Major Project Synopsis

On

TITANIC SURVIVOR PREDICTIONS

In partial fulfillment of requirements for the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Anupreksha Shukla[20100BTCSDSI07260]
Anushka Sitoke[20100BTCSDSI07261]
Ashlesha Budholia[20100BTCSDSI07265]
Ekta Gupta [20100BTCSDSI07270]

Under the guidance of

Prof. Omkant Sharma

DEPARTMENT OF INFORMATION TECHNOLOGY
SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY
SHRI VAISHNAV VIDHYAPEETH VISHWAVIDYALAYA, INDORE

JULY-DEC- 2022

Abstract

Sociological transactions play an important role in human behavior and social standing. The Titanic was the perfect example as the passengers belonged to high income, middle-income, and low-income groups. It is interesting to see how social factors influenced who was going to survive. The data was collected from the website "Kaggle.com", and machine learning algorithms were applied after carrying out an exploratory and visual analysis. The hypothesis that women and children were saved (which became famous after Steven Spielberg's Titanic (1975)) was tested by random forest algorithm as well as the hypothesis that family density played a major role in survival. The results showed that title and sex were the most important factors influencing if the passenger was to survive.

1. Introduction

Using data provided by www.kaggle.com, our goal is to apply machine-learning techniques to successfully predict which passengers survived the sinking of the Titanic. Features like ticket price, age, sex, and class will be used to make the predictions.

We take several approaches to this problem in order to compare and contrast the different machine learning techniques. By looking at the results of each technique we can make some insights about the problem. The methods used in the project include Naïve Bayes, SVM, and decision tree. Using these methods, we try to predict the survival of passengers using different combinations of features.

The challenge boils down to a classification problem given a set of features. One way to make predictions would be to use Naïve Bayes [1]. Another would be to use SVM to map our features to a higher dimensional space. Our approach will be to first use Naïve Bayes as a baseline measure of what is achievable. Once this is complete, we use SVM [2] on our data to see if we can achieve better results. Lastly, we use decision tree analysis [3] and find the optimal decision boundaries.

2. Data Set

The data we used for our project was provided on the Kaggle website. We were given 891 passenger samples for our training set and their associated labels of whether the passenger survived. For each passenger, we were given his/her passenger class, name, sex, age, number of siblings/spouses aboard, number of parents/children aboard, ticket number, fare, cabin embarked, and port of embarkation. For the test data, we had 418 samples in the same format.

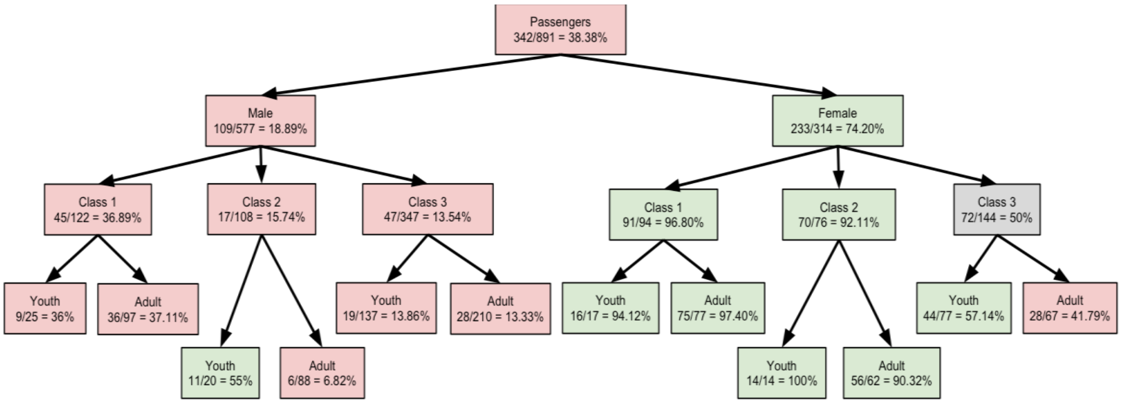


Figure 1. Data breakdown by sex, class and age. Percentages are percent that survived. Red boxes indicate mostly died, green boxes indicate mostly survived, and grey indicates 50% survival

2. Problem Domain

The ground-breaking contribution, the short, medium, and long-term consequences of disasters have been analyzed by economists. Psychologists and sociologists have stringently studied people’s behavior during disasters and rejected the notion that in the event of a disaster people become stunned, panicked, and unable to act rationally.

(RMS) Titanic was a British passenger liner operated by the white star line that sank in the North Atlantic Ocean on 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1500 died, making the sinking at the time one of the deadliest of the sinking ship and the deadliest peacetime sinking of a super-liner or cruise ship to date. With much public attention, the disaster has since been the material of many artistic works and a founding material of the disaster film genre.

Our main objective is to predict if any arbitrary passenger on Titanic would survive the sinking or not.

3. Solution Domain: -

We will use Machine Learning to predict the survival of Titanic passengers. In this problem, we will use real data from Titanic to calculate conditional probabilities and expectations and we're also going to use information from the titanic.csv dataset. The dataset for this problem is taken from: https://www.kaggle.com/c/titanic/data

We will create a model with the following steps:

- Download and explore the dataset
- Prepare the dataset for training
- Create a logistic regression model
- Train the model to fit the data
- Make predictions using the trained model

4. System Domain: -

To code, as we know we need a suitable environment, here in our case we have used Jupyter Notebook, as it reduces the hectic task of compiling and running the program on PC. We can use any editor as we like.
The foremost that we need to do is import the dependencies that we will be using in our code.

Importing dependencies:

- We will be using: NumPy, pandas, matplotlib, seaborn, sklearn. As we move ahead, you will get to know the use of each of these modules.

- Now, we need to upload the downloaded dataset, into this program, so that our code can read the data and perform the necessary actions using it.

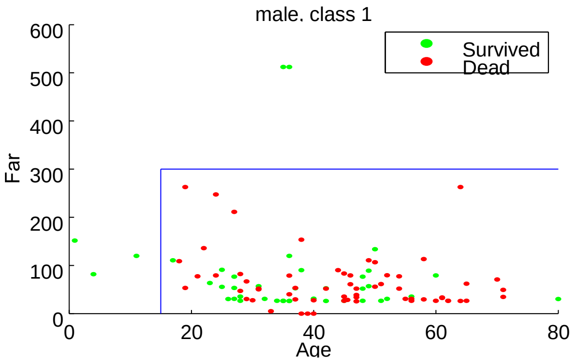
- As we have downloaded a CSV file, we shall be using Pandas to store that data in a variable. Our dataset is now stored in the variable named titanic_data. To get a brief idea about how the data is loaded, we use the command "variable_name. head()" to get a glimpse of the dataset in the form of a table.

- **Machine Learning Models**

Various machine learning models are implemented to validate and predict survivals.

1. **Logistic Regression: -** Logistic regression is the technique which works best when dependent variable is dichotomous (binary or categorical). The data description and explaining the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables is done with the help of logistic regression.
   It is used to solve binary classification problem, some of the real life examples are spam detection- predicting if an email is spam or not, health-Predicting if a given mass of tissue is benign or malignant, marketing- predicting if a given user will buy an insurance product or not.

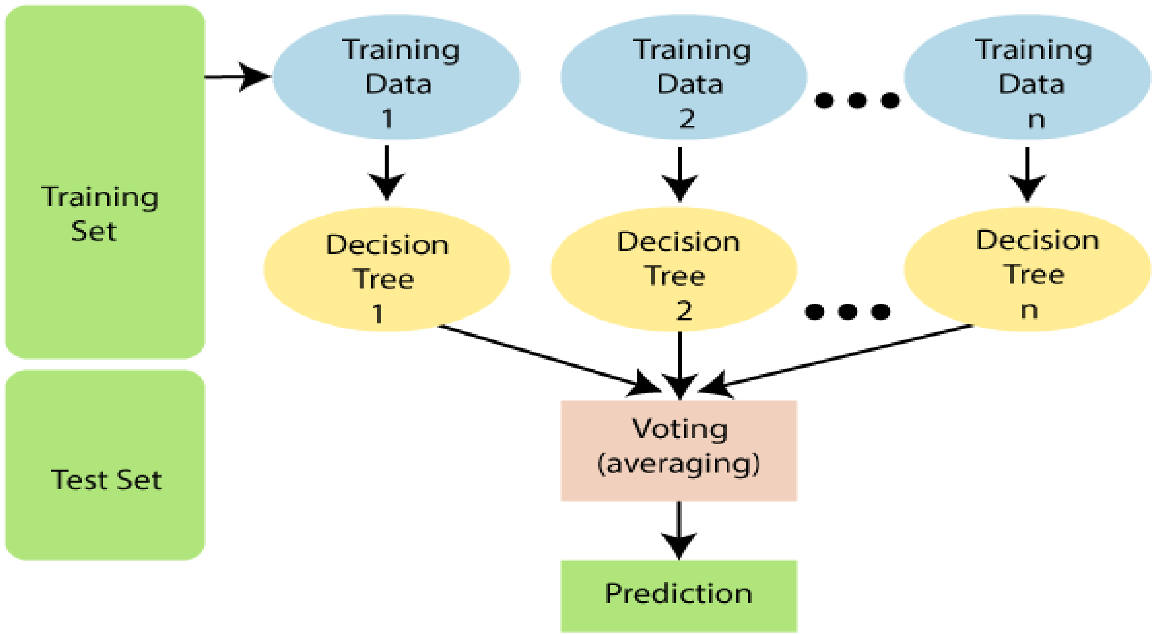
2. **Decision Tree: -** Decision tree is a supervised learning algorithm. This is generally used in problems based on classification. It is suitable for both categorical and continuous input and output variables.
   Each root node represents a single input variable (x) and a split point on that variable. The dependent variable (y) is present at leaf nodes. For example: Suppose there are two independent variables, i.e. input variables (x) which are height in centimeter and weight in kilograms and the task to find gender of person based on the given data. (Hypothetical example, for demonstration purpose only).



3. **K-Nearest neighbours: -** K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
   It assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories and stores all the available data and classifies a new data point based on the similarity.
   This means when new data appears then it can be easily classified into a

well suite category by using K- NN algorithm. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

4. **RandomForestClassifier:** - Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
   The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



5. Expected Outcomes: -

If the overall error rate falls below 20%, the model is better prepared to predict death (red line) than survival (green line).

Machine learning is a process that helps us approach a new stage in computing, and this is an abstraction. In this paper, two machine learning approaches were used to find the determinants that played a significant role in predicting passengers' survival. Since the variables present in the data set were related to the passengers' social classifications, the study's scope is sociological rather than technical. However, the algorithm provides us with solid evidence that title, sex, and fare were the top three variables that decided the fate of the passengers.

6. References: -

- **Kaggle Titanic Dataset**

- Beesley, L. (1912). The loss of the S.S.Titanic. New York: Dover Pub lications.

- Breiman. (2001). Random forests. Machine Learning, 45(1), 5-32.

- Bryceson, D. (1912). The Titanic disaster: British National Press. New York: W.W. Norton & Company Inc.

- Chen, Y., Sze, V. & Zhang, Z. (2017). Hardware for machine learn ing: Challenges and opportunities. 2017 IEEE Custom Integrat ed Circuits Conference, 1-8.

SHRI VAISHNAV VIDHYAPEETH VISHWAVIDHALAYA

DEPARTMENT OF INFORMATION TECHNOLOGY

INTRODUCTION

TO

DATA SCIENCE

BTIBM505

CODE FILE

Submitted To

Prof. Omkant Sharma Sir

Submitted By

Anupreksha Shukla [20100BTCSDSI07260]

Anushka Sitoke [20100BTCSDSI07261]

Ashlesha Budholia [20100BTCSDSI07265]

Ekta Gupta [20100BTCSDSI07270]

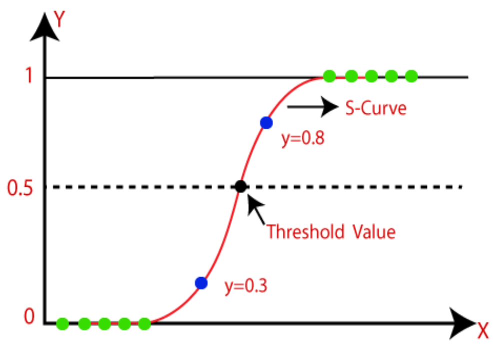
TITANIC SURVIVOR PREDICTIONS USING MACHINE LEARNING ALGORITHM

Our Team use Logistics Regression to predict the Titanic survivor

LOGISTICS REGRESSIONS:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.



PROGRAM CODE:

1. **Import Library**
   - import pandas as pd
   - import numpy as np
   - import matplotlib.pyplot as plt
   - import seaborn as sns

```
In [29]:  #Titanic Survival
          #Import Library

          import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns
```

2. **Read Dataset**
   - anupreksha_anushka_train_df = pd.read_csv("C:\\Users\\HP\\Downloads\\train.csv")
   - ashlesha_ekta_test_df = pd.read_csv("C:\\Users\\HP\\Downloads\\test.csv")

```
#Read the Dataset

anupreksha_anushka_train_df = pd.read_csv("C:\\Users\\HP\\Downloads\\train.csv")
ashlesha_ekta_test_df = pd.read_csv("C:\\Users\\HP\\Downloads\\test.csv")
```

3. **Analyse dataset**
   - #Analyzing by describing data
   - print(anupreksha_anushka_train_df.columns.values)

```
In [31]:  #Analyzing by describing data
          print(anupreksha_anushka_train_df.columns.values)

          ['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
           'Ticket' 'Fare' 'Cabin' 'Embarked']
```

print(ashlesha_ekta_test_df.columns.values)

```
In [32]:  print(ashlesha_ekta_test_df.columns.values)

          ['PassengerId' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch' 'Ticket' 'Fare'
           'Cabin' 'Embarked']
```

preview the data

anupreksha_anushka_train_df.head()

```
In [33]: # preview the data
         anupreksha_anushka_train_df.head()
Out[33]:
```

	PassengerId	Survived	Pclass	Name	Sex	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

anupreksha_anushka_train_df.tail()

```
In [34]: anupreksha_anushka_train_df.tail()
Out[34]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

anupreksha_anushka_train_df.info()

print('_'*40)

ashlesha_ekta_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
________________________________________
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
```

anupreksha_anushka_train_df.describe()

In [4]:
```
anupreksha_anushka_train_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

ashlesha_ekta_test_df.describe(include=['O'])

```
In [37]:  ashlesha_ekta_test_df.describe(include=['O'])
Out[37]:
```

	Name	Sex	Ticket	Cabin	Embarked
count	418	418	418	91	418
unique	418	2	363	76	3
top	Kelly, Mr. James	male	PC 17608	B57 B59 B63 B66	S
freq	1	266	5	3	270

#Analyze by pivoting features

anupreksha_anushka_train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)

```
In [39]:  ivoting features
ushka_train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)
Out[39]:
```

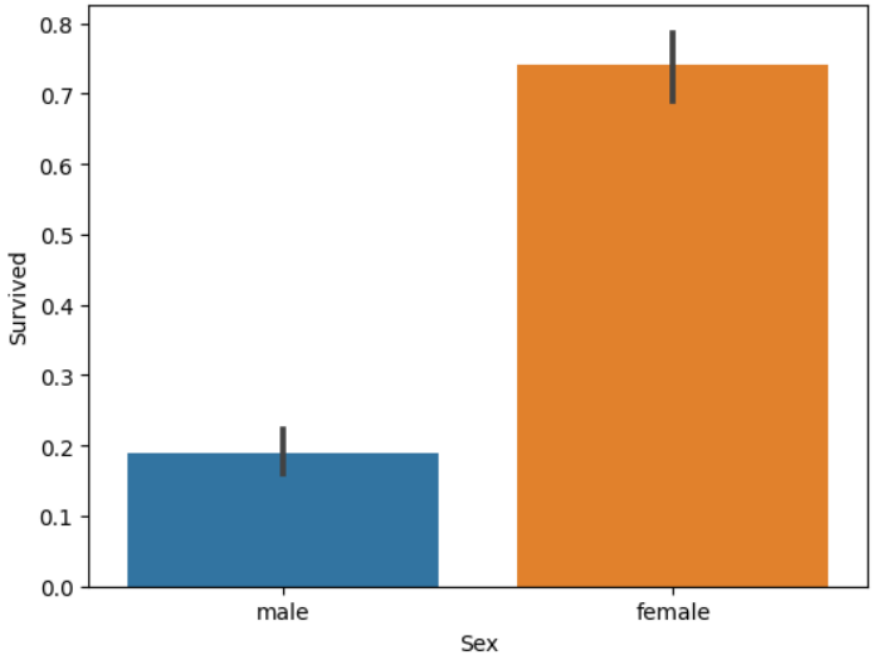
	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

anupreksha_anushka_train_df[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().sort_values(by='Survived', ascending=False)

	Sex	Survived
0	female	0.742038
1	male	0.188908

sns.barplot(x='Sex', y='Survived', data= anupreksha_anushka_train_df)

```
In [14]:  sns.barplot(x='Sex', y='Survived', data= anupreksha_anushka_train_df)
Out[14]: <AxesSubplot: xlabel='Sex', ylabel='Survived'>
```



anupreksha_anushka_train_df[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_values(by='Survived', ascending=False)

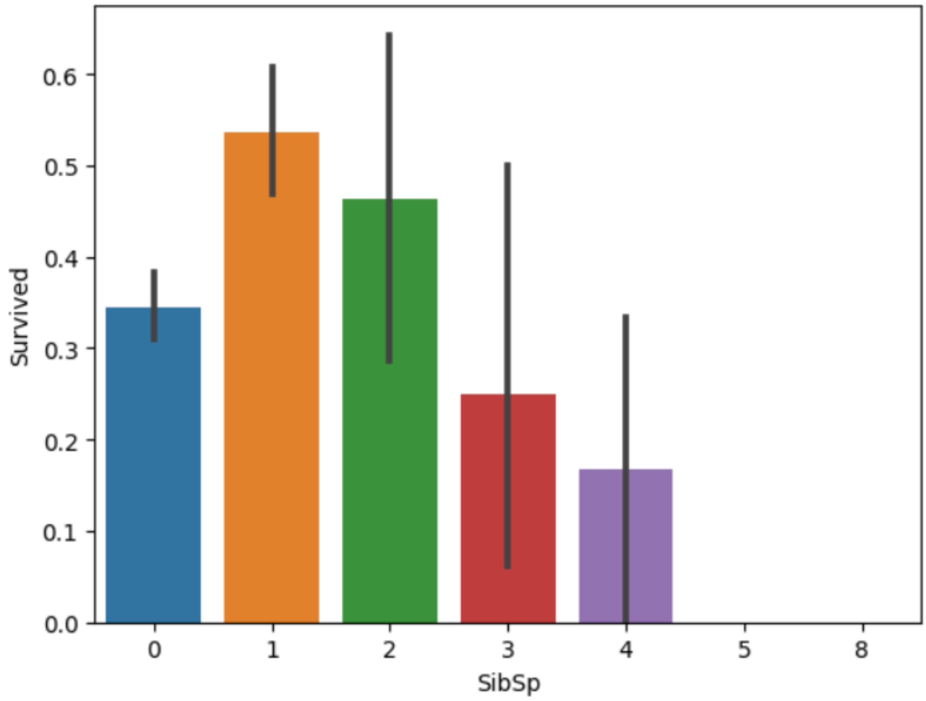
```
In [41]:  anushka_train_df[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[41]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

sns.barplot(x='SibSp', y='Survived', data= anupreksha_anushka_train_df)

```
In [15]:  sns.barplot(x='SibSp', y='Survived', data= anupreksha_anushka_train_df)
Out[15]: <AxesSubplot: xlabel='SibSp', ylabel='Survived'>
```



corr=ashlesha_ekta_test_df.corr()

corr

```
In [42]:  corr=ashlesha_ekta_test_df.corr()
          corr

C:\Users\HP\AppData\Local\Temp\ipykernel_8784\4251158441.py:1: FutureWarning: The
e.corr is deprecated. In a future version, it will default to False. Select only v
ic_only to silence this warning.
  corr=ashlesha_ekta_test_df.corr()

Out[42]:
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.026751	-0.034102	0.003818	0.043080	0.008211
Pclass	-0.026751	1.000000	-0.492143	0.001087	0.018721	-0.577147
Age	-0.034102	-0.492143	1.000000	-0.091587	-0.061249	0.337932
SibSp	0.003818	0.001087	-0.091587	1.000000	0.306895	0.171539
Parch	0.043080	0.018721	-0.061249	0.306895	1.000000	0.230046
Fare	0.008211	-0.577147	0.337932	0.171539	0.230046	1.000000

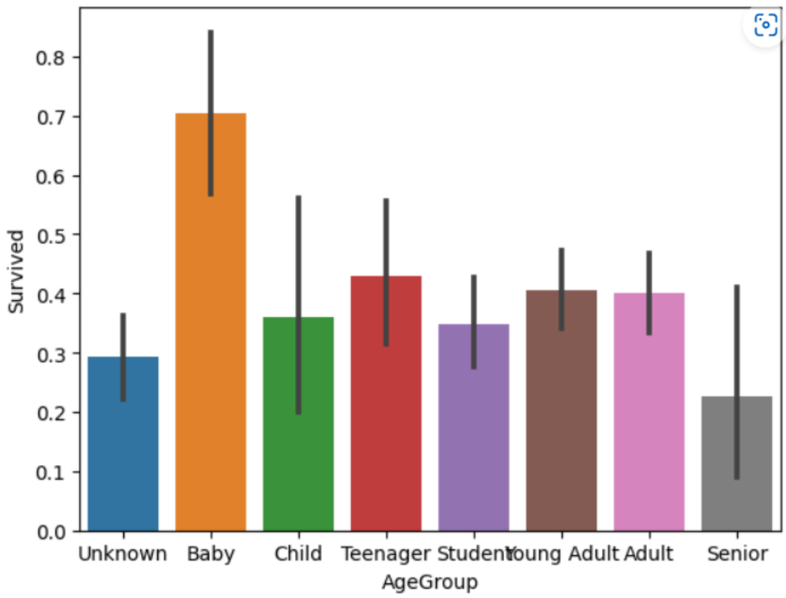
anupreksha_anushka_train_df=ashlesha_ekta_test_df

anupreksha_anushka_train_df.columns

```
In [43]:  anupreksha_anushka_train_df=ashlesha_ekta_test_df
          anupreksha_anushka_train_df.columns

Out[43]:  Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
                 'Ticket', 'Fare', 'Cabin', 'Embarked'],
                dtype='object')
```

sns.barplot(x='AgeGroup', y='Survived', data= anupreksha_anushka_train_df)

```
In [33]:  sns.barplot(x='AgeGroup', y='Survived', data= anupreksha_anushka_train_df   )

Out[33]:  <AxesSubplot: xlabel='AgeGroup', ylabel='Survived'>
```



4. **Classification**

   -In the case of Classification we need to change label/target output in to descrete from labelencoder

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

anupreksha_anushka_train_df['Embarked']=lb.fit_transform(anupreksha_anushka_train_df[‘Survived'])

```
In [9]:  #Classification
         from sklearn.preprocessing import LabelEncoder
         lb=LabelEncoder()
         anupreksha_anushka_train_df['Survived']=lb.fit_transform(anupreksha_anushka_train_df['Survived'])
```

5. **Defining depenedent and independent variable**

   X=anupreksha_anushka_train_df[['PassengerId','Pclass','SibSp','Parch']]
   Y=anupreksha_anushka_train_df['Embarked']

```
In [18]: X=anupreksha_anushka_train_df[['PassengerId','Pclass','SibSp','Parch']]
         Y=anupreksha_anushka_train_df['Embarked']
```

6. Spliting x and y in to train and test dataset

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25)

```
In [14]: #Spliting x and y in to train and test dataset
         from sklearn.model_selection import train_test_split
         X_train,X_test,Y_train,Y_test=train_test_split(X,Y, test_size =.25)
```

7. Import the model/Algorithm

from sklearn.linear_model import LogisticRegression

logreg=LogisticRegression()

```
In [20]: # Import the model/Algorithm
         from sklearn.linear_model import LogisticRegression
         logreg = LogisticRegression()
```

8. Train Model with x_train and y_train

logreg.fit(x_train,y_train)

```
In [21]: logreg.fit(X_train, Y_train)
```

9. Predict with x_train and y_train

```
In [22]: Y_pred_lr=logreg.predict(X_test)
         Y_pred_lr[:5],Y_test.values[:5]

Out[22]: (array([2, 2, 2, 2, 2]), array([2, 2, 2, 2, 2]))

In [23]: print(logreg.score(X_train,Y_train))
         print(logreg.score(X_test,Y_test))

         0.6987951807228916
         0.8072289156626506
```

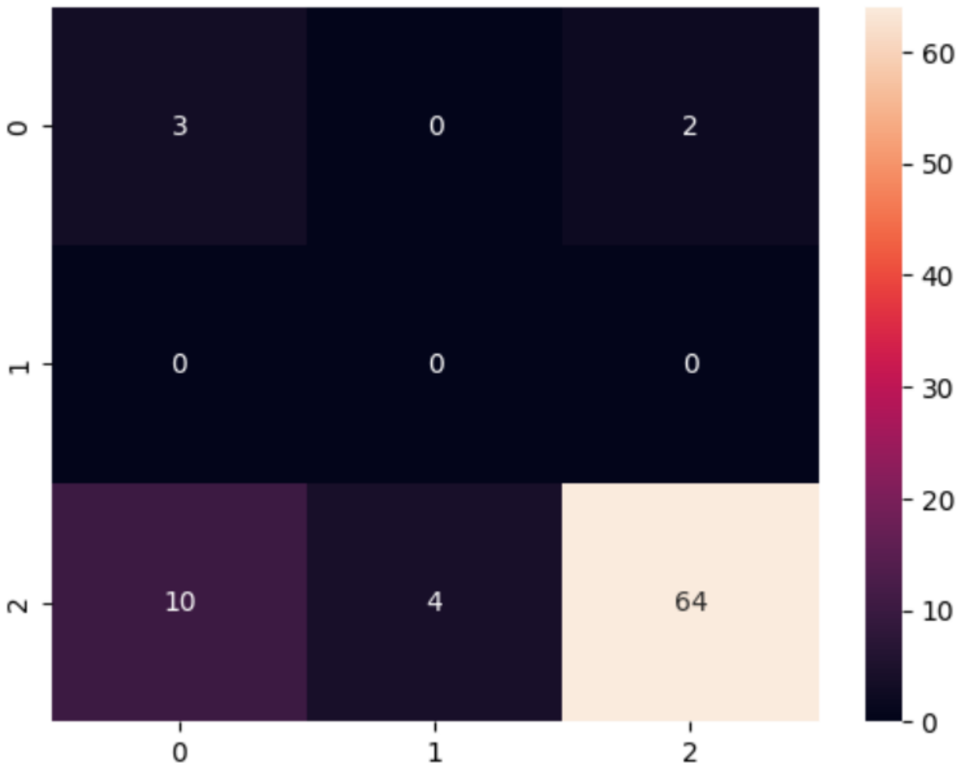
10. Evaluate using confusion matrix classification report and accuracy score

```
In [24]: from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
         print(confusion_matrix(Y_pred_lr,Y_test))
         print(classification_report(Y_pred_lr,Y_test))
         print(f'model_score-{logreg.score(X_test,Y_test)}')
         print(f'accuracy_score-{accuracy_score(Y_pred_lr,Y_test)}')
```

OUTPUT:

```
[[ 3  0  2]
 [ 0  0  0]
 [10  4 64]]
              precision    recall  f1-score   support

           0       0.23      0.60      0.33         5
           1       0.00      0.00      0.00         0
           2       0.97      0.82      0.89        78

    accuracy                           0.81        83
   macro avg       0.40      0.47      0.41        83
weighted avg       0.93      0.81      0.86        83

model_score-0.8072289156626506
accuracy_score-0.8072289156626506
```

```
In [25]: cm=confusion_matrix(Y_pred_lr,Y_test)
         sns.heatmap(cm,annot=True)
         plt.show()
```



DECISION TREE:

Program Code:

Step 1-6 will be same as depicted in above model.

1. Import the library
2. Read dataset
3. Analyze dataset
4. Classification
5. Defining dependent and independent variable
6. Splitting X and Y into train and test dataset
7. **Import model/ algorithm:**

```
In [20]: # Import the model/Algorithm
         from sklearn.tree import DecisionTreeClassifier
         dt = DecisionTreeClassifier()
```

8. **Train model with X_train and Y_train**

```
In [21]: dt.fit(X_train, Y_train)

Out[21]: ▾ DecisionTreeClassifier
         DecisionTreeClassifier()
```

9. **Predict X_train and Y_train**

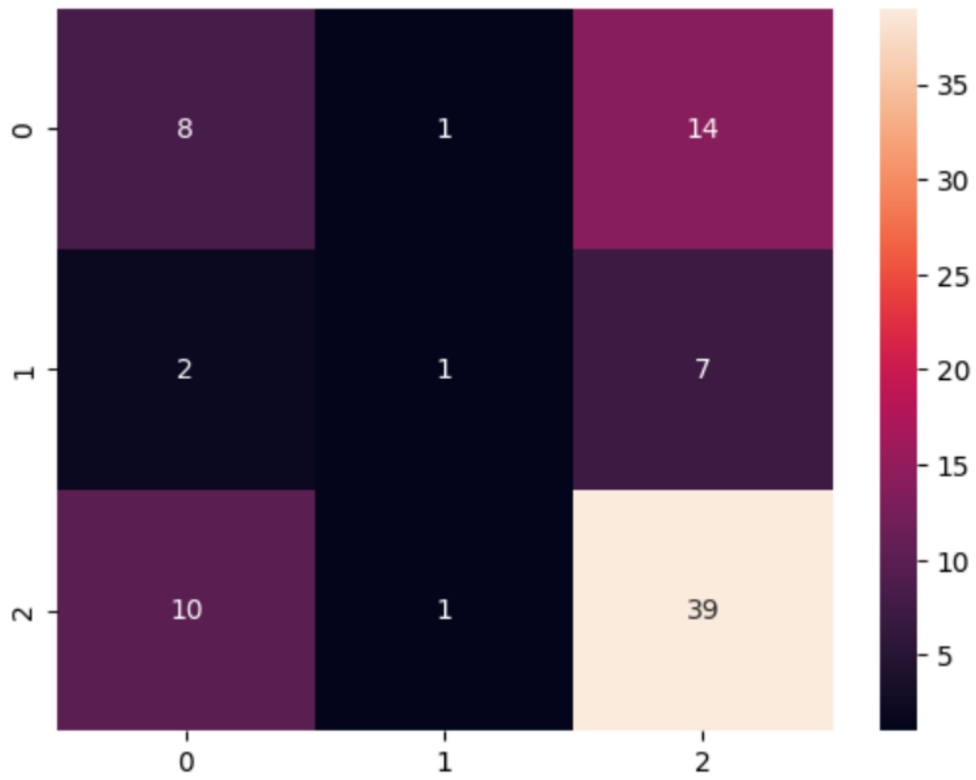
```
In [22]: Y_pred_dt=dt.predict(X_test)
         Y_pred_dt[:5],Y_test.values[:5]

Out[22]: (array([2, 2, 2, 0, 2]), array([2, 2, 2, 2, 2]))

In [23]: print(dt.score(X_train,Y_train))
         print(dt.score(X_test,Y_test))

         1.0
         0.5783132530120482
```

10. **Evaluate using confusion matrix report and accuracy score.**

In [24]:
```python
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(confusion_matrix(Y_pred_dt,Y_test))
print(classification_report(Y_pred_dt,Y_test))
print(f'model_score-{dt.score(X_test,Y_test)}')
print(f'accuracy_score-{accuracy_score(Y_pred_dt,Y_test)}')
```

```
[[ 8  1 14]
 [ 2  1  7]
 [10  1 39]]
              precision    recall  f1-score   support

           0       0.40      0.35      0.37        23
           1       0.33      0.10      0.15        10
           2       0.65      0.78      0.71        50

    accuracy                           0.58        83
   macro avg       0.46      0.41      0.41        83
weighted avg       0.54      0.58      0.55        83

model_score-0.5783132530120482
accuracy_score-0.5783132530120482
```

In [25]:
```python
cm=confusion_matrix(Y_pred_dt,Y_test)
sns.heatmap(cm,annot=True)
plt.show()
```



K-N NEIGHBOUR:

Program Code:

Step 1-6 will be same as depicted in above model.

1. Import the library
2. Read dataset
3. Analyze dataset
4. Classification
5. Defining dependent and independent variable
6. Splitting X and Y into train and test dataset
7. **Import model/ algorithm:**

```
In [20]: # Import the model/Algorithm
         from sklearn.neighbors import KNeighborsClassifier
         kn = KNeighborsClassifier()
```

8. **Train model with X_train and Y_train**

```
In [21]: kn.fit(X_train, Y_train)

Out[21]: ▾ KNeighborsClassifier
         KNeighborsClassifier()
```

9. **Predict X_train and Y_train**

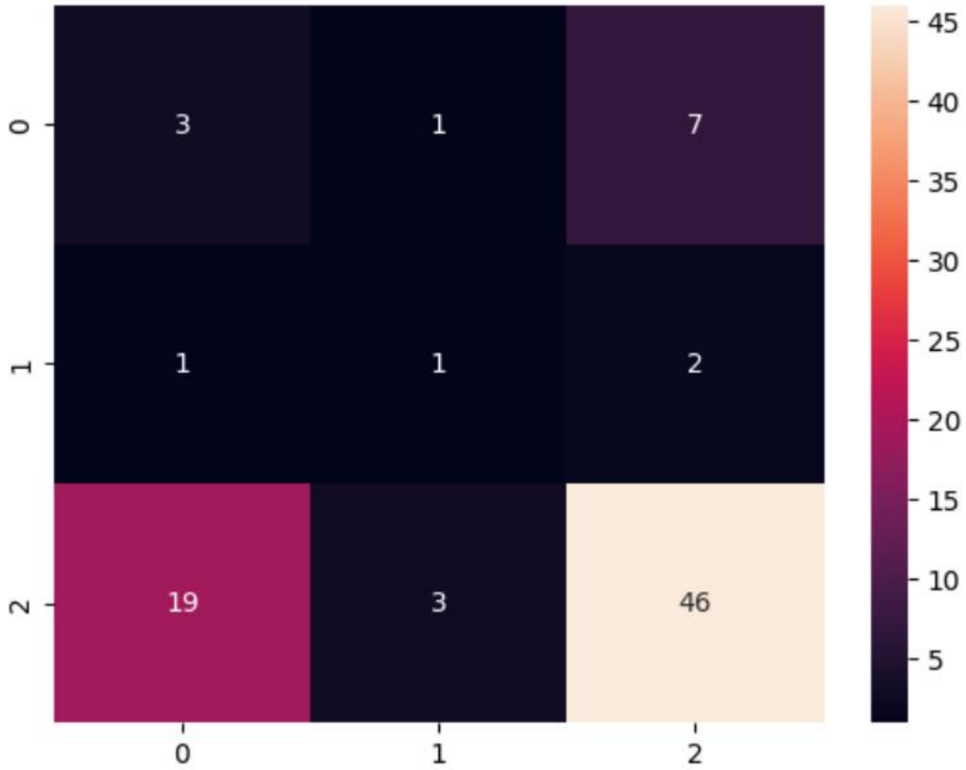
```
In [22]: Y_pred_kn=kn.predict(X_test)
         Y_pred_kn[:5],Y_test.values[:5]

Out[22]: (array([2, 2, 2, 0, 2]), array([1, 0, 2, 0, 2]))

In [23]: print(kn.score(X_train,Y_train))
         print(kn.score(X_test,Y_test))

         0.751004016064257
         0.6024096385542169
```

10. **Evaluate using confusion matrix report and accuracy score.**

In [24]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(confusion_matrix(Y_pred_kn,Y_test))
print(classification_report(Y_pred_kn,Y_test))
print(f'model_score-{kn.score(X_test,Y_test)}')
print(f'accuracy_score-{accuracy_score(Y_pred_kn,Y_test)}')
```

```
[[ 3  1  7]
 [ 1  1  2]
 [19  3 46]]
              precision    recall  f1-score   support

           0       0.13      0.27      0.18        11
           1       0.20      0.25      0.22         4
           2       0.84      0.68      0.75        68

    accuracy                           0.60        83
   macro avg       0.39      0.40      0.38        83
weighted avg       0.71      0.60      0.65        83

model_score-0.6024096385542169
accuracy_score-0.6024096385542169
```

In [25]:
```
cm=confusion_matrix(Y_pred_kn,Y_test)
sns.heatmap(cm,annot=True)
plt.show()
```



RANDOM FOREST:

Program Code:

Step 1-6 will be same as depicted in above model.

1. Import the library
2. Read dataset
3. Analyze dataset
4. Classification
5. Defining dependent and independent variable
6. Splitting X and Y into train and test dataset
7. **Import model/ algorithm:**

```
In [20]: # Import the model/Algorithm
         from sklearn.ensemble import RandomForestClassifier
         rf= RandomForestClassifier()
```

8. **Train model with X_train and Y_train**

```
In [21]: rf.fit(X_train, Y_train)

Out[21]: ▾ RandomForestClassifier
         RandomForestClassifier()
```

9. **Predict X_train and Y_train**

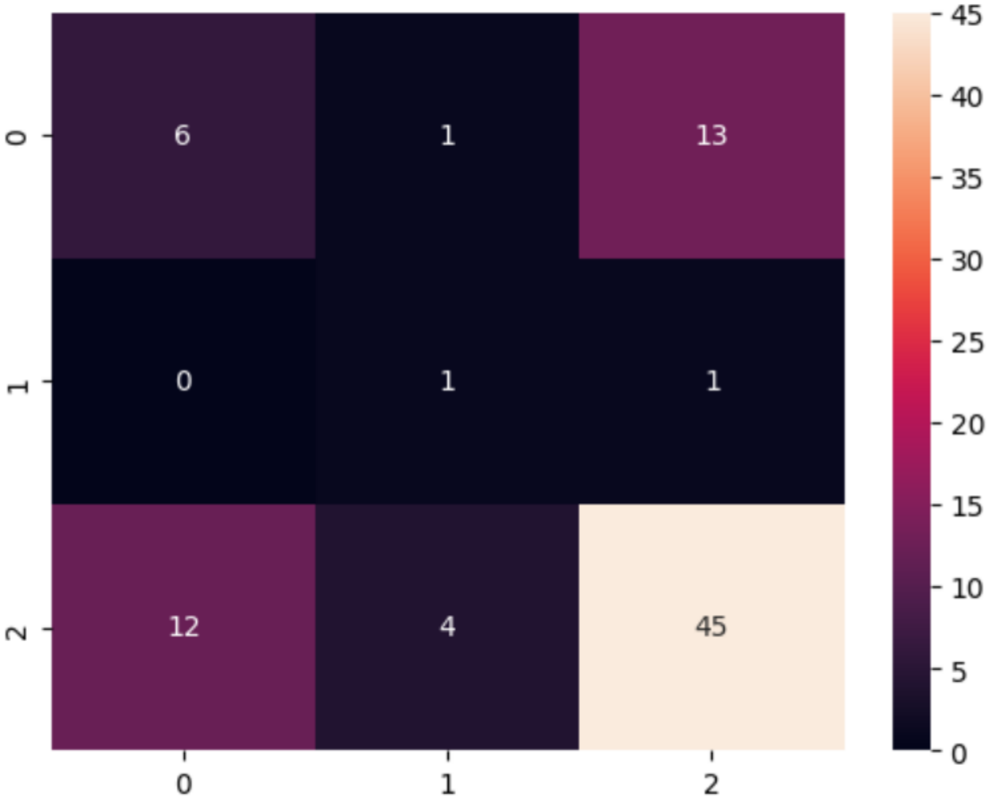
```
In [22]: Y_pred_rf=rf.predict(X_test)
         Y_pred_rf[:5],Y_test.values[:5]

Out[22]: (array([0, 2, 0, 2, 0]), array([2, 2, 2, 2, 2]))

In [23]: print(rf.score(X_train,Y_train))
         print(rf.score(X_test,Y_test))

         1.0
         0.6265060240963856
```

10. **Evaluate using confusion matrix report and accuracy score.**

In [24]:
```python
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(confusion_matrix(Y_pred_rf,Y_test))
print(classification_report(Y_pred_rf,Y_test))
print(f'model_score-{rf.score(X_test,Y_test)}')
print(f'accuracy_score-{accuracy_score(Y_pred_rf,Y_test)}')
```

```
[[ 6  1 13]
 [ 0  1  1]
 [12  4 45]]
              precision    recall  f1-score   support

           0       0.33      0.30      0.32        20
           1       0.17      0.50      0.25         2
           2       0.76      0.74      0.75        61

    accuracy                           0.63        83
   macro avg       0.42      0.51      0.44        83
weighted avg       0.64      0.63      0.63        83

model_score-0.6265060240963856
accuracy_score-0.6265060240963856
```

In [25]:
```python
cm=confusion_matrix(Y_pred_rf,Y_test)
sns.heatmap(cm,annot=True)
plt.show()
```



TOPIC:
TITANIC SURVIVOR PREDICTIONS USING LOGISTIC REGRESSION

GROUP PRESENTATION

Members:	Enrollment no:
Anupreksha Shukla	20100BTCSDSI07260
Anushka Sitoke	20100BTCSDSI07261
Ashlesha Budholia	20100BTCSDSI07265
Ekta Gupta	20100BTCSDSI07270


TITANIC SURVIVOR
PREDICTIONS

INTRODUCTION

- The Sinking of the RMS Titanic is one of the most infamous shipwrecks in history.
- In this Challenge, we ask you to complete the analysis of what sorts of people were likely to survive.
- In Particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy

GOALS AND OBJECTIVE

- The purpose of this project is to document the process I went through to create my predictions for Titanic Survivor Prediction.

- The Objective of this project was to build a classification model that could successfully determine whether a Titanic Passenger lived or died.

SOFTWARE REQUIRED

- TOOLS USED

  1. Jupyter Notebook

- LIBRARY USED

  1. Analyzing: Numpy, Pandas, Sci-kit Learn

  2. Visualization: Matplotlib, Seaborn

LOGISTIC REGRESSION

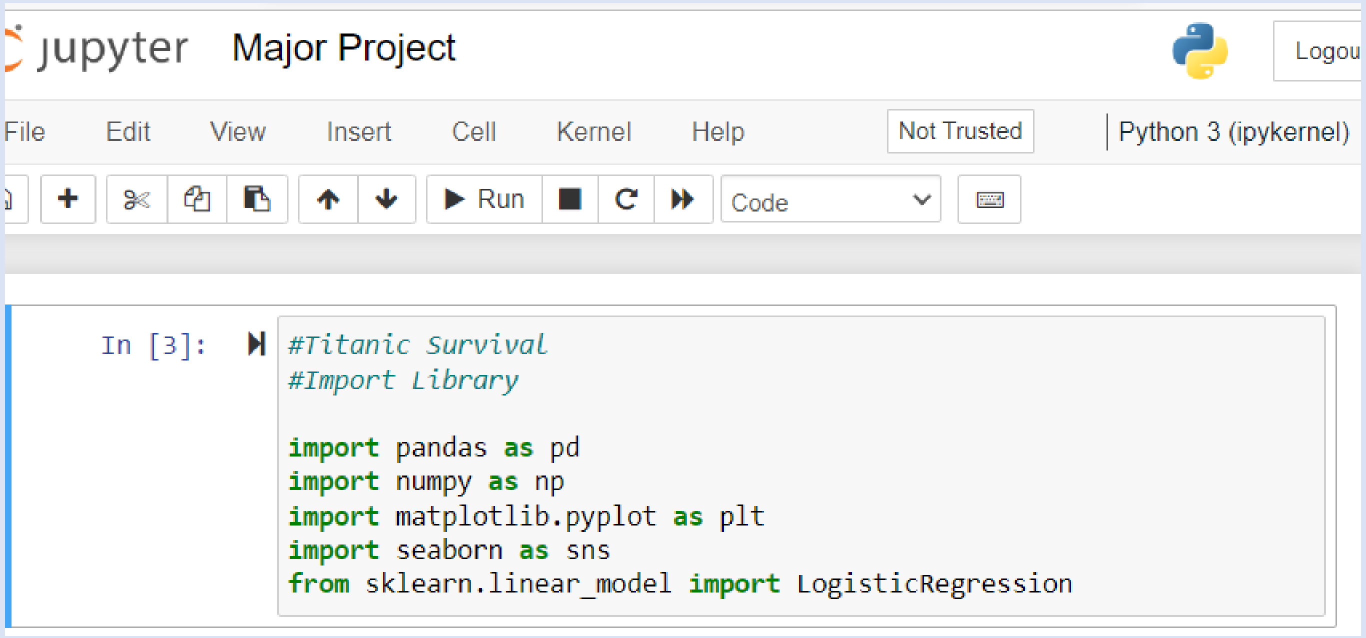
- Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features(or inputs), x

- Some of the examples of classification problems are Email spam or not spam, online transactions Fraud or not Fraud.

IMPLEMENTATION

- Importing the necessary Libraries
- Importing the dataset
- Cleaning and analyzing the dataset
- Building the model
- Using logistic regression for making prediction

IMPORTING THE NECESSARY LIBRARIES



READ AND EXPLORE THE DATA

In [4]:

```
#Read the Dataset

anupreksha_anushka_train_df = pd.read_csv("C:\\Users\\HP\\Downloads\\train.csv")
ashlesha_ekta_test_df = pd.read_csv("C:\\Users\\HP\\Downloads\\test.csv")
combine = [anupreksha_anushka_train_df, ashlesha_ekta_test_df]
```

In [12]:

```
anupreksha_anushka_train_df.describe()
```

Out[12]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	8
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	5

CLEANING AND ANALYSING THE DATA

In [15]:
```
pd.isnull(anupreksha_anushka_train_df).sum
```

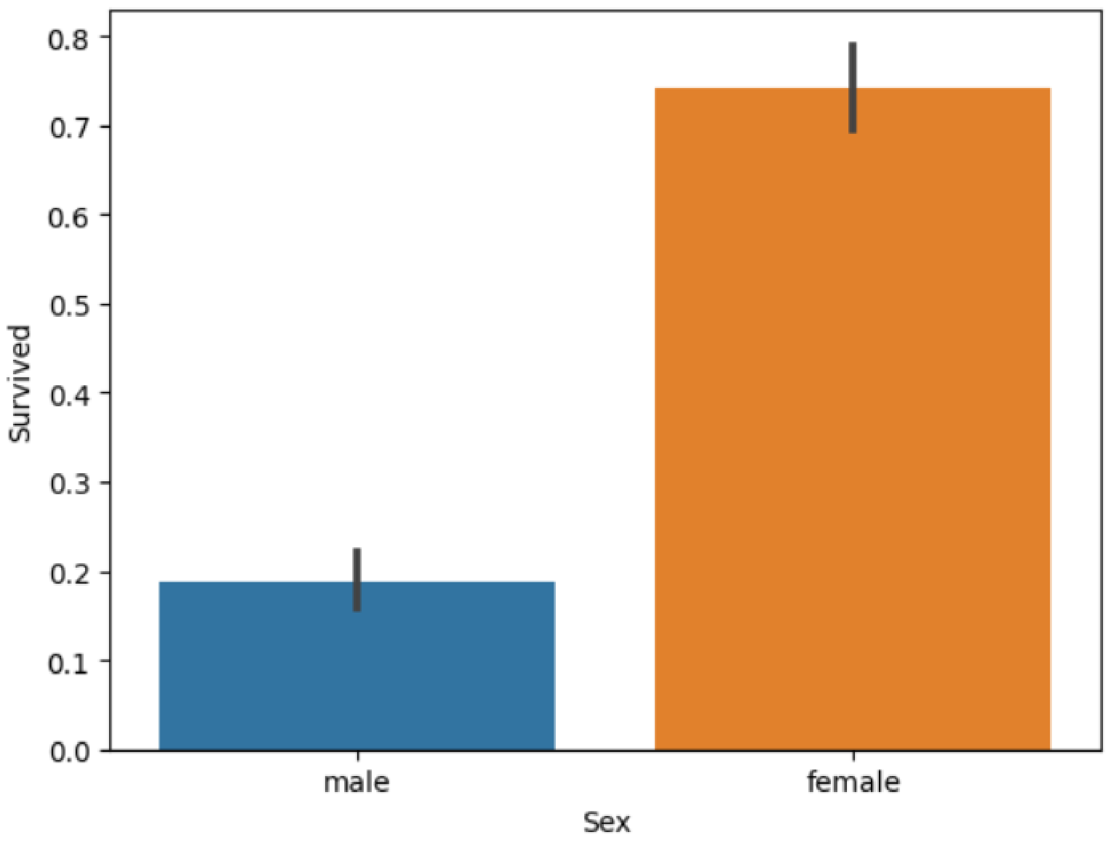
Out[15]:
```
<bound method NDFrame._add_numeric_operations.<locals>.sum of
PassengerId  Survived  Pclass   Name    Sex    Age  SibSp  Parch
Ticket   \
0           False     False   False  False  False  False  False  Fa
lse   False
1           False     False   False  False  False  False  False  Fa
lse   False
2           False     False   False  False  False  False  False  Fa
lse   False
3           False     False   False  False  False  False  False  Fa
lse   False
4           False     False   False  False  False  False  False  Fa
lse   False
..            ...       ...     ...    ...    ...    ...    ...
...      ...
886         False     False   False  False  False  False  False  Fa
lse   False
887         False     False   False  False  False  False  False  Fa
lse   False
888         False     False   False  False  False   True  False  Fa
lse   False
889         False     False   False  False  False  False  False  Fa
lse   False
890         False     False   False  False  False  False  False  Fa
lse   False

      Fare  Cabin  Embarked
0    False   True     False
1    False  False     False
2    False   True     False
```

DATA VISUALIZATION

Seeing How Independent variable are effecting survival

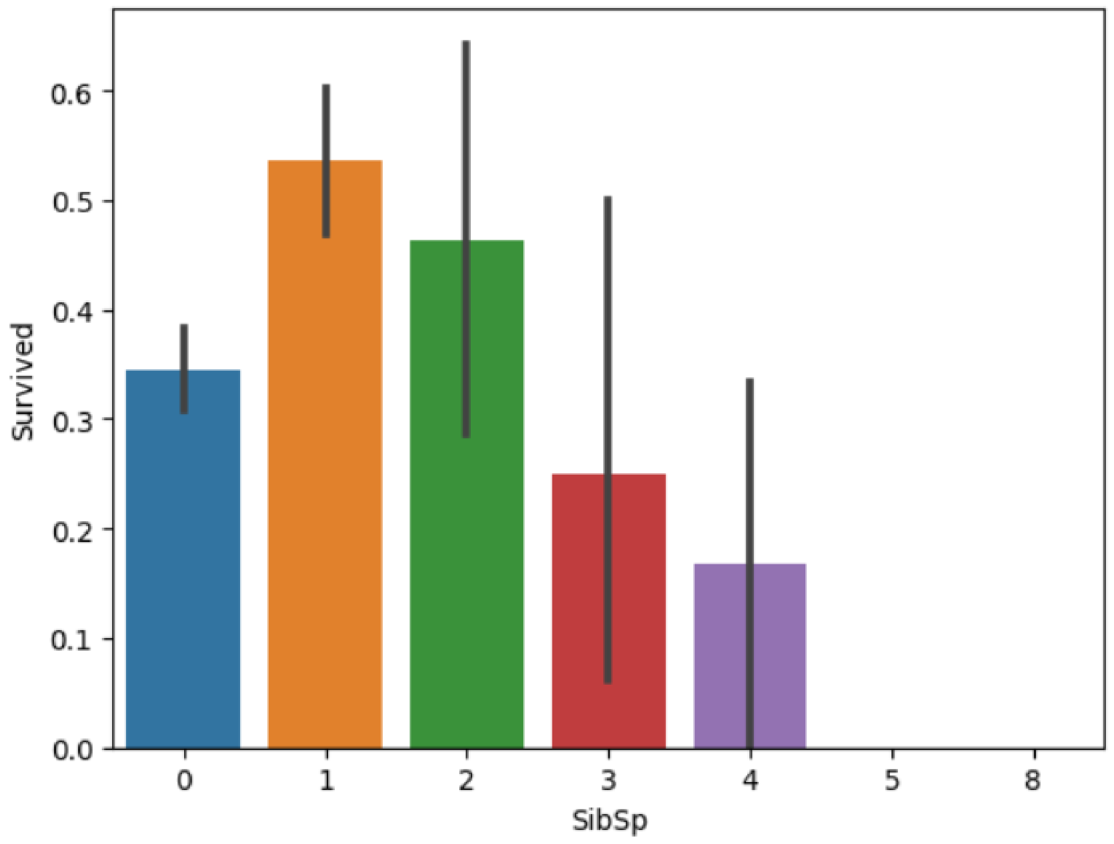
SEX FEATURES:

```
In [10]:  sns.barplot(x='Sex', y='Survived', data=anupreksha_anushka_train_df_train_df)

Out[10]:  <AxesSubplot: xlabel='Sex', ylabel='Survived'>
```



SIBLINGS OR SPOUSES FEATURES

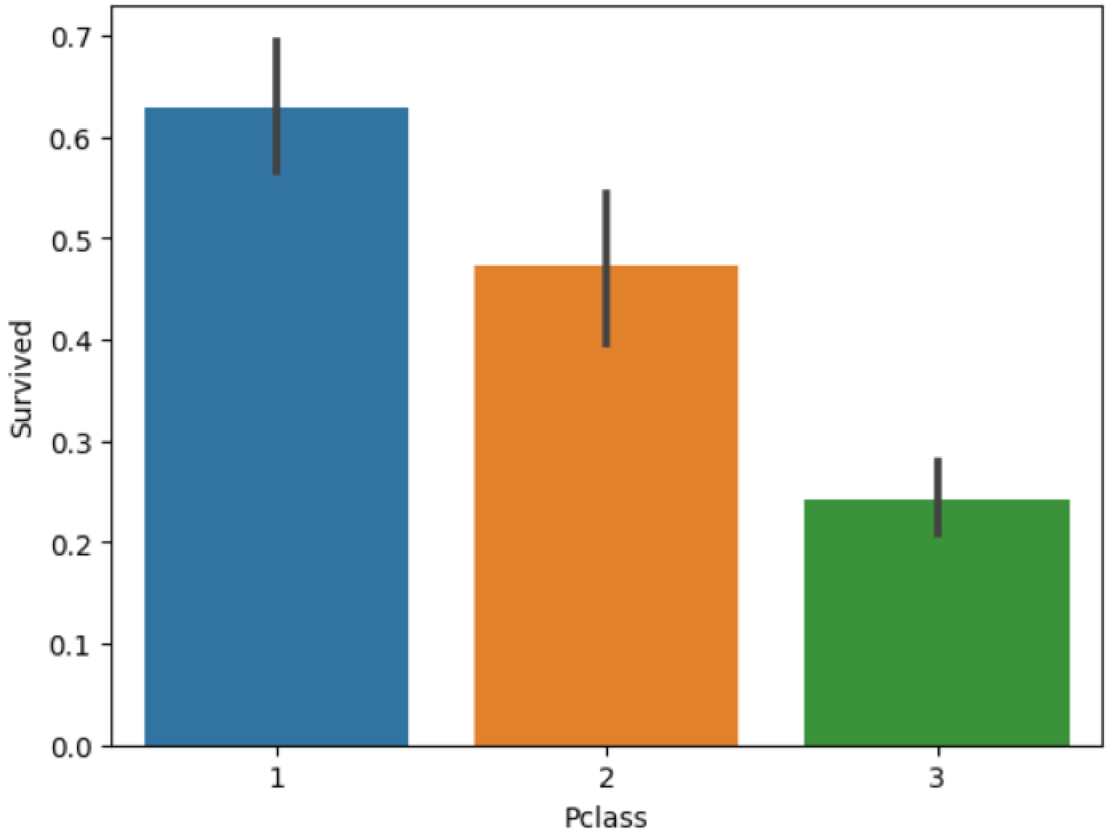
```
In [21]:  sns.barplot(x='SibSp', y='Survived', data= anupreksha_anushka_train_df)

Out[21]: <AxesSubplot: xlabel='SibSp', ylabel='Survived'>
```



Pclass: Ticket Class

In [25]:
```
sns.barplot(x='Pclass', y='Survived', data= anupreksha_anushka_train_df)
```

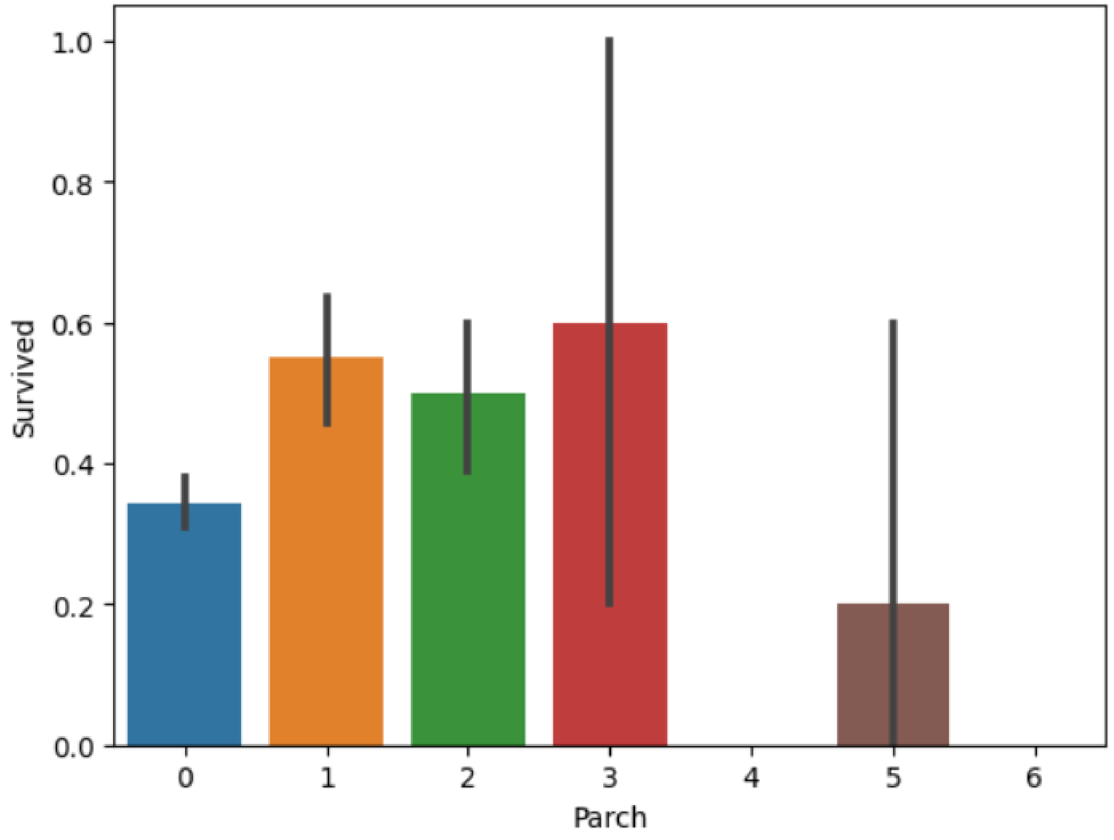
Out[25]: <AxesSubplot: xlabel='Pclass', ylabel='Survived'>



Parch Features: Number of Parents of Children

In [26]:
```
sns.barplot(x='Parch', y='Survived', data= anupreksha_anushka_train_df )
```

Out[26]: <AxesSubplot: xlabel='Parch', ylabel='Survived'>



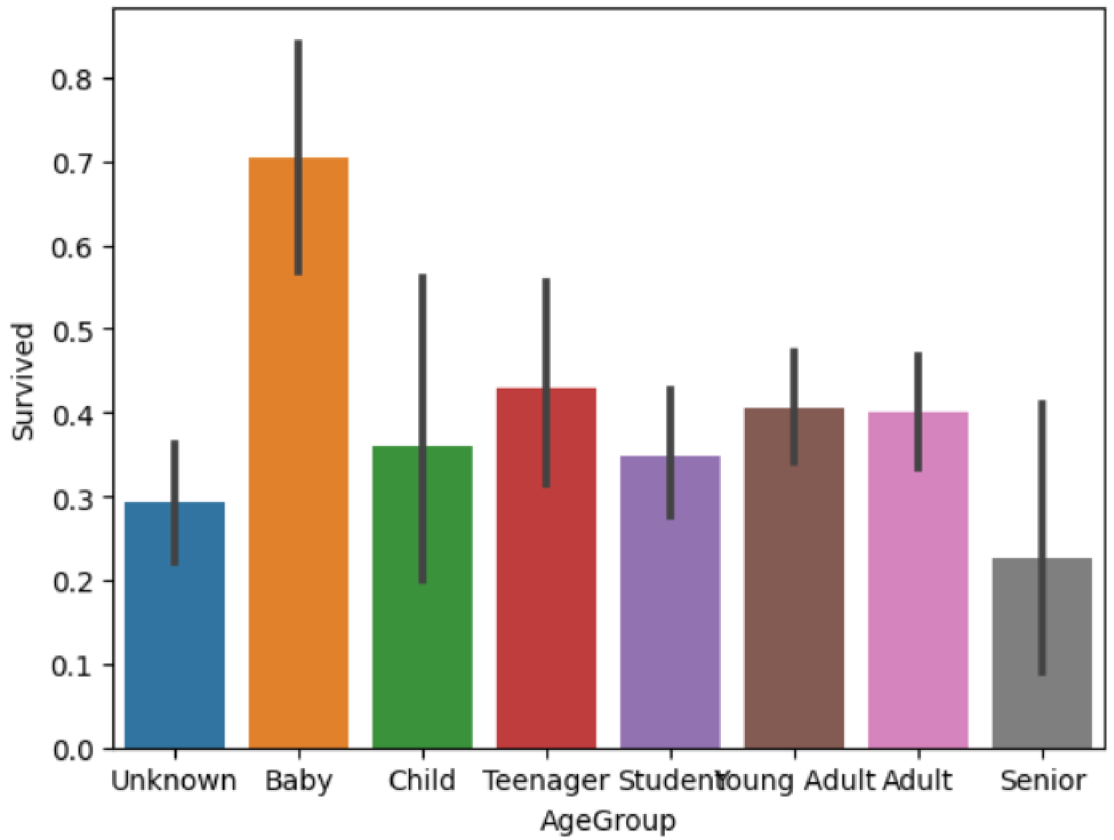
AGE FEATURES:
Sort the age into categorical Categories

In [31]:

```
anupreksha_anushka_train_df ["Age"]=anupreksha_anushka_train_df ['Age'].fillna(-0.5)
test_df["Age"]=test_df['Age'].fillna(-0.5)
bins = [-1,0,5,12,18,24,35,60,np.inf]
labels=['Unknown','Baby','Child','Teenager','Student','Young Adult','Adult','Senior']
train_df['AgeGroup']=pd.cut(train_df["Age"],bins,labels=labels)
test_df['AgeGroup']=pd.cut(test_df["Age"],bins,labels=labels)
```

AGE GROUP

```
In [33]:  sns.barplot(x='AgeGroup', y='Survived', data= anupreksha_anushka_train_df   )

Out[33]: <AxesSubplot: xlabel='AgeGroup', ylabel='Survived'>
```



BUILDING THE MODEL

In [91]:

```
#Spliting x and y in to train and test dataset
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=.25)
```

RESULT:

0: DEAD

1: SURVIVE

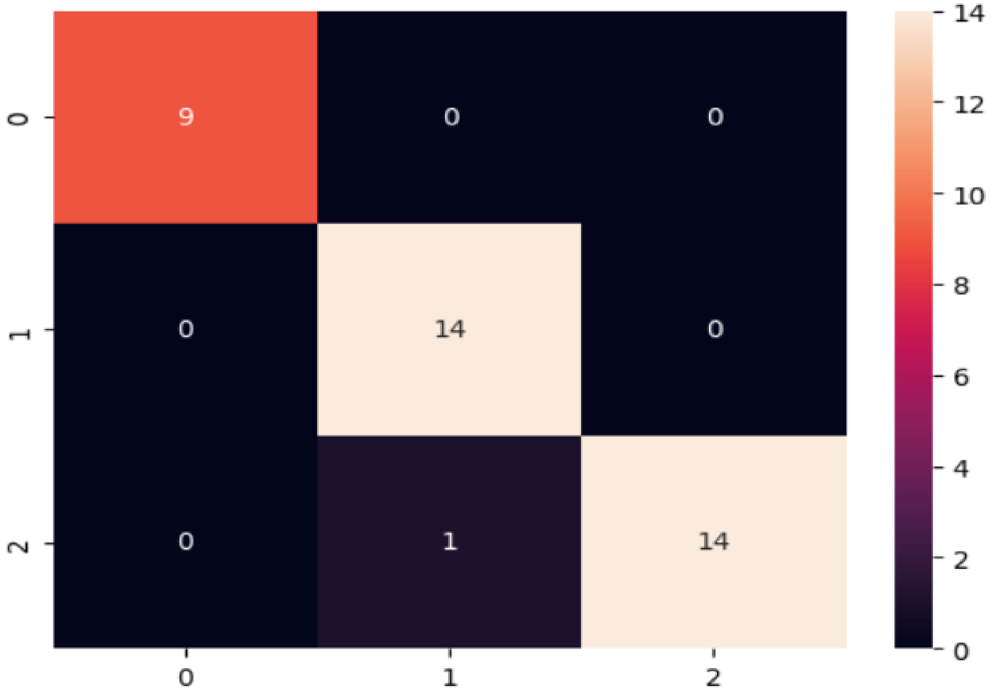
```
All survival predictions done.
All predictions exported to submission.csv file.
     PassengerId  Survived
0            892         0
1            893         0
2            894         0
3            895         0
4            896         1
5            897         0
6            898         1
7            899         0
8            900         1
9            901         0
10           902         0
11           903         0
12           904         1
13           905         0
14           906         1
```

Making Predictions and Calculating Accuracy

```
Anupreksha, Anushka, Ashlesha, Ekta Your confusion_matrix
[[13  0  0]
 [ 0 10  0]
 [ 0  0 15]]
Anupreksha, Anushka, Ashlesha, Ekta Your Classification_repor
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00        15

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38

 Anupreksha, Anushka, Ashlesha, Ekta Your Accuracy_score is :
Anupreksha, Anushka, Ashlesha, Ekta Your Model_score is- 1.0
```

```
Anupreksha, Anushka, Ashlesha, Ekta Your Accuracy_score of LOgistic Regression is : 77.09
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_l
ing: 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 in:
    https://scikit-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-regression
  n_iter_i = _check_optimize_result(
```



CONCLUSION

- I have Removed variables like “Passenger Id”, “Name”, “Ticket”, “Fare”, “Cabin”, as they are not effecting the target variable much.
- Women , children, and first class passengers as well as people with a small family had a better chance at survival. The Embarked dosen’t seem to have an effect as the percentages are in line with the amount of people embarked from each port
- And We are getting an accuracy of 77.09%.

REFERENCES

- Analyzing Titanic disaster using Machine learning algorithms – computing, communication and Automation (ICCCA), 2017 International Conference on 21 December 2017, IEEE.

- Eric Lam, Chongxuan Tang, “Titanic Machine
- Learning From Disaster”, LamTangTitanic Machine
- Learning From Disaster, 2012

- A.NG CS229 Notes, Stanford University , 2012


Thank You !