In [ ]:

```
#1. Apply logistic regression.
```

In [1]:

```
import pandas as pd
df=pd.read_csv('titanic.csv')
df.head()
```

Out[1]:

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

In [2]:

```
df1=df.dropna(axis=0,how='any',thresh=None,inplace=False)
df1.isnull().sum()
```

Out[2]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [3]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  183 non-null    int64  
 1   Survived     183 non-null    int64  
 2   Pclass       183 non-null    int64  
 3   Name         183 non-null    object 
 4   Sex          183 non-null    object 
 5   Age          183 non-null    float64
 6   SibSp        183 non-null    int64  
 7   Parch        183 non-null    int64  
 8   Ticket       183 non-null    object 
 9   Fare         183 non-null    float64
 10  Cabin        183 non-null    object 
 11  Embarked     183 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 18.6+ KB
```

In [4]:

```
print(df1["Survived"].value_counts())
```

```
1    123
0     60
Name: Survived, dtype: int64
```

In [5]:

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df1['Sex']=encoder.fit_transform(df1['Sex'])
df1['Age']=encoder.fit_transform(df1['Age'])
df1['Cabin']=encoder.fit_transform(df1['Cabin'])
df1.head()
```

```
<ipython-input-5-924684a41234>:3: 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-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1['Sex']=encoder.fit_transform(df1['Sex'])
<ipython-input-5-924684a41234>:4: 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-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1['Age']=encoder.fit_transform(df1['Age'])
<ipython-input-5-924684a41234>:5: 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-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1['Cabin']=encoder.fit_transform(df1['Cabin'])
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cab
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	32	1	0	PC 17599	71.2833	7
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	28	1	0	113803	53.1000	4
6	7	0	1	McCarthy, Mr. Timothy J	1	49	0	0	17463	51.8625	1
10	11	1	3	Sandstrom, Miss. Marguerite Rut	0	4	1	1	PP 9549	16.7000	13
11	12	1	1	Bonnell, Miss. Elizabeth	0	53	0	0	113783	26.5500	4

In [6]:

```
#Logistic Regression Code
#import relevant libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

#features extraction
x = df1[['Sex', 'Age','Cabin','Pclass']]
y = df1['Survived']

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)  #split

logreg = LogisticRegression()   #build our logistic model
logreg.fit(x_train, y_train)  #fitting training data
y_pred  = logreg.predict(x_test)    #testing model's performance
print("Accuracy={:.2f}".format(logreg.score(x_test, y_test)))
```

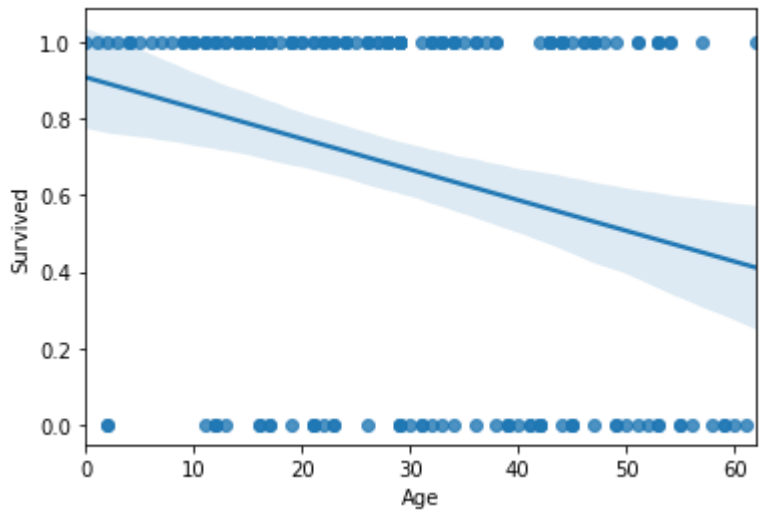
```
Accuracy=0.85
```

In [7]:

```
#Logistic Regression Distribution
import seaborn as sns
sns.regplot(x='Age',y='Survived',data=df1)
```

Out[7]:

```
<AxesSubplot:xlabel='Age', ylabel='Survived'>
```

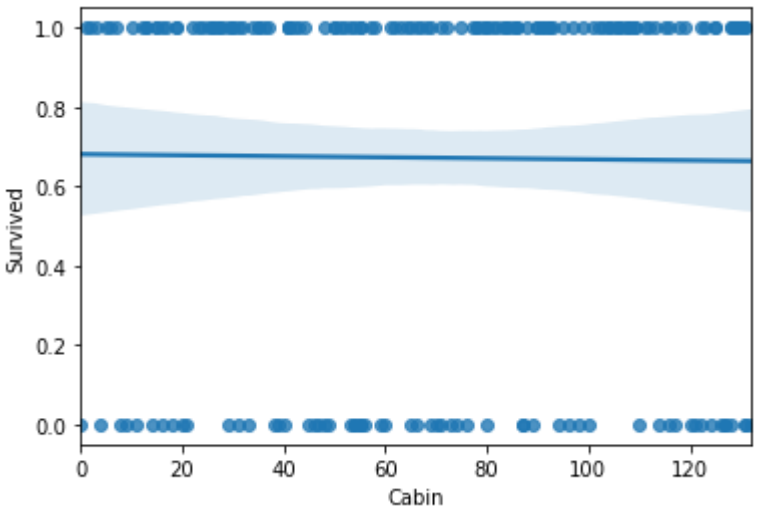


In [8]:

```
#Logistic Regression Distribution
import seaborn as sns
sns.regplot(x='Cabin',y='Survived',data=df1)
```

Out[8]:

```
<AxesSubplot:xlabel='Cabin', ylabel='Survived'>
```

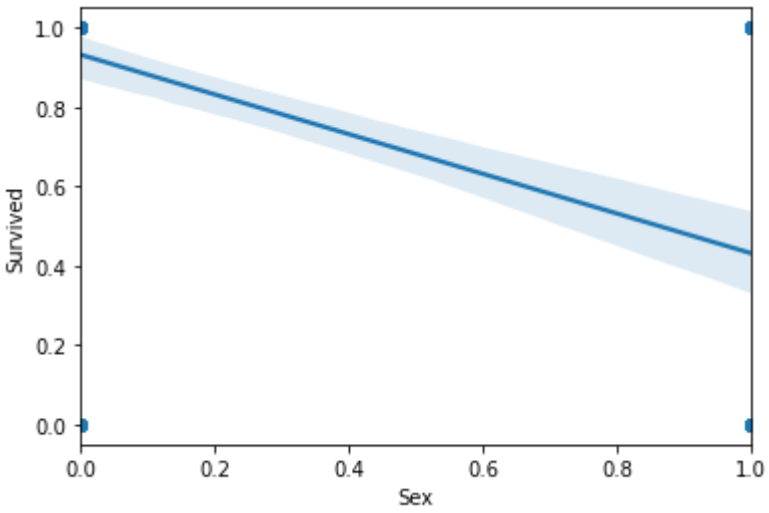


In [9]:

```
#Logistic Regression Distribution
import seaborn as sns
sns.regplot(x='Sex',y='Survived',data=df1)
```

Out[9]:

```
<AxesSubplot:xlabel='Sex', ylabel='Survived'>
```

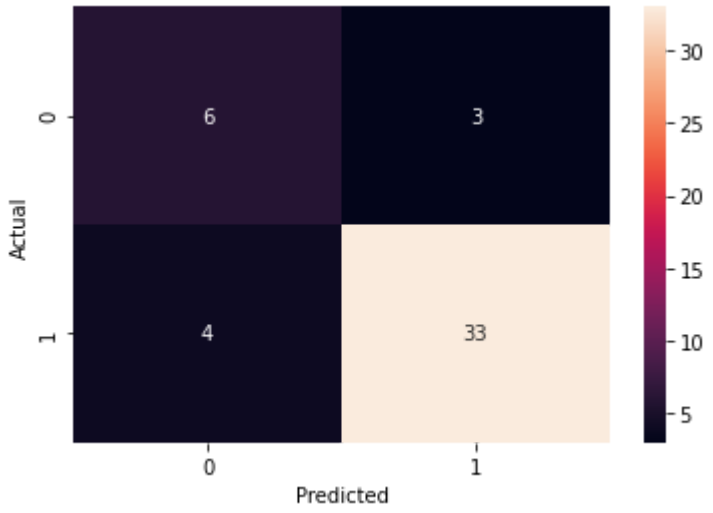


In [10]:

```
#2. Use confusion matrix to validate your model.
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sns.heatmap(confusion_matrix, annot=True)
```

Out[10]:

```
<AxesSubplot:xlabel='Predicted', ylabel='Actual'>
```



In [16]:

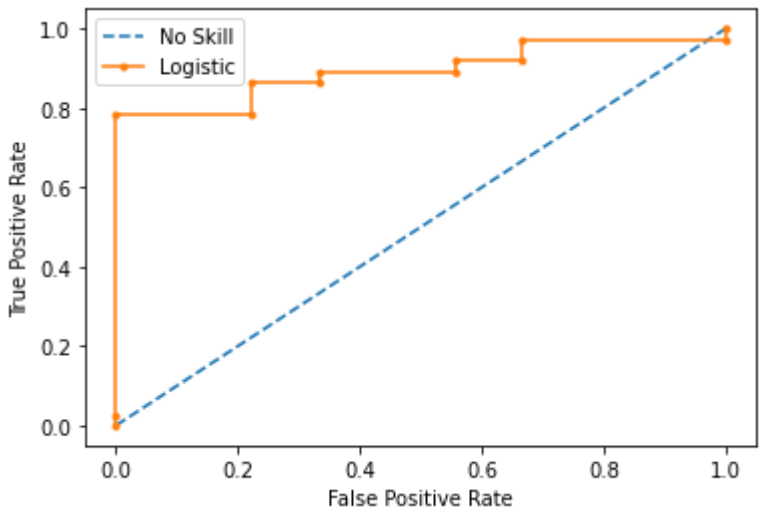
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
print (y_pred) #predicted values
```

```
              precision    recall  f1-score   support

           0       0.60      0.67      0.63         9
           1       0.92      0.89      0.90        37

    accuracy                           0.85        46
   macro avg       0.76      0.78      0.77        46
weighted avg       0.85      0.85      0.85        46

[1 0 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 0 1]
```

In [24]:

```
# 3. Another validation matrix for classification is ROC / AUC , do your research on them ex
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
#features extraction
x = df1[['Sex', 'Age','Cabin','Pclass']]
y = df1['Survived']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)  #splitt
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(y_test))]
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(x_train, y_train)
# predict probabilities
lr_probs = model.predict_proba(x_test)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.895
```



In [ ]: