29	2683	1227	12280	
2	1	1428	1097	336	22	50	1036	99	11250	
3	1	417	349	137	60	89	510	63	12960	
4	1	193	146	55	16	44	249	869	7560	

In [5]:
```
X = data.iloc[:,1:]
Y = data["Private"]

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, rand
om_state=10)
```

In [6]:
```
# 3.Fit a linear svm from scikit learn and observe the accuracy.[Hint:Use Line
ar SVC]
model_svm = svm.LinearSVC()
model_svm.fit(x_train, y_train)
predicted_values = model_svm.predict(x_test)

print("\nAccuracy Score")
print(metrics.accuracy_score(predicted_values,y_test))
```

```
Accuracy Score
0.8974358974358975

C:\Users\hp\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: Convergence
Warning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

In [7]:
```
# 4.Preprocess the data using StandardScalar and fit the same model again and
 observe the change in accuracy.
# [Hint: Refer to scikitlearn's preprocessing methods]
scaler_df = StandardScaler().fit_transform(X)
scaler_df = pd.DataFrame(X, columns = X.columns)

X = scaler_df
Y = data["Private"]

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, ran
dom_state=10)

model_svm = svm.LinearSVC()
model_svm.fit(x_train, y_train)

predicted_values = model_svm.predict(x_test)
metrics.accuracy_score(predicted_values, y_test)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: Convergence
Warning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[7]:
```
0.9294871794871795
```

In [8]:
```
# 5.Use scikit learn's gridsearch to select the best hyperparameter for a non-
linear SVM,identify the model with
# best score and its parameters.
# [Hint: Refer to model_selection module of Scikit learn]

parameter_candidates = [
    {'C': [1, 10, 100, 1000], 'kernel': ['poly']},
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]
```

In [9]:
```
parameter_candidates
```

Out[9]:
```
[{'C': [1, 10, 100, 1000], 'kernel': ['poly']},
 {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
 {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}]
```

In [10]:
```
# Create a classifier object with the classifier and parameter candidates
cv = ShuffleSplit()
clf = GridSearchCV(estimator=svm.SVC(max_iter=1000), param_grid=parameter_cand
idates, n_jobs=-1, cv=cv)
```

In [11]:
```
# Training the classifier
clf.fit(x_train, y_train)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\svm\_base.py:231: Convergence
Warning: Solver terminated early (max_iter=1000).  Consider pre-processing yo
ur data with StandardScaler or MinMaxScaler.
  % self.max_iter, ConvergenceWarning)
```

Out[11]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size=None,
train_size=None),
             error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=1000,
                           probability=False, random_state=None, shrinking=Tr
ue,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=-1,
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['poly']},
                         {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [16]:
```
clf.best_params_
```

Out[16]:
```
{'C': 10, 'kernel': 'poly'}
```

In [17]:
```
# View accuracy score
print("Best score for data>>>", clf.score(x_train, y_train))
```

```
Best score for data>>> 0.9371980676328503
```

In [18]:
```
# View the best parameters for the model found using grid search
print('Best C: ',clf.best_estimator_.C)
print('Best Kernel: ',clf.best_estimator_.kernel)
print('Best Gamma: ',clf.best_estimator_.gamma)
```

```
Best C:  10
Best Kernel:  poly
Best Gamma:  scale
```

In [ ]: