```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving titanic.csv to titanic.csv

```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
titanic = pd.read_csv('titanic.csv')
titanic.shape
```

(891, 12)

```
titanic.head(5)
```

	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

```
titanic.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
682	683	0	3	Olsvigen, Mr. Thor Anderson	male	20.0	0	0	6563	9.225	NaN	S
790	791	0	3	Keane, Mr. Andrew "Andy"	male	NaN	0	0	12460	7.750	NaN	Q
417	418	1	2	Silven, Miss. Lyyli Karoliina	female	18.0	0	2	250652	13.000	NaN	S
560	561	0	3	Morrow, Mr. Thomas Rowan	male	NaN	0	0	372622	7.750	NaN	Q
440	441	1	2	Hart, Mrs. Benjamin (Esther Ada Bloomfield)	female	45.0	1	1	F.C.C. 13529	26.250	NaN	S

```
X = pd.DataFrame(titanic.loc[ : , \
           ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']])
X
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	S
887	1	female	19.0	0	0	30.0000	S
888	3	female	NaN	1	2	23.4500	S
889	1	male	26.0	0	0	30.0000	C
890	3	male	32.0	0	0	7.7500	Q

891 rows × 7 columns

```
Y = pd.DataFrame(titanic.loc[:, ['Survived']])
Y
```

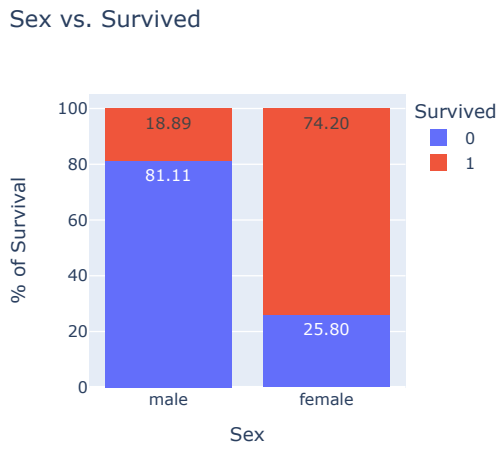
	Survived
0	0
1	1
2	1
3	1
4	0
...	...
886	0
887	1
888	0
889	1
890	0

891 rows × 1 columns

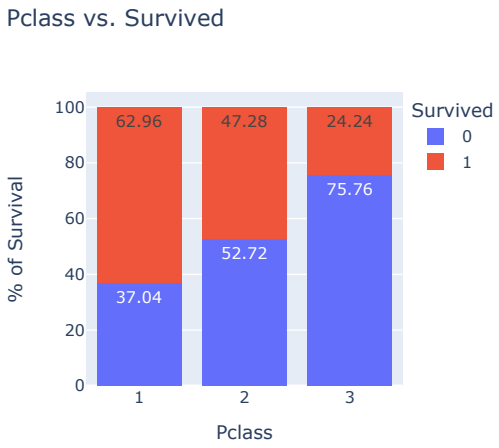
```
Y.Survived.unique()
```

```
array([0, 1])
```

```
fig = px.histogram( titanic ,
                    x = 'Sex' ,
                    color = 'Survived' ,
                    width = 400,
                    height = 400,
                    title = 'Sex vs. Survived',
                    text_auto = '.2f',
                    barnorm = 'percent'
                   )

fig.update_layout(yaxis_title = '% of Survival')

fig.show()
```

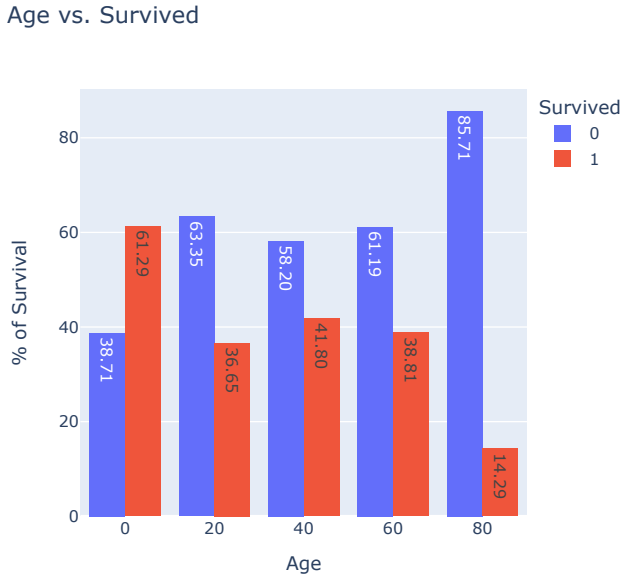



```
fig = px.histogram(titanic ,
                    x = 'Pclass' ,
                    color = 'Survived' ,
                    nbins = 3,
                    width = 400,
                    height = 400,
                    title = 'Pclass vs. Survived',
                    text_auto = '.2f',
                    barnorm = 'percent')

fig.update_layout( bargap = 0.2 , yaxis_title = '% of Survival')
fig.show()
```




```
fig = px.histogram(titanic ,
                   x = 'Fare' ,
                   color = 'Survived' ,
                   nbins = 5,
                   width = 500,
                   height = 500,
                   title = 'Fare vs. Survived',
                   text_auto = '.2f',
                   barnorm = 'percent',
                   barmode = 'group')

fig.update_layout(yaxis_title = '% of Survival')

fig.show()
```

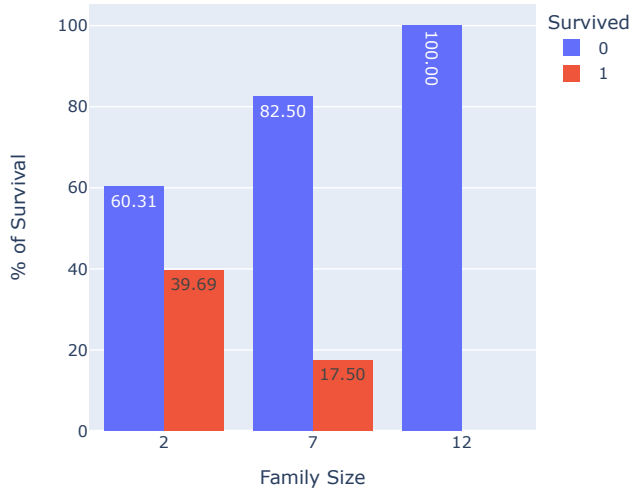

```
fig = px.histogram(titanic ,
                   x = 'Age' ,
                   color = 'Survived' ,
                   nbins = 5,
                   width = 500,
                   height = 500,
                   title = 'Age vs. Survived',
                   text_auto = '.2f',
                   barnorm = 'percent',
                   barmode = 'group' )

fig.update_layout(yaxis_title = '% of Survival')

fig.show()
```



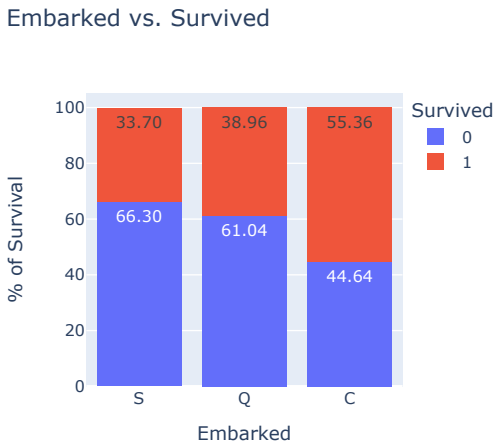

```
family_size = X.SibSp + X.Parch

family_size.unique()
```

```
array([ 1,  0,  4,  2,  6,  5,  3,  7, 10])
```

```
fig = px.histogram(titanic ,
                   x = family_size ,
                   color = 'Survived' ,
                   nbins = 4,
                   width = 500,
                   height = 500,
                   title = 'Family Size vs. Survived',
                   text_auto = '.2f',
                   barnorm = 'percent',
                   barmode = 'group',
                   labels = {'x' : 'Family Size' , })

fig.update_layout(yaxis_title = '% of Survival')


fig.show()
```



```
fig = px.histogram(titanic ,
                   x = 'Embarked' ,
                   color = 'Survived' ,
                   width = 400,
                   height = 400,
                   barnorm = 'percent',
                   text_auto = '.2f',
                   title = 'Embarked vs. Survived')


fig.update_layout(yaxis_title = '% of Survival')


fig.show()
```




```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       714 non-null    float64
 3   SibSp     891 non-null    int64  
 4   Parch     891 non-null    int64  
 5   Fare      891 non-null    float64
 6   Embarked  889 non-null    object 
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

```
print('% of missing values')
X.isnull().mean()*100
```

```
% of missing values
Pclass       0.000000
Sex          0.000000
Age         19.865320
SibSp        0.000000
Parch        0.000000
Fare         0.000000
Embarked     0.224467
dtype: float64
```

```
meanAge = X['Age'].mean()

meanAge
```

```
29.69911764705882
```

```
X['Age'].fillna( meanAge , inplace = True)


X.isna().sum()
```

```
Pclass      0
Sex         0
```

```
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

```
modeEmbarked = X['Embarked'].mode()[0]

modeEmbarked
```

```
'S'
```

```
X['Embarked'].fillna( modeEmbarked , inplace = True)
X.isna().sum()
```

```
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

```
X.Sex.unique()
X.Embarked.unique()
```

```
array(['S', 'C', 'Q'], dtype=object)
```

```
from sklearn import preprocessing

# fit the 'Sex' attribute for label encoding

label_encoder_Sex = preprocessing.LabelEncoder().fit(X['Sex'])
X['Sex'] = label_encoder_Sex.transform(X['Sex'])
```

```
X.head(5)
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	S
1	1	0	38.0	1	0	71.2833	C
2	3	0	26.0	0	0	7.9250	S
3	1	0	35.0	1	0	53.1000	S
4	3	1	35.0	0	0	8.0500	S

```
label_encoder_Embarked = preprocessing.LabelEncoder().fit(X['Embarked'])
X['Embarked'] = label_encoder_Embarked.transform(X['Embarked'])
```

```
X.head(5)
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	71.2833	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2

```
X.corrwith(Y.Survived, method='pearson')
```

```
Pclass     -0.338481
Sex        -0.543351
Age        -0.069809
SibSp      -0.035322
Parch       0.081629
Fare        0.257307
Embarked   -0.167675
dtype: float64
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.25,
                                                   random_state = 35)
X_train
```

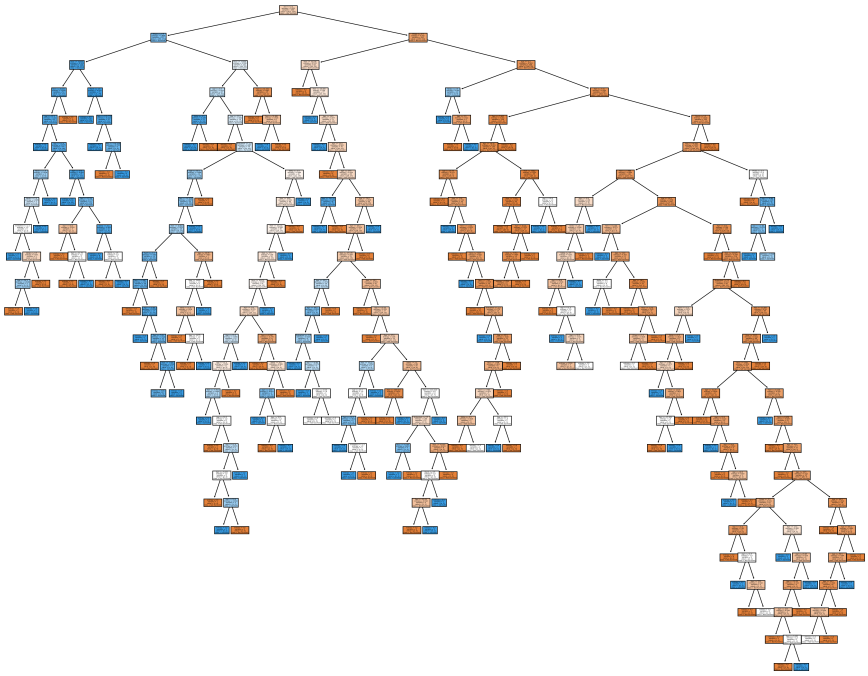
	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
786	3	0	18.0	0	0	7.4958	2
636	3	1	32.0	0	0	7.9250	2
401	3	1	26.0	0	0	8.0500	2
811	3	1	39.0	0	0	24.1500	2
780	3	0	13.0	0	0	7.2292	0
...	...	...	...	...	...	...	...
249	2	1	54.0	1	0	26.0000	2
448	3	0	5.0	2	1	19.2583	0
33	2	1	66.0	0	0	10.5000	2
271	3	1	25.0	0	0	0.0000	2
713	3	1	29.0	0	0	9.4833	2

668 rows × 7 columns

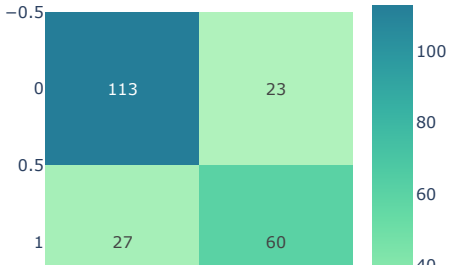
```
from sklearn.tree import DecisionTreeClassifier

# creating an object of KNeighborsClassifier class
dtc = DecisionTreeClassifier( criterion = 'entropy' , random_state = 1 )

# train the  model
dtc.fit(X_train, Y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=1)
```

```
from sklearn import tree

fig = plt.figure( figsize = (25 , 20) , dpi = 200.0)

_ = tree.plot_tree( dtc,
                   feature_names = ['Pclass' ,'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked'],
                   class_names = ['Not Survived' , 'Survived'] ,
                   filled = True)
```



```
Y_pred = dtc.predict(X_test)


from sklearn.metrics import confusion_matrix , accuracy_score

# make a confusion matrix
cm = confusion_matrix(Y_test, Y_pred)


# display confusion matrix as a heatmap
fig = px.imshow(cm ,
                width = 400,
                height = 400 ,
                text_auto = True,
                color_continuous_scale = 'tealgrn')

fig.show()


# compute and display the the accuracy of the KNN model
print('The % of Accuracy is : {0:.2f} '.format(accuracy_score(Y_test , Y_pred)*100))
```


−0.5
0
0.5
1
113
23
27
60
100
80
60