In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,cross_validate,cross_validate, classification_report
from sklearn.metrics.cluster.unsupervised import silhouette_score,sil
```

In [2]:
```
df = pd.read_csv('D:\ZA - Machine_Learning\Download Files\titanic_train.csv')
df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

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

Out[5]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

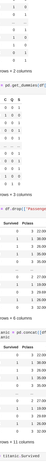
In [6]:
```
df.info()
```

Out[6]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Gender       891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7 KB
```

In [7]:
```
sns.heatmap(df.isnull())
```

Out[7]:
```
<AxesSubplot:>
```

In [10]:
```
plt.scatter(df.Pclass, df.Survived, marker = '+', color = 'red');
```



In [10]:
```
percnan = pd.DataFrame([col, df[col].isna().mean()*100] for col in df), columns = ['Features', 'Percentage'])
percnan
```

Out[10]:

	Features	Percentage
0	PassengerId	0.000000
1	Survived	0.000000
2	Pclass	0.000000
3	Name	0.000000
4	Gender	0.000000
5	Age	19.865320
6	SibSp	0.000000
7	Parch	0.000000
8	Ticket	0.000000
9	Fare	0.000000
10	Cabin	77.104377
11	Embarked	0.224467

In [11]:
```
# Age Cabin & Embarked features contain NaN values
# Age contains
```

In [12]:
```
df['Age'] = df['Age'].fillna(df['Age'].mean())
df
```

Out[12]:
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_10108\2.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-copy
```

In [12]:
```
df['Embarked'].mode()
```

Out[12]:
```
0    S
Name: Embarked, dtype: object
```

In [13]:
```
df['Embarked'] = df['Embarked'].fillna('S')
df
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	Q

891 rows × 11 columns

In [14]:
```
gd = pd.get_dummies(df['Gender'])
gd
```

Out[14]:

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

891 rows × 2 columns

In [15]:
```
em = pd.get_dummies(df['Embarked'])
em
```

Out[15]:

	C	Q	S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
...	...	...	...
886	0	0	1
887	0	0	1
888	0	0	1
889	1	0	0
890	0	1	0

891 rows × 3 columns

In [16]:
```
df = df.drop(['PassengerId', 'Name', 'Gender', 'Ticket', 'Embarked'],axis=1)
df
```

Out[16]:

	Survived	Pclass	Age	SibSp	Parch	Fare
0	0	3	22.00000	1	0	7.2500
1	1	1	38.00000	1	0	71.2833
2	1	3	26.00000	0	0	7.9250
3	1	1	35.00000	1	0	53.1000
4	0	3	35.00000	0	0	8.0500
...	...	...	...	...	...	...
886	0	2	27.00000	0	0	13.0000
887	1	1	19.00000	0	0	30.0000
888	0	3	29.69912	1	2	23.4500
889	1	1	26.00000	0	0	30.0000
890	0	3	32.00000	0	0	7.7500

891 rows × 6 columns

In [16]:
```
titanic = pd.concat([df, gd, em], axis=1)
titanic
```

Out[16]:

	Survived	Pclass	Age	SibSp	Parch	Fare	female	male	C	Q	S
0	0	3	22.00000	1	0	7.2500	0	1	0	0	1
1	1	1	38.00000	1	0	71.2833	1	0	1	0	0
2	1	3	26.00000	0	0	7.9250	1	0	0	0	1
3	1	1	35.00000	1	0	53.1000	1	0	0	0	1
4	0	3	35.00000	0	0	8.0500	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	27.00000	0	0	13.0000	0	1	0	0	1
887	1	1	19.00000	0	0	30.0000	1	0	0	0	1
888	0	3	29.69912	1	2	23.4500	1	0	0	0	1
889	1	1	26.00000	0	0	30.0000	0	1	1	0	0
890	0	3	32.00000	0	0	7.7500	0	1	0	1	0

891 rows × 11 columns

In [17]:
```
Y = titanic.Survived
```

In [17]:
```
X = titanic.drop(['Survived',],axis=1)
X
```

Out[17]:

	Pclass	Age	SibSp	Parch	Fare	female	male	C	Q	S
0	3	22.00000	1	0	7.2500	0	1	0	0	1
1	1	38.00000	1	0	71.2833	1	0	1	0	0
2	3	26.00000	0	0	7.9250	1	0	0	0	1
3	1	35.00000	1	0	53.1000	1	0	0	0	1
4	3	35.00000	0	0	8.0500	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...
886	2	27.00000	0	0	13.0000	0	1	0	0	1
887	1	19.00000	0	0	30.0000	1	0	0	0	1
888	3	29.69912	1	2	23.4500	1	0	0	0	1
889	1	26.00000	0	0	30.0000	0	1	1	0	0
890	3	32.00000	0	0	7.7500	0	1	0	1	0

891 rows × 10 columns

In [18]:
```
from sklearn import preprocessing
# get column names first
names = X.columns
# Creating scaler object
scaler = preprocessing.StandardScaler()
# Fitting and transforming
scaled_df = scaler.fit_transform(X)
X = pd.DataFrame(scaled_df, columns=names)
```

In [19]:
```
X
```

In [20]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=41)
```

In [21]:
```
LR = LogisticRegression()
LR.fit(X_train,Y_train)
```

Out[21]:
```
LogisticRegression()
```

In [22]:
```
Y_pred = LR.predict(X_test)
Y_pred
```

Out[22]:
```
array([0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0], dtype=int64)
```

In [24]:
```
Y_test
```

Out[24]:
```
454    0
624    0
537    1
493    1
206    0
      ..
586    1
524    0
671    0
432    1
304    0
Name: Survived, Length: 179, dtype: int64
```

In [24]:
```
LR.score(X_test, Y_test)
```

Out[24]:
```
0.8380586924217877
```

In [27]:
```
cf = confusion_matrix(Y_test, Y_pred)
cf
```

Out[27]:
```
array([93, 12],
       [21, 53], dtype=int64)
```

In [28]:
```
sns.heatmap(cf, annot = True)
plt.axis('equal')
plt.show()
```

In [31]:
```
# CLASSIFICATION REPORT
print(classification_report(Y_test, Y_pred))
```

Out[31]:
```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       105
           1       0.82      0.72      0.76        74

    accuracy                           0.82       179
   macro avg       0.82      0.80      0.81       179
weighted avg       0.82      0.82      0.81       179
```

In [32]:
```
# CONFUSION MATRIX
print(confusion_matrix(Y_test, Y_pred))
```

Out[32]:
```
[[93 12]
 [21 53]]
```

In [33]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(7,4))
    sns.set(font_scale=2)
    sns.heatmap(cf, annot = True, square=True, annot_kws={'size':20}, linewidth=3, cmap='cubehelix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.xticks([0,1])
    plt.title('Confusion Matrix with Sklearn and Seaborn', fontweight='bold', fontsize=20)
    plt.show()
```