In [1]:

```
#ionosphere
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df=pd.read_csv(r"C:\Users\samit\Downloads\archive.zip")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...
...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...

350 rows × 35 columns

In [3]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [4]:

```
print('This DataFrame has %d Rows and %d Columns'%(df.shape))
pd.set_option('display.width',95)
```

```
This DataFrame has 350 Rows and 35 Columns
```

In [5]:

```
1 df.head()
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.5087
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.7308
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.0000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.5279
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.0378

In [6]:

```
1 features_matrix=df.iloc[:,0:34]
```

Type *Markdown* and LaTeX: α^2

In [7]:

```
1 target_vector=df.iloc[:,-1]
```

In [8]:

```
1 print('The features matrix Has %d Rows and %d Column(s)'%(features_matrix.shape))
```

```
The features matrix Has 350 Rows and 34 Column(s)
```

In [9]:

```
1 print('The target matrix Has %d Rows and %d Column(s)'%(np.array(target_vector).res
```

```
The target matrix Has 350 Rows and 1 Column(s)
```

In [10]:

```
1 features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [11]:

```
1 algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=T
```

In [12]:

```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [13]:

```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,
              0.8339799999999999,-0.37708,1.0,0.0376,
              0.8524299999999999,-0.17755,0.59755,-0.44945,0.60536,
              -0.38223,0.8435600000000001,-0.38542,0.58212,-0.32192,
             0.56971,-0.29674,0.36946,-0.47357,0.56811,-0.51171,
             0.41070000000000003,-0.46160000000000003,0.21266,-0.3409,
             0.42267,-0.54487,0.18641,-0.453]]
```

In [14]:

```
predictions=Logistic_Regression_Model.predict(observation)
print('The Model Predicted The Observation To Belong To Class %s'%(predictions))
```

```
The Model Predicted The Observation To Belong To Class ['g']
```

In [15]:

```
print('The Algorithm Was Trained To Predict One Of The Two Classes:%s'%(algorithm.c
```

```
The Algorithm Was Trained To Predict One Of The Two Classes:['b' 'g']
```

In [16]:

```
print("""The Model Says The Probability Of The Observation We Passed Belonging To C
      (algorithm.predict_proba(observation)[0][0]))
print()
print("""The Model Says The Probability Of The Observation We Passed Belonging To C
      (algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The Observation We Passed Belonging To
Class['b'] is 3.317282690573631e-05

The Model Says The Probability Of The Observation We Passed Belonging To
Class['g'] is 0.9999668271730943
```

In [17]:

```
#2
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline
digits=load_digits()
```

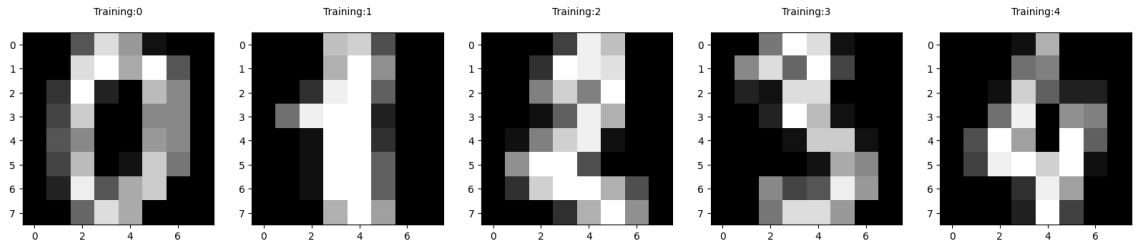
In [18]:

```
print("Image Data Shape",digits.data.shape)
print(" Label Data Shape",digits.target.shape)
```

```
Image Data Shape (1797, 64)
 Label Data Shape (1797,)
```

In [19]:

```
plt.figure(figsize=(20,4))
for index,(image,label)in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Training:%i\n'%label,fontsize=10)
```



In [20]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=
```

In [21]:

```
print(x_train.shape)
```

```
(1257, 64)
```

In [22]:

```
print(y_train.shape)
```

```
(1257,)
```

In [23]:

```
1 print(x_test.shape)
```

```
(540, 64)
```

In [24]:

```
1 print(y_test.shape)
```

```
(540,)
```

In [25]:

```
1 from sklearn.linear_model import LogisticRegression
2 logisticRegr=LogisticRegression(max_iter=1000)
3 logisticRegr.fit(x_train,y_train)
4 print(logisticRegr.predict(x_test))
```

```
[4 0 9 1 8 7 1 5 1 6 6 7 6 1 5 5 8 6 2 7 4 6 4 1 5 2 9 5 4 6 5 6 3 4 0 9
9
 8 4 6 8 8 5 7 9 8 9 6 1 7 0 1 9 7 3 3 1 8 8 8 9 8 5 8 4 9 3 5 8 4 3 1 3
8
 7 3 3 0 8 7 2 8 5 3 8 7 6 4 6 2 2 0 1 1 5 3 5 7 1 8 2 2 6 4 6 7 3 7 3 9
4
 7 0 3 5 4 5 0 3 9 2 7 3 2 0 8 1 9 2 1 5 1 0 3 4 3 0 8 3 2 2 7 3 1 6 7 2
8
 3 1 1 6 4 8 2 1 8 4 1 3 1 1 9 5 4 8 7 4 8 9 5 7 6 9 4 0 4 0 0 9 0 6 5 8
8
 3 7 9 2 0 8 2 7 3 0 2 1 9 2 7 0 6 9 3 1 1 3 5 2 5 5 2 1 2 9 4 6 5 5 5 9
7
 1 5 9 6 3 7 1 7 5 1 7 2 7 5 5 4 8 6 6 2 8 7 3 7 8 0 9 5 7 4 3 4 1 0 3 3
5
 4 1 3 1 2 5 1 4 0 3 1 5 5 7 4 0 1 0 9 5 5 5 4 0 1 8 6 2 1 1 1 7 9 6 7 9
7
 0 4 9 6 9 2 7 2 1 0 8 2 8 6 5 7 8 4 5 7 8 6 4 2 6 9 3 0 0 8 0 6 6 7 1 4
5
 6 9 7 2 8 5 1 2 4 1 8 8 7 6 0 8 0 6 1 5 7 8 0 4 1 4 5 9 2 2 3 9 1 3 9 3
2
 8 0 6 5 6 2 5 2 3 2 6 1 0 7 6 0 6 2 7 0 3 2 4 2 3 6 9 7 7 0 3 5 4 1 2 2
1
 2 7 7 0 4 9 8 5 6 1 6 5 2 0 8 2 4 3 3 2 9 3 8 9 9 5 9 0 3 4 7 9 8 5 7 5
0
 5 3 5 0 2 7 3 0 4 3 6 6 1 9 6 3 4 6 4 6 7 2 7 6 3 0 3 0 1 3 6 1 0 4 3 8
4
 3 3 4 8 6 9 6 3 3 0 5 7 8 9 1 5 3 2 5 1 7 6 0 6 9 5 2 4 4 7 2 0 5 6 2 0
8
 4 4 4 7 1 0 4 1 9 2 1 3 0 5 3 9 8 2 6 0 0 4]


C:\Users\samit\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

In [26]:

```
score=logisticRegr.score(x_test,y_test)
print(score)
```

```
0.9537037037037037
```

In [27]:

```
#gender submission
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [28]:

```
df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\gender_submission.csv")
df
```

Out[28]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
5	897	0
6	898	1
7	899	0
8	900	1
9	901	0

In [29]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [30]:

```
print('This DataFrame has %d Rows and %d Columns'%(df.shape))
pd.set_option('display.width',95)
```

```
This DataFrame has 418 Rows and 2 Columns
```

In [31]:

```
1 df.head()
```

Out[31]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [32]:

```
1 features_matrix=df.iloc[:,0:34]
```

In [33]:

```
1 target_vector=df.iloc[:,-1]
```

In [34]:

```
1 print('The features matrix Has %d Rows and %d Column(s)'%(features_matrix.shape))
```

```
The features matrix Has 418 Rows and 2 Column(s)
```

In [35]:

```
1 print('The target matrix Has %d Rows and %d Column(s)'%(np.array(target_vector).res
```

```
The target matrix Has 418 Rows and 1 Column(s)
```

In [36]:

```
1 features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [37]:

```
1 algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=T
```

In [38]:

```
1 Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [39]:

```
observation=[[1,0,]]
```

In [40]:

```
predictions=Logistic_Regression_Model.predict(observation)
print('The Model Predicted The Observation To Belong To Class %s'%(predictions))
```

```
The Model Predicted The Observation To Belong To Class [0]
```

In [41]:

```
print('The Algorithm Was Trained To Predict One Of The Two Classes:%s'%(algorithm.c
```

```
The Algorithm Was Trained To Predict One Of The Two Classes:[0 1]
```

In [42]:

```
print("""The Model Says The Probability Of The Observation We Passed Belonging To C
      (algorithm.predict_proba(observation)[0][0]))
print()
print("""The Model Says The Probability Of The Observation We Passed Belonging To C
      (algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The Observation We Passed Belonging To
Class['b'] is 0.9855942924591363

The Model Says The Probability Of The Observation We Passed Belonging To
Class['g'] is 0.014405707540863693
```

In [43]:

```
#train gender
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
#plt.rc("font",size=14)
import seaborn as sns
sns.set(style="white")#while background style for seaborn plots
sns.set(style="whitegrid",color_codes=True)

import warnings
warnings.simplefilter(action='ignore')
```

In [44]:

```
train_df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\train.gender_submiss
train_df
test_df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\train.gender_submissi
test_df
```

Out[44]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.0500

In [45]:

```
train_df.head()
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
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 [46]:

```
1 train_df.shape
```

Out[46]:

```
(891, 12)
```

In [47]:

```
1 test_df.head()
```

Out[47]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
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 [48]:

```
1 test_df.shape
```

Out[48]:

```
(891, 12)
```

In [49]:

```
1 train_df.describe()
```

Out[49]:

	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 [50]:

```
1 train_df.info()
```

```
<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   Sex          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 [51]:

```
1 test_df.describe
```

Out[51]:

```
<bound method NDFrame.describe of      PassengerId  Survived  Pclass
Name
0              1         0       3                            Braund,
Mr. Owen Harris  \
1              2         1       1  Cumings, Mrs. John Bradley (Floren
ce Briggs Th...
2              3         1       3                             Heikkin
en, Miss. Laina
3              4         1       1       Futrelle, Mrs. Jacques Heath
(Lily May Peel)
4              5         0       3                           Allen, M
r. William Henry
5              6         0       3                                   M
oran, Mr. James
6              7         0       1                            McCarth
y, Mr. Timothy J
7              8         0       3                     Palsson, Maste
```

In [52]:

```
1 test_df.info()
```

```
<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   Sex          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 [53]:

```
1 train_df.isnull().sum()
```

Out[53]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [54]:

```
1 test_df.isnull().sum()
```

Out[54]:

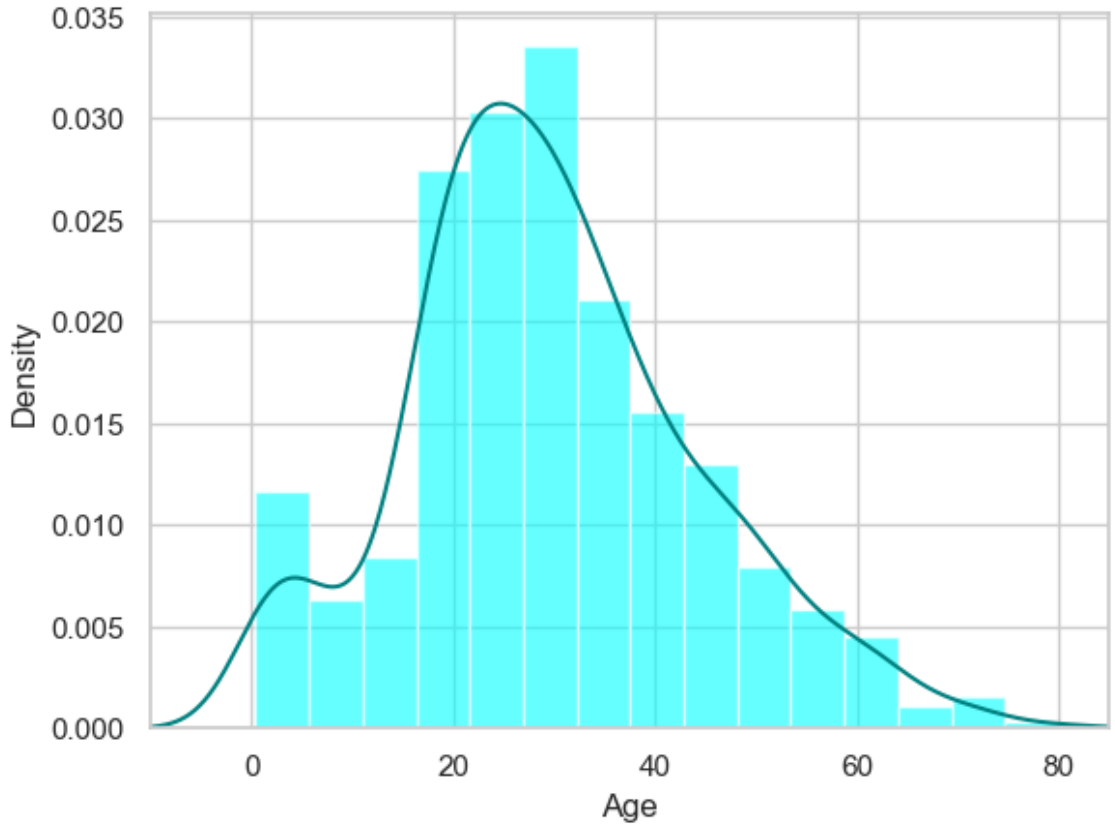
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [55]:

```
ax=train_df["Age"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
train_df["Age"].plot(kind='density',color='teal')
ax.set(xlabel='Age')
plt.xlim(-10,85)
```

Out[55]:

```
(-10.0, 85.0)
```



In [56]:

```
print(train_df["Age"].mean(skipna=True))
print(train_df["Age"].median(skipna=True))
```

```
29.69911764705882
28.0
```

In [57]:

```
print((train_df['Cabin'].isnull().sum()/train_df.shape[0])*100)
```

```
77.10437710437711
```

In [58]:

```
print((train_df['Embarked'].isnull().sum()/train_df.shape[0])*100)
```

```
0.22446689113355783
```

In [59]:

```
print('Boarded passengers grouped by port of embarkation(C=Cherbourg,Q=Queenstown,S=
```

```
Boarded passengers grouped by port of embarkation(C=Cherbourg,Q=Queenstow
n,S=Southampton):
```

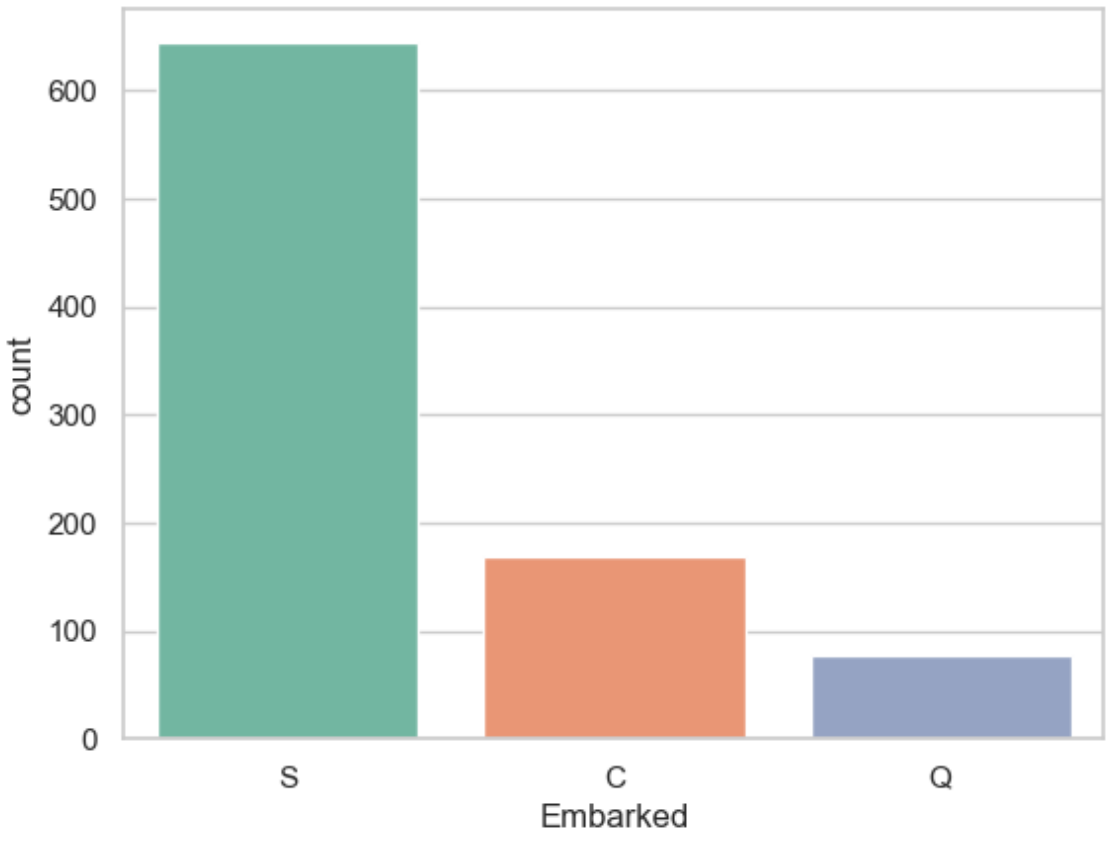
In [60]:

```
print(train_df['Embarked'].value_counts())
```

```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

In [61]:

```
sns.countplot(x='Embarked',data=train_df,palette='Set2')
plt.show()
```



In [62]:

```
print(train_df['Embarked'].value_counts().idxmax())
```

S

In [63]:

```
train_data=train_df.copy()
train_data["Age"].fillna(train_df["Age"].median(skipna=True),inplace=True)
train_data["Embarked"].fillna(train_df["Embarked"].value_counts().idxmax(),inplace=
train_data.drop('Cabin',axis=1,inplace=True)
```

In [64]:

```
train_data.isnull().sum()
```

Out[64]:

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

In [65]:

```
1 train_data.head()
```

Out[65]:

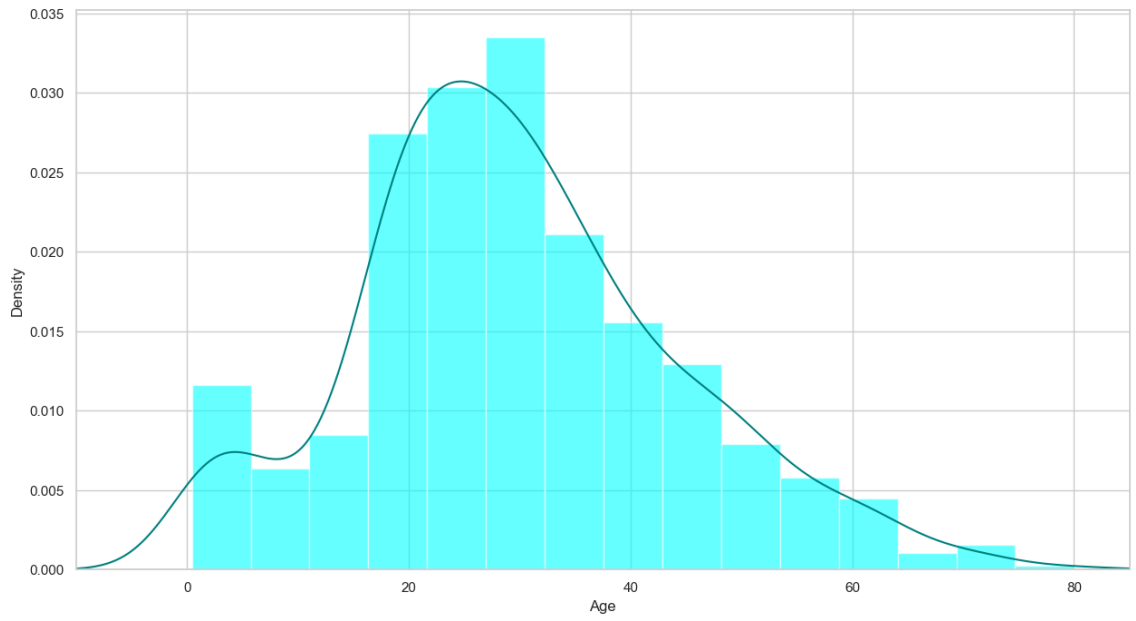
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, 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 [66]:

```
plt.figure(figsize=(15,8))
ax=train_df["Age"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
train_df["Age"].plot(kind='density',color='teal')
ax.set(xlabel='Age')
plt.xlim(-10,85)
plt.show
```

Out[66]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [68]:

```
train_data['TravelAlone']=np.where((train_data["SibSp"]+train_data["Parch"])>0,0,1)
train_data.drop('SibSp',axis=1,inplace=True)
train_data.drop('Parch',axis=1,inplace=True)
```

In [72]:

```
training=pd.get_dummies(train_data,columns=["Pclass","Embarked","Sex"])
training.drop('Sex_female',axis=1,inplace=True)
training.drop('PassengerId',axis=1,inplace=True)
training.drop('Name',axis=1,inplace=True)
training.drop('Ticket',axis=1,inplace=True)

final_train=training
final_train.head()
```

Out[72]:

	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarl
0	0	22.0	7.2500	0	False	False	True	False	
1	1	38.0	71.2833	0	True	False	False	True	
2	1	26.0	7.9250	1	False	False	True	False	
3	1	35.0	53.1000	0	True	False	False	False	
4	0	35.0	8.0500	1	False	False	True	False	

In [73]:

```
test_df.isnull().sum()
```

Out[73]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [76]:

```
test_data=test_df.copy()
test_data["Age"].fillna(train_df["Age"].median(skipna=True),inplace=True)
test_data["Fare"].fillna(train_df["Fare"].median(skipna=True),inplace=True)
test_data.drop('Cabin',axis=1,inplace=True)

test_data['TravelAlone']=np.where((test_data["SibSp"]+test_data["Parch"])>0,0,1)

test_data.drop('SibSp',axis=1,inplace=True)
test_data.drop('Parch',axis=1,inplace=True)

testing=pd.get_dummies(train_data,columns=["Pclass","Embarked","Sex"])
testing.drop('Sex_female',axis=1,inplace=True)
testing.drop('PassengerId',axis=1,inplace=True)
testing.drop('Name',axis=1,inplace=True)
testing.drop('Ticket',axis=1,inplace=True)
```

In [77]:

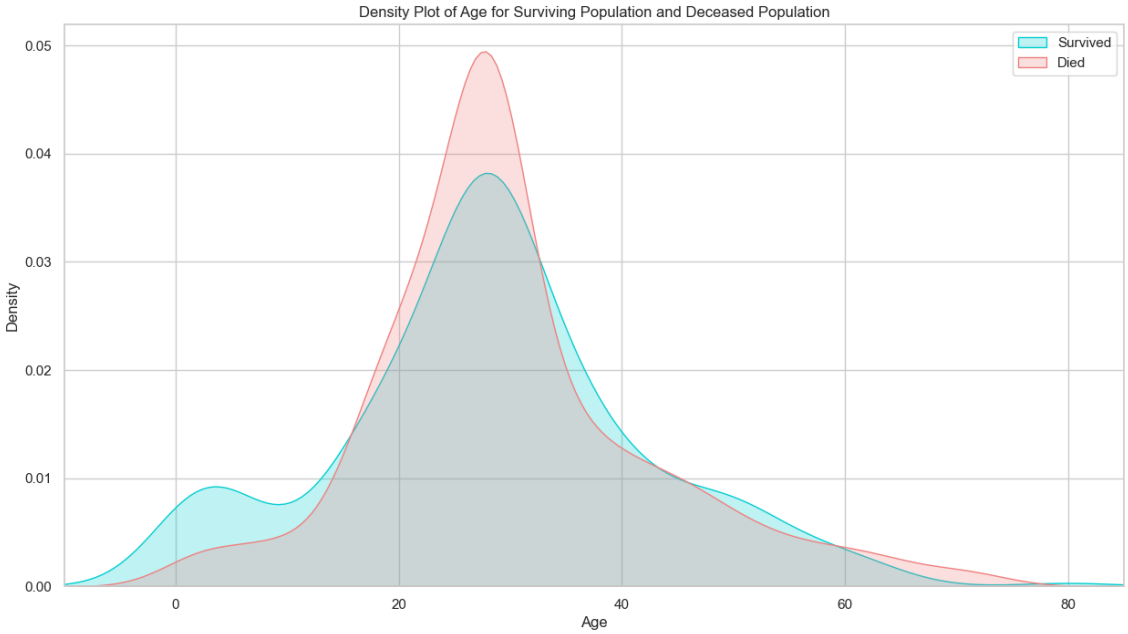
```
final_test=testing
final_test.head()
```

Out[77]:

	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarl
0	0	22.0	7.2500	0	False	False	True	False	
1	1	38.0	71.2833	0	True	False	False	True	
2	1	26.0	7.9250	1	False	False	True	False	
3	1	35.0	53.1000	0	True	False	False	False	
4	0	35.0	8.0500	1	False	False	True	False	

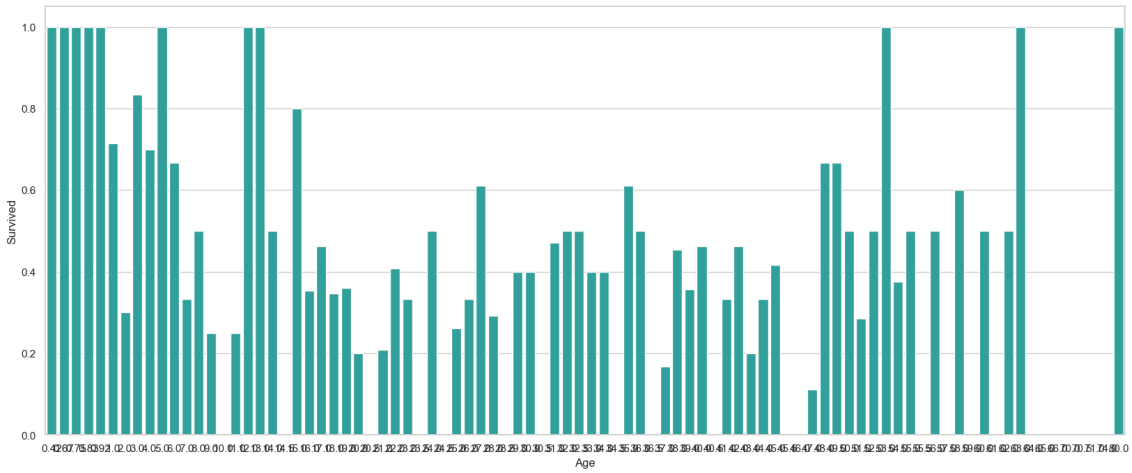
In [80]:

```
plt.figure(figsize=(15,8))
ax = sns.kdeplot(final_train["Age"][final_train.Survived == 1], color="darkturquois
sns.kdeplot(final_train["Age"][final_train.Survived == 0], color="lightcoral", shad
plt.legend(['Survived', 'Died'])
plt.title('Density Plot of Age for Surviving Population and Deceased Population')
ax.set(xlabel='Age')
plt.xlim(-10,85)
plt.show()
```



In [81]:

```
plt.figure(figsize=(20,8))
avg_survival_byage = final_train[["Age", "Survived"]].groupby(['Age'], as_index=Fal
g = sns.barplot(x='Age', y='Survived', data=avg_survival_byage, color="LightSeaGree
plt.show()
```



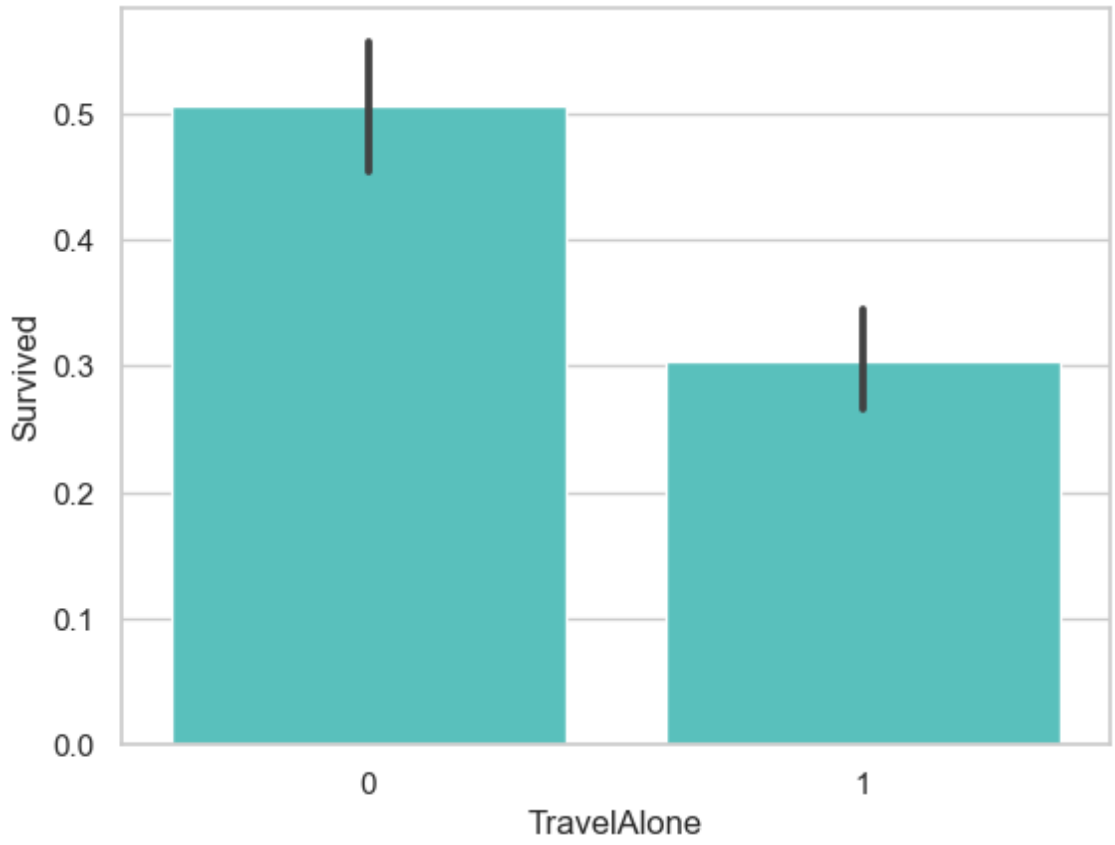
In [82]:

```
final_train['IsMinor']=np.where(final_train['Age']<=16, 1, 0)
print(final_train['IsMinor'])
```

```
16    1
17    0
18    0
19    0
20    0
21    0
22    1
23    0
24    1
25    0
26    0
27    0
28    0
29    0
30    0
31    0
32    0
33    0
34    0
35    0
```

In [83]:

```
sns.barplot(x='TravelAlone', y='Survived', data=final_train, color="mediumturquoise
plt.show()
```



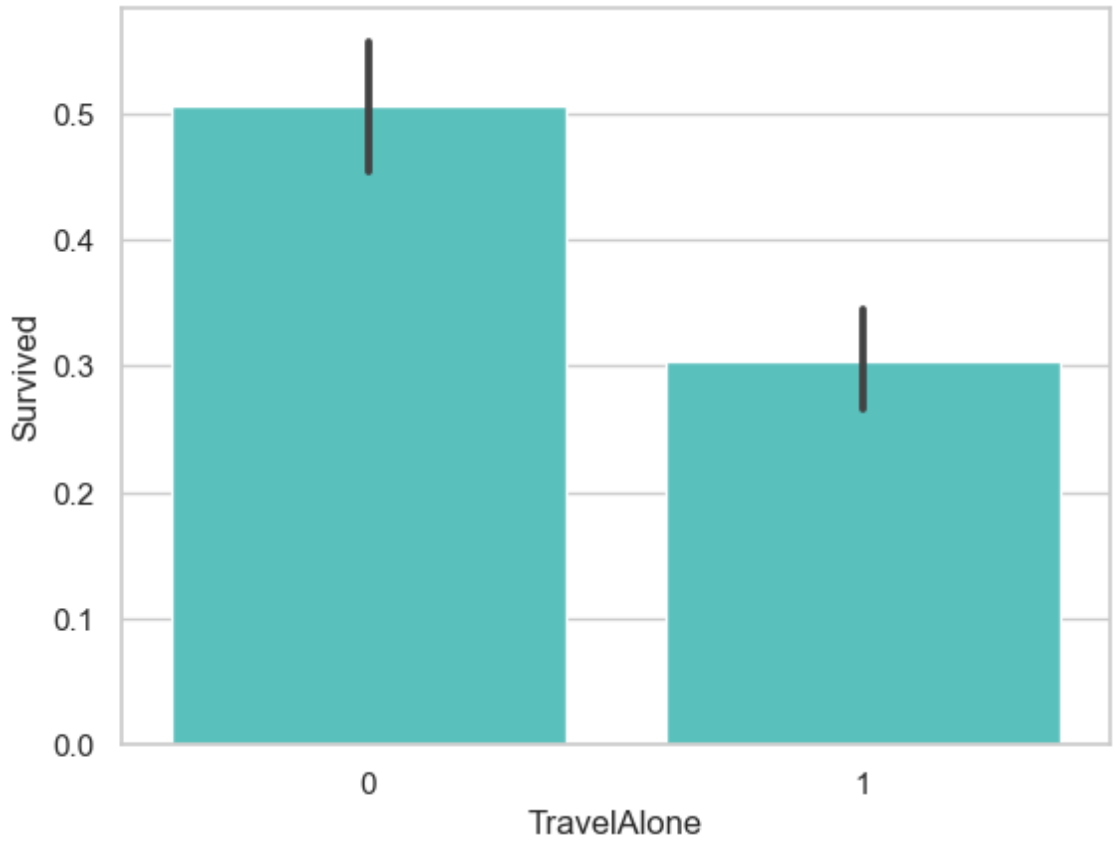
In [84]:

```
final_test['IsMinor']=np.where(final_test['Age']<=16, 1, 0)
print(final_test['IsMinor'])
```

```
802    1
803    1
804    0
805    0
806    0
807    0
808    0
809    0
810    0
811    0
812    0
813    1
814    0
815    0
816    0
817    0
818    0
819    1
820    0
821    0
```

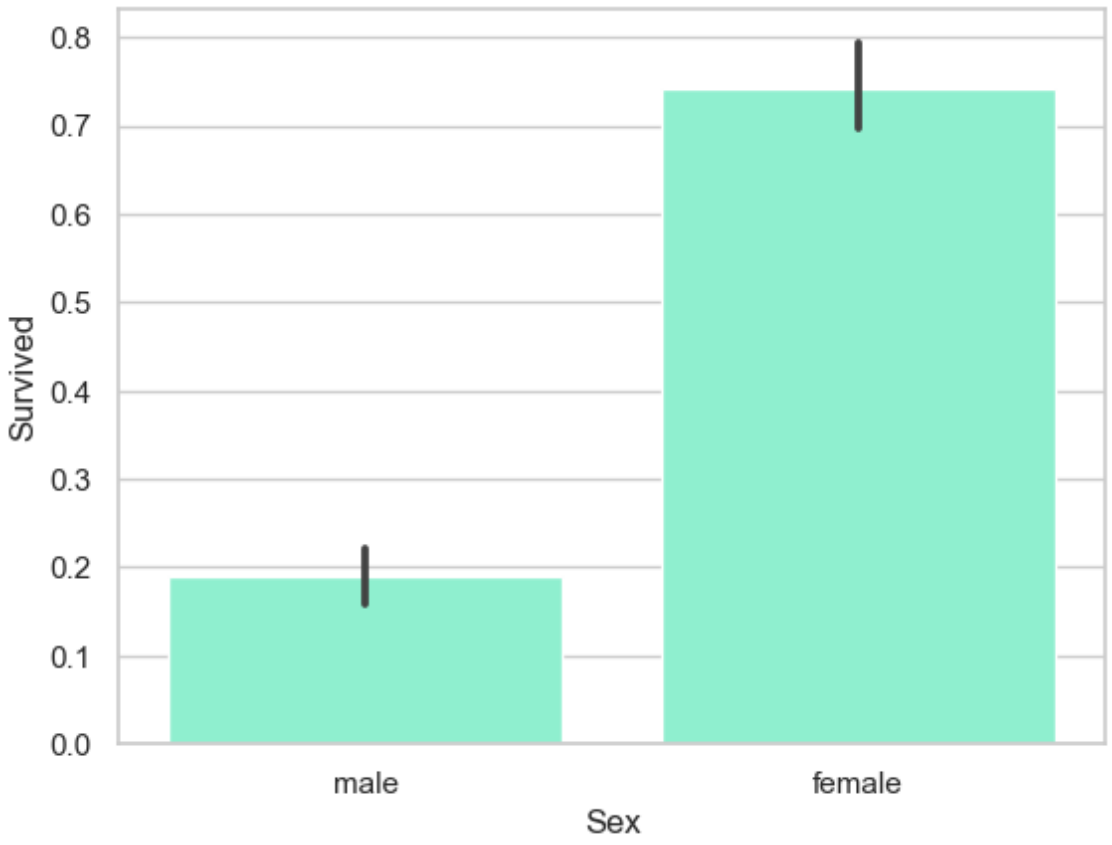
In [86]:

```
sns.barplot(x='TravelAlone', y='Survived', data=final_train, color="mediumturquoise
plt.show()
```



In [87]:

```
import seaborn as sns
import matplotlib.pyplot as plt
# Assuming 'train_df' is your DataFrame containing the data
sns.barplot(x='Sex', y='Survived', data=train_df, color='aquamarine')
plt.show()
```



In [ ]:

```
1
```