Import the required lib

In [101]:

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

Load Datasets

In [102]:

```
training_dataset=pd.read_csv('train.csv')
```

In [93]:

```
testing_dataset=pd.read_csv('test.csv')
```

In [94]:

```
#gender_submission=pd.read_csv('gender_submission.csv')
```

Q1: In training set, which features are available?

In [95]:

```
training_dataset.head()
```

Out[95]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [96]:
```
training_dataset.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
```

Q5: In training set, which features contain

1. null

In [97]:
```
training_dataset.isnull().any(axis=0) # this function is use to show all null val
                                      # axis=0, we are checking the null values i
```

Out[97]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

In [98]:

```
training_dataset.isnull().any(axis=1) # axis =1, checking null values in the trai
```

Out[98]:

```
0       True
1      False
2       True
3      False
4       True
       ...  
886     True
887    False
888     True
889    False
890     True
Length: 891, dtype: bool
```

In [99]:

```
training_dataset.isnull().sum()
```

Out[99]:

```
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 [100]:

```
d = {'empty': pd.Series([0,1,2,3,4,5,6,7,8,9,10], index=['PassengerId', 'Survived
df = pd.DataFrame(d)
print (df)
```

```
              empty
PassengerId       0
Survived          1
Pclass\tName      2
Sex               3
Age               4
SibSp             5
Parch             6
Ticket            7
Fare              8
Cabin             9
Embarked         10
```

In [ ]:

```
training_dataset.dtypes
```

# DataFrame-empty property

the empty property indicates whether DataFrame is empty or not. this properties return in term of True and False

return in term of True and False
1. True if DataFrame is entirely empty means any of the axes are of the length 0.
2. False if the DataFrame is not Entirely empty means any of the axes are of the length not zero
special case:
if DataFrame contains only NaNS(Not a Numbers), then it is still not considerted empty

In [ ]:

```
training_dataset.empty
```

In [ ]:

```
#  Blank?
```

# Q2: In training set, which features are categorical?

# categorical features in the traning dataset

1. Name
2. Sex
3.Ticket
4.

# Q3: In training set, which features are numerical (e.g., discrete, continuous, or time series based)?

In [103]:

```
trainig_numerical_data=training_dataset[['PassengerId','Survived','Pclass','Age',
```

In [53]:

```
trainig_numerical_data.head()
```

Out[53]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.2500
1	2	1	1	38.0	1	0	71.2833
2	3	1	3	26.0	0	0	7.9250
3	4	1	1	35.0	1	0	53.1000
4	5	0	3	35.0	0	0	8.0500

# Q7: To understand the distribution of numerical feature values across the samples, please list the

properties, including count, mean, std, min, 25% percentile, 50% percentile, 75% percentile, max, of
numerical features?

In [54]:
```
trainig_numerical_data.describe()
```

Out[54]:

	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 training set, which features are mixed data types?

In [104]:
```
trainig_category_data=training_dataset[['Name','Sex','Ticket','Cabin','Embarked']
```

In [56]:
```
trainig_category_data.head()
```

Out[56]:

	Name	Sex	Ticket	Cabin	Embarked
0	Braund, Mr. Owen Harris	male	A/5 21171	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C85	C
2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	NaN	S
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	C123	S
4	Allen, Mr. William Henry	male	373450	NaN	S

In [57]:
```
trainig_category_data.count()
```

Out[57]:
```
Name        891
Sex         891
Ticket      891
Cabin       204
Embarked    889
dtype: int64
```

In [58]:

```
trainig_category_data.describe()
```

Out[58]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Madill, Miss. Georgette Alexandra	male	1601	G6	S
freq	1	577	7	4	644

In [105]:
```
training_dataset
```

Out[105]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
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	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

891 rows × 12 columns

```
Q9: Can you observe significant correlation (average survivied ratio>0.5) among
the group of
Pclass=1 and Survived? If Pclass has significant correlation with Survivied, we
should include this
feature in the predictive model. Based on your computation, will you include
this feature in the
```

predictive model?

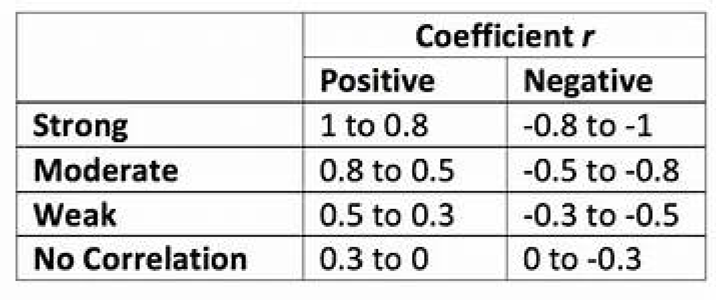
In [60]:

```
from PIL import Image
image = Image.open('pic.jpg')
image.show()
```

In [61]:

```
image # A claculated Ratios charts of correlation and coefficents, we are keeping
```

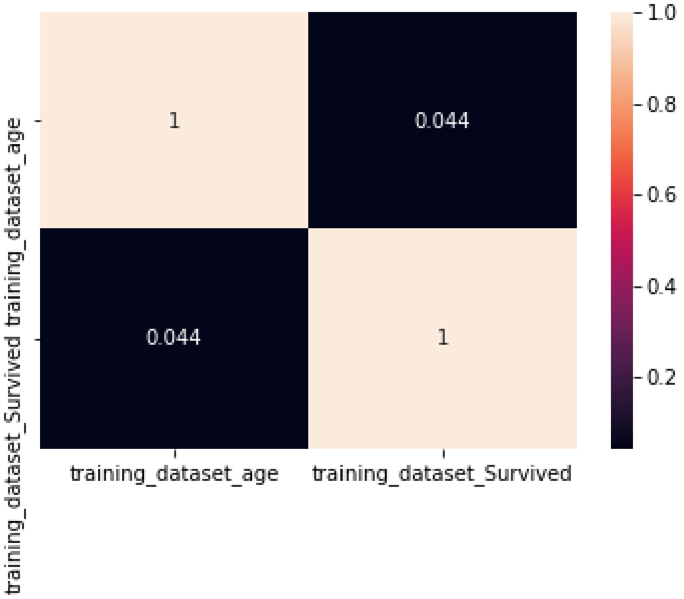
Out[61]:



	Coefficient *r*	
	Positive	**Negative**
Strong	1 to 0.8	-0.8 to -1
Moderate	0.8 to 0.5	-0.5 to -0.8
Weak	0.5 to 0.3	-0.3 to -0.5
No Correlation	0.3 to 0	0 to -0.3

In [62]:

```
corrMatrix = training_dataset.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



```
the correlation between pclass and survived is 'weak'(-0.34) which less then
0.5 we can not include this feature in our model.
```

In [106]:

```
dataset_sex_survived=training_dataset[['Sex','Survived']]
```

In [107]:

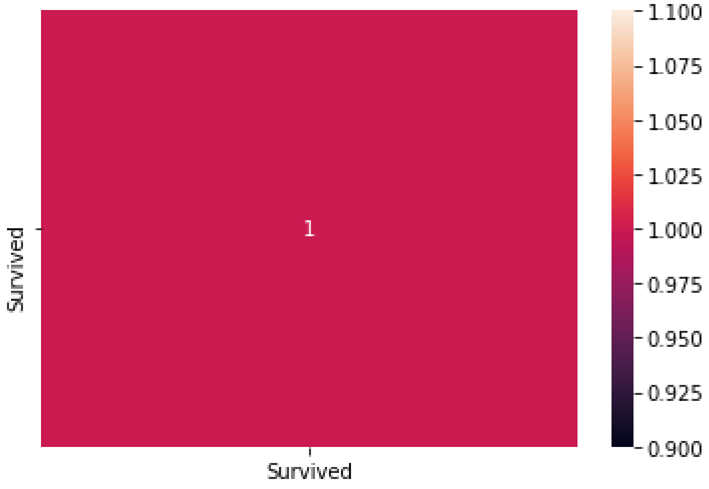
```
dataset_sex_survived.head()
```

Out[107]:

	Sex	Survived
0	male	0
1	female	1
2	female	1
3	female	1
4	male	0

In [108]:

```
corrMatrix = dataset_sex_survived.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



In [109]:

```
dataset_age=training_dataset[['Age','Survived']]
dataset_age.head()
```

Out[109]:

	Age	Survived
0	22.0	0
1	38.0	1
2	26.0	1
3	35.0	1
4	35.0	0

In [66]:

```
#plt.hist(dataset_age,bins=[0,3,15,25,80])

#plt.show()
```

In [110]:
```
training_dataset
```

Out[110]:

	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
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

891 rows × 12 columns

***

```
Q11: Let us start by understanding correlations between a numeric feature (Age)
and our predictive
```

goal (Survived). A histogram chart is useful for analyzing continuous numerical variables like Age
where banding or ranges will help identify useful patterns. The histogram can indicate distribution of
samples using automatically defined bins or equally ranged bands. This helps us answer questions
relating to specific bands (e.g., infants, old). Please plot the histogram plots between ages and
Survived (Figure 1 is an example), and answer the following questions:
• Do infants (Age <=4) have high survival rate?
• Do oldest passengers (Age = 80) survive?
• Do large number of 15-25 year olds not survive?

Solution for Q11:
a histogram shows the number of occurances of differnet values in a dataset in our case is training_dataset, in our dataset we calculate the occurances between age and survived attributes.the age and survived in our dataset is one big python list, so as the principel of statistical average and statistical variablity we have to compress these numbers into a few values that are easier to understand yet describe our dataset well enough, such as under:
1. mean
2. median
3. STD
based on these values we can get a good sense data
BINS AND RANGES:
we see that in the age attributs we have mean value 29.699118,ang survived mean vlaue 0.383838, and std value for the age is 14.526497 and std value for the survivied is 0.486592, the age dataset have unique value 29___ 30 and for survivied have unique value 14_____ 15 unique values if we simply count the unique values in the dataset and put that on a bar chart we got the following visualization. i used a random generator to generat the data point of the both data sets. this will generate two dataset with 250, point in each and we also fixed the parameter of the random generator. Thus we will get the very same numpy arrays with same data points that we have in the required attributes.
in the training_dataset_age we will get 250 values of age, in the training_dataset_survived there are 250 survived values of survived people
Note: .hist() grouping into bins is not the same as grouping by unique values as a bin usually conatin a range of values

In [68]:
```
%matplotlib inline
Age= 29.699118 # mean of age
Survived= 0.383838# mean of survived
sample=250
np.random.seed(0)
training_dataset_age=np.random.normal(Age,Survived,sample).astype(int)

Age= 14.526497 # std of age
Survived= 0.486592# Std of survived
sample=250
np.random.seed(1)
training_dataset_Survived=np.random.normal(Age,Survived,sample).astype(int)
```

In [69]:
```
training_dataset_age
```

Out[69]:
```
array([30, 29, 30, 30, 30, 29, 30, 29, 29, 29, 29, 30, 29, 29, 29, 29, 30,
       29, 29, 29, 28, 29, 30, 29, 30, 29, 29, 29, 30, 30, 29, 29, 29, 28,
       29, 29, 30, 30, 29, 29, 29, 29, 29, 30, 29, 29, 29, 29, 29, 29, 29,
       29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
       29, 29, 29, 29, 30, 29, 29, 29, 29, 29, 29, 29, 29, 30, 29, 29, 30,
       30, 30, 29, 29, 30, 29, 30, 29, 30, 29, 29, 29, 30, 29, 29, 30, 29,
       29, 30, 29, 30, 29, 29, 30, 30, 30, 30, 29, 30, 29, 30, 30, 29, 29,
       30, 29, 29, 29, 30, 29, 29, 29, 30, 29, 29, 29, 29, 29, 29, 29, 29,
       29, 29, 29, 29, 29, 29, 29, 29, 30, 30, 29, 30, 29, 29, 29, 30, 29,
       29, 29, 29, 30, 29, 29, 29, 29, 30, 30, 29, 29, 30, 29, 29, 30, 29,
       30, 29, 30, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 28, 29, 29, 29,
       29, 29, 30, 29, 29, 29, 29, 29, 29, 29, 30, 30, 30, 29, 29, 30, 29,
       29, 29, 29, 29, 29, 29, 30, 29, 30, 29, 29, 29, 29, 29, 30, 29, 29,
       29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 30, 28,
       29, 29, 29, 29, 29, 29, 29, 29, 30, 30, 29, 29])
```

In [70]:
```
training_dataset_Survived
```

Out[70]:
```
array([15, 14, 14, 14, 14, 13, 15, 14, 14, 14, 15, 13, 14, 14, 15, 13, 14,
       14, 14, 14, 13, 15, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
       14, 14, 13, 14, 15, 14, 14, 14, 14, 15, 14, 14, 14, 15, 14, 14, 14,
       14, 13, 14, 14, 14, 14, 14, 14, 14, 14, 15, 14, 14, 14, 14, 15, 15,
       15, 13, 13, 14, 14, 14, 14, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14,
       14, 14, 14, 14, 15, 15, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15,
       14, 14, 13, 14, 14, 14, 14, 14, 13, 14, 14, 14, 14, 13, 14, 13, 15,
       14, 14, 14, 15, 15, 13, 15, 15, 14, 13, 14, 14, 14, 13, 14, 14, 14,
       14, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 13, 15,
       15, 14, 14, 14, 14, 14, 13, 14, 14, 14, 14, 14, 14, 15, 14, 15, 13,
       14, 14, 15, 14, 14, 14, 15, 14, 14, 14, 13, 14, 14, 15, 14, 14, 14,
       14, 14, 14, 15, 14, 15, 15, 14, 13, 14, 14, 14, 15, 14, 14, 14, 15,
       13, 13, 13, 14, 13, 15, 14, 13, 15, 14, 13, 14, 14, 14, 15, 14, 15,
       14, 15, 14, 13, 15, 14, 14, 14, 14, 14, 14, 14, 15, 14, 13, 13, 14,
       13, 14, 14, 14, 14, 13, 14, 14, 14, 14, 15, 13])
```

In [71]:
```
training_dataset=pd.DataFrame({'training_dataset_age':training_dataset_age,'trai
In [111]: training_dataset
```

Out[111]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
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	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

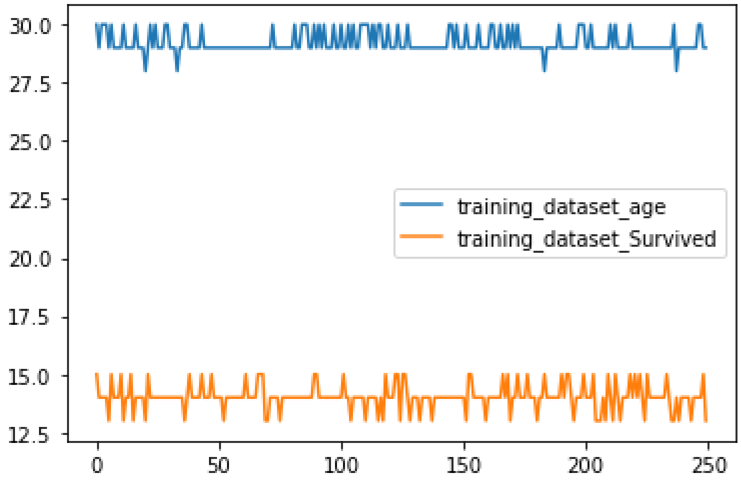
891 rows × 12 columns

In [73]:

```
training_dataset.plot() # bar chart for visualization
```

Out[73]:

```
<AxesSubplot:>
```



In [74]:

```
training_dataset.groupby('training_dataset_age').count()
```

Out[74]:

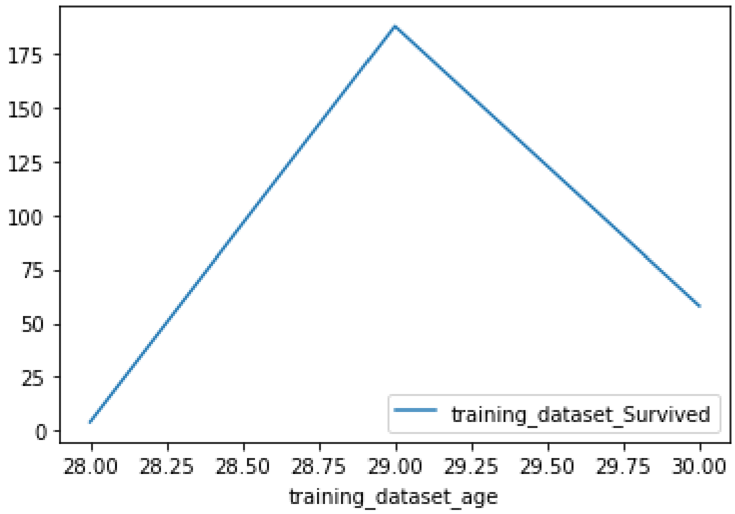
	training_dataset_Survived
training_dataset_age	
28	4
29	188
30	58

In [75]:

```
training_dataset.groupby('training_dataset_age').count().plot()
```

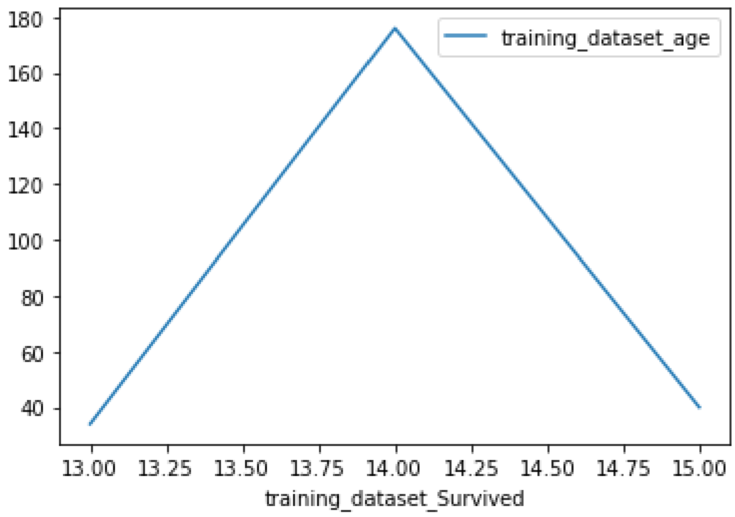
Out[75]:

```
<AxesSubplot:xlabel='training_dataset_age'>
```



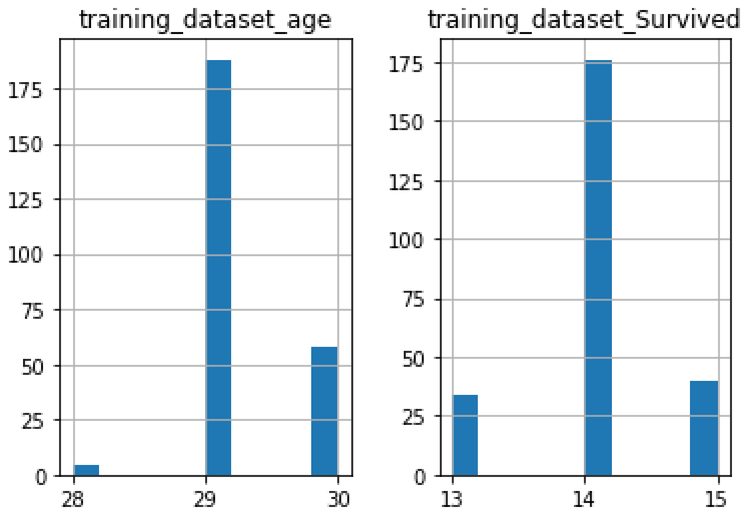
```
In [76]: training_dataset.groupby('training_dataset_Survived').count().plot()

Out[76]: <AxesSubplot:xlabel='training_dataset_Survived'>
```



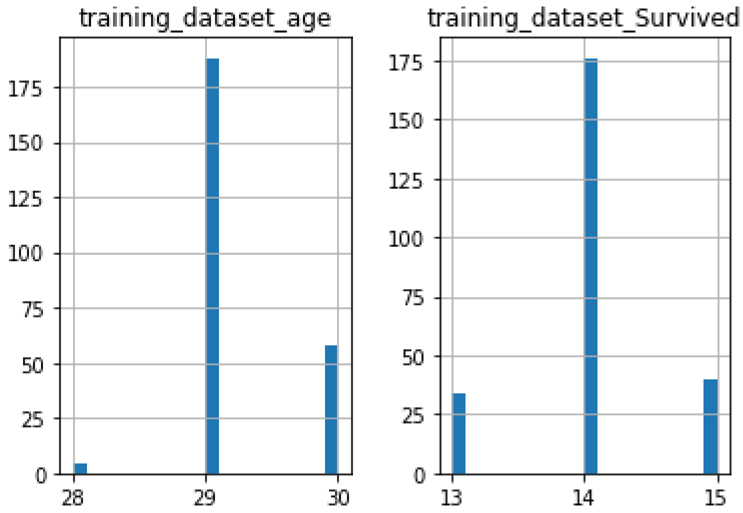
```
In [77]: training_dataset.hist()# these unique values is grouped into ranges, these ranges
                                #and in python the default number of bins is 10

Out[77]: array([[<AxesSubplot:title={'center':'training_dataset_age'}>,
                 <AxesSubplot:title={'center':'training_dataset_Survived'}>]],
               dtype=object)
```



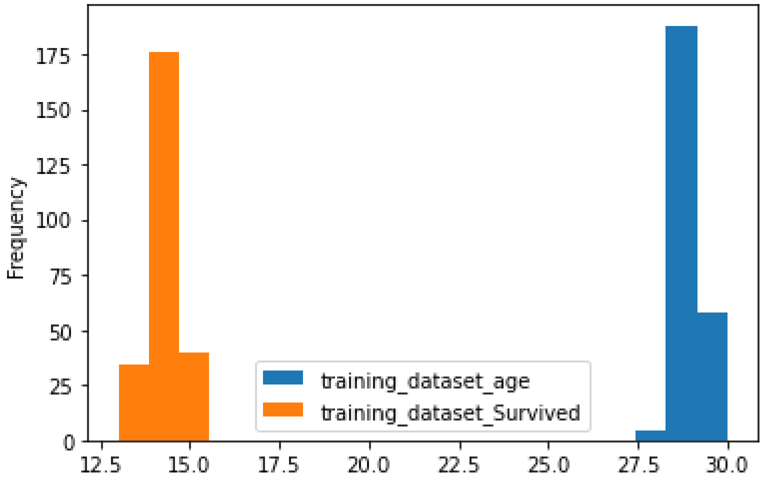
```
In [78]: training_dataset.hist(bins=20)
```

```
Out[78]: array([[<AxesSubplot:title={'center':'training_dataset_age'}>,
                 <AxesSubplot:title={'center':'training_dataset_Survived'}>]],
               dtype=object)
```



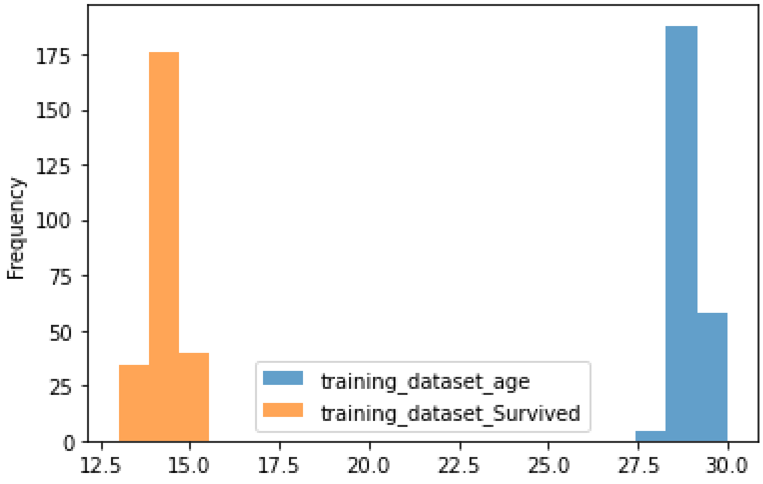
```
In [79]: training_dataset.plot.hist(bins=20)
```

```
Out[79]: <AxesSubplot:ylabel='Frequency'>
```



In [80]:
```
training_dataset.plot.hist(bins=20,alpha=0.7)
```

Out[80]:
```
<AxesSubplot:ylabel='Frequency'>
```



# Q16: We can convert features which contain strings to numerical values. This is required by most

```
model algorithms. Doing so will also help us in achieving the feature
completing goal. In this
question ,please convert Sex feature to a new feature called Gender where
female=1 and male=0.
```

In [81]:
```
d = {'Gender': pd.Series([0,1], index=['male','female'])}
df = pd.DataFrame(d)
print (df)
```

```
        Gender
male         0
female       1
```

In [82]:
```
training_dataset.head()
```

Out[82]:

	training_dataset_age	training_dataset_Survived
0	30	15
1	29	14
2	30	14
3	30	14
4	30	14

In [83]:
```
header = training_dataset.iloc[0] #how to change the column name?
```

In [84]:
```
print(header)
```

```
training_dataset_age         30
training_dataset_Survived    15
Name: 0, dtype: int32
```

# Q18: Completing a categorical feature: Embarked feature takes S, Q, C values based on port of

```
embarkation. Our training dataset has some missing values. Please simply fill
these with the most
common occurrences.
```

In [85]:
```
training_dataset.ffill(axis = 1)
```

Out[85]:

	training_dataset_age	training_dataset_Survived
0	30	15
1	29	14
2	30	14
3	30	14
4	30	14
...	...	...
245	29	14
246	30	14
247	30	14
248	29	15
249	29	13

250 rows × 2 columns

In [113]:
```
n = 3
training_dataset['Embarked'].value_counts()[:n].index.tolist()
```

Out[113]:
```
['S', 'C', 'Q']
```

In [114]:
```
training_dataset['Embarked'].value_counts() # here we are cheecking the most occu
```

Out[114]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

# in the above example S occur=646, Q occure=77 ,and C occur=168 so we replace the value of missing values with S

In [115]:
```
training_dataset['Embarked']=training_dataset['Embarked'].fillna('S')# replacemen
```

In [89]:
```
training_dataset.head()
```

Out[89]:

	training_dataset_age	training_dataset_Survived
0	30	15
1	29	14
2	30	14
3	30	14
4	30	14

In [90]:
```
training_dataset.isnull().sum()
```

Out[90]:
```
training_dataset_age        0
training_dataset_Survived   0
dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: