TASK TO PERFORM

NOTE:- Select a dataset for your project form kaggle or choose one that suits your needs

TITANIC CLASSIFICATION:-

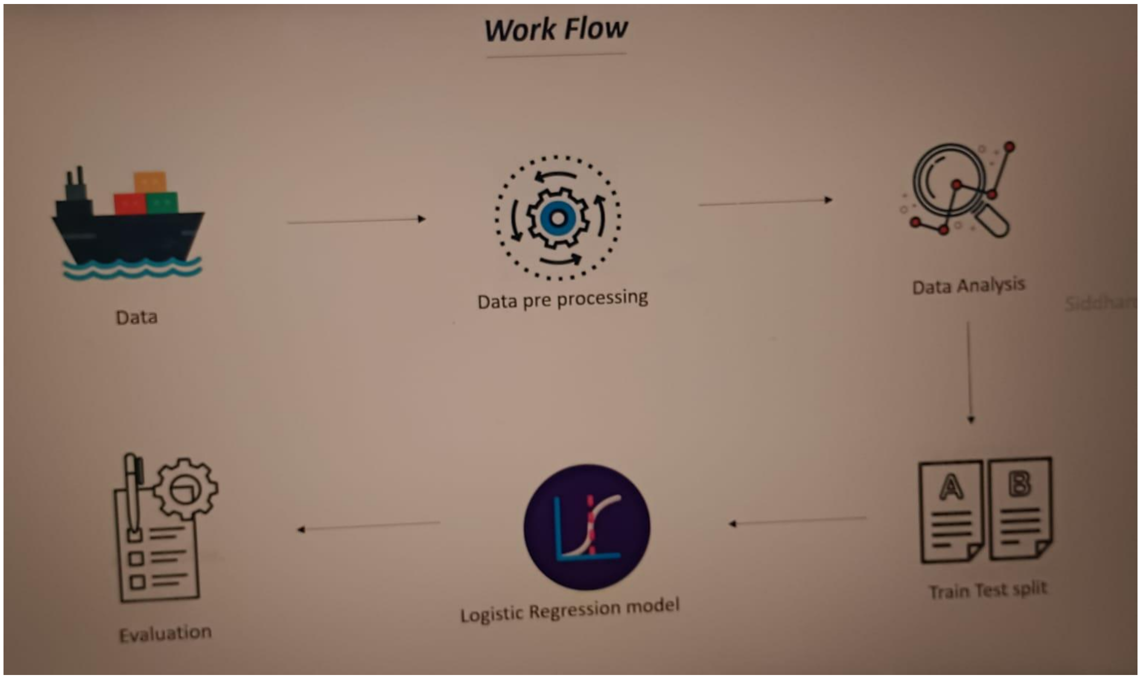
Build a predictive model to determine the like hood of survival for passengers on the titanic using data science techniques in python.

SURVIVED OR NOT SURVIVED:-

WORKFLOW:-

DATA---------→DATA PER

PROCESSING------→DATA ANALYSIS



EVALUTION<--------LOGISTIC

REGRESSION<---------TRAIN TEST SPILT

(1) IMPORTING THE DEPENDENCIES:-

*Import numpy as np

*Import pandas as pd

*Import Matplotlib. Pyplot as plt

*Import seaborn as SNS

*From Sklearn. model_ selection import train _test_ split

*Form sklearn. linear_ modal import logistic regression

*Form sklearn . Metrics import accuracy_ score

(2) DATA COLLECTION AND PROCESSING :-

#load the data from csv file to pandas Data frame

Titanic _ data=pd. read_ csv('/content/train.csv')

#printing the first 5Rows of the data frame

Titanic_ data. head()

(3) TABLE:-

	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

(4) #Number of rows and columns

Titanic_ data. shape

(891,12)

#Getting some information about the data

Titanic_data.info()

(5) <class 'pandas' .cone. frame. 'Data frame'>

Range Index: 891 entire, 0 to 890

Data Column (TOTAL 12 COLUMN):

#	Column	Non-Null Count	d type
---	-------	----------------	-------
0	Passenger	891 non-null	int64
1	Survived	891 non-null	int64
2	P class	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

dttype:float64(2),int64(5),object(5)

memory usages:83.7+KB

(6) #Check the numbers of missing values in each columns

Titanic_ data .is null ().sum().

Passenger ID	0
Survived	0
P class	0
Name	0
Sex	0
Age	0
Sibsp	0
Parch	0
Ticket	0
Fare	0
Cabin	0
Embarked	0

D type: int64

HANDLING THE MISSING VALUES:-

(7) #Drop the "cabin" column from the data frame.

Tianice_data.=titanic_ data .drop[column='cabin' ,axis=1]

(8) #Replacing the missing values in "AGE" column with mean value

Titanic_ data['AGE']. Fill na (titanic-data['AGE'].MEAN(),in place=true)

(9) #Finding the value made value of "Embarked" column

print(titanic_ data['Embarked'].mode())

0   S

D type : Object

(10) #1Print (titanic_ data['Embarked'].mode()[0])

s

(11) #Replacing the missing values in "Embarked" Column with mode value

Titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],inplace=true)

(12) #Check the number of missing values in each column

Titanic_ data .is null().sum()

Passenger ID	0
Survived	0
P class	0
Name	0
Sex	0
Age	0
Sibsp	0
Parch	0
Ticket	0
Fare	0
Embarked	0

D type: int64

DATA ANALYSIS:-

(13) #Getting same statistical measure about the data.

Titanic_ data. describe()

TABLE:-

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	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	13.002015	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	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

(14) #Finding the number of people survived and not survived.

Titanic _ data['Survived'] value_ counts()

0 549

1 342

Name : Survived,dtype:int64.

DATA VISUALIZATIONS:-

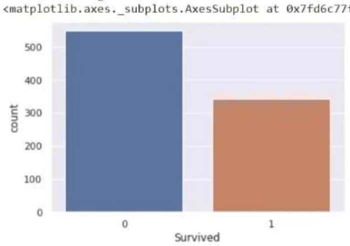
(15) 1 SNS. Set()

Making a count plot for 'Survived' column .

SNS. Count plot ("Survived", data=titanic_data0

(16) GRAPH:- SURVIVED

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid pc
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6c77f16d0>
```



Ti tanic_ data['sex'].value_ counts()

Male   577

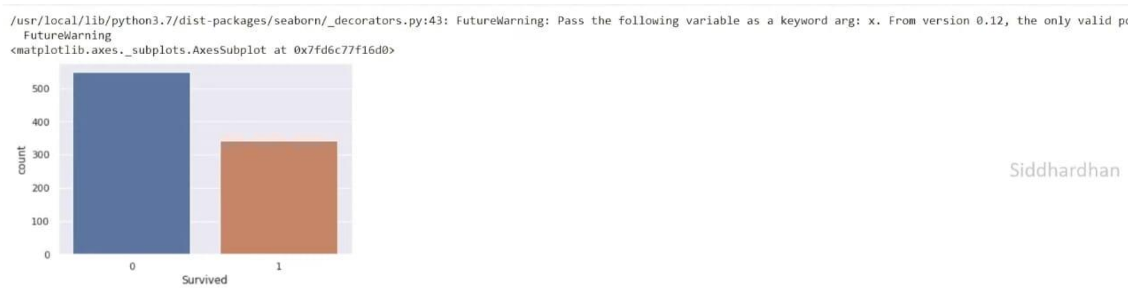
Female   314

Name: sex, d types: int64

(17) #Making a count plot for "sex" column

SNS. Count plots('sex', data=titanic_ data)

GRAPH:- SEX



(18) #Number of survivors gender wise

SNS. COUN TPLOT('SEX', hue= 'survived', data=titanic_ data)

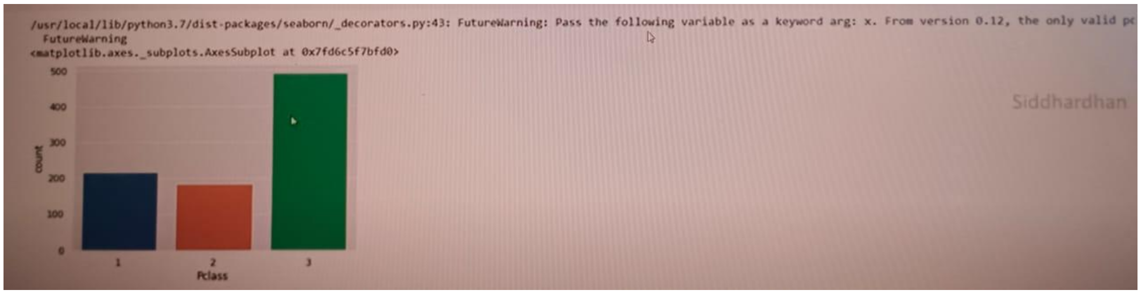
GRAPH:- MALE AND FEMALE

SEX

(19) #Making a count plot for "p class" column

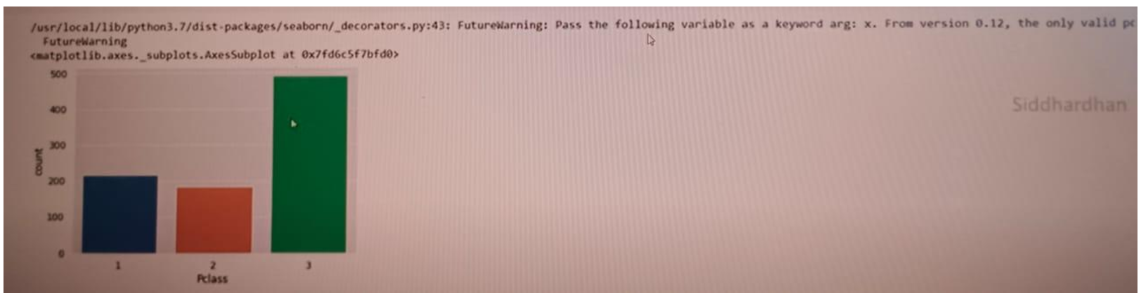
SNS. Count plot('pi class', data=titanic_ data)

GRAPH:- PICLASS



(20) SNS. Count Piot ('pclass' , hue='survived' ,data=titanic_ data)

GRAPH:-PICLASS 2



ECONDING THE CATEGORICAL COLUNMS

(21) Titanic_ data['SEX'].value_ count()

Male 577

Female 314

Name: Sex, d type: int64

(22) Titanic_ data['Embarked'].value_ count()

S 646

C 168

Q 77

Name: Embarked, d type: int64

(23) #converted categorical Columns

Titanic_data.replace('sex':{'male':0,'female':1),'Embarked':{'S':0,'C':1,'Q':2}},inplace=true)

Titanic_ data .head()

TABLE:- PASSANGERS

	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

SEPARTING FEATURES AND TARGET

(24) X=titanic_data.drop(columns=['passangersID','Name','survived'],axis=1)

Y=titanic_ data['survived']

(25) Print(x)

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	0	22.000000	1	0	7.2500	0
1	1	1	38.000000	1	0	71.2833	1
2	3	1	26.000000	0	0	7.9250	0
3	1	1	35.000000	1	0	53.1000	0
4	3	0	35.000000	0	0	8.0500	0
..	...	...	...	...	...	...	...
886	2	0	27.000000	0	0	13.0000	0
887	1	1	19.000000	0	0	30.0000	0
888	3	1	29.699118	1	2	23.4500	0
889	1	0	26.000000	0	0	30.0000	1
890	3	0	32.000000	0	0	7.7500	2

[891 ROWS X 7 COLUMN]

(26) Print(Y)

0	1
1	1
2	1
3	1
4	0

---

886 0

887 1

888 0

889 1

890 0

NAME: SURVIVED, LENGTH: 891, DTYPE: INT64

SPLITTING THE DATA INTO TRAINING DATA AND TEST DATA

(27) X_ Train, X_ test, Y_ Train, Y Test=Train_ Test_ Split

(X,Y),Test_ Size=02,Random_State=2)

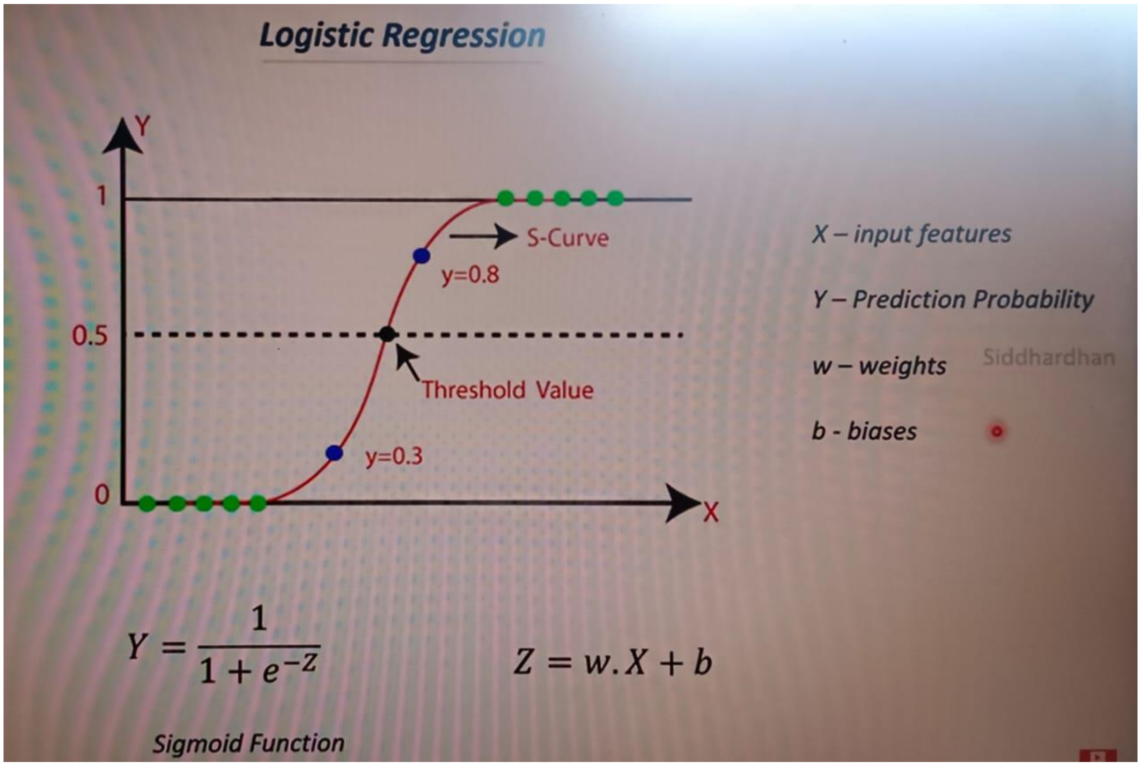
(28) Print(X. Shape ,X_ Train .Shape ,X_ Test .Shape)

(891,7) (712,7) (179,7)

MODAL TRAINING

LOGISTICE REGRESSION

DIAGRAM:-



(30) 1Modal=logistic regression()

(31) #Training the logistic regression modal with training data.

Modal. fit(X_ Train, Y_ Train)

/user/local/lib/python3.7/dist-packages/sklearn/linear_modal/_logistic.py:940:convergencewarming:lbfgs failed converge (status=1):

STOP: TOTAL NO OF ITERATIONS REACHED LIMIT.

Increases the number of iteration (max_ iter) or scale the data as shown in:

please also refer to the documentation for alternative solver options:

extra _warming_ msg =_LOGISTIC_SLOVER_CONVERGRNCE_MSG)

Logistic regression(C=1.0, Class_ Weight=None, dual=False ,fit_ intercept=True,

intercept_ scaling=1, 11_ratio=None, max_ iter=100,

multi_ class=auto', n_ jobs=None ,Penalty='12',

random_ state=None, solver='lbfgs', toltal =0.0001, verbose=0, warm_ start=false)

MODAL EVALUATION

ACCURACY SCORE

(32) #Accuracy on training data

X_ Train_ prediction=modal .predict(X_ train)

Print (X_ train_ prediction

[0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 1

0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0

0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1

1 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0

0 1 1 1 0 1 0 1 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0

1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 0 1 1

0 1 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1 0 0 1

1 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0

0 1 1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1

1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0

0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 1

1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 0 0 1

0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 1 0 1

1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1

0 0 1 0 0 1 1 0 0]

(33) Training_data_accuracy=accuracy_score(Y_Train,X_Train_predication)

Print('Accuracy score of training data:', training_ data_ accuracy

accuracy score of training data:0.8875842696629213

(34) #Accuracy on test data.

X_ test_ predication=modal _predict(X_ test

(35 print(X_ test_ prediction)

[0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1

0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0

1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0

0 0 0 1 1 0 1 0 1 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 1 0

0 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0]

(36) Test_ data_ accuracy=accuracy _score(Y_ test, X_ test_ predication)

Print('Accuracy score of test data:' ,Test _data_ accuracy)

Accuracy score of test data: 0.7821229050279329.