In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot

In [2]: titanic_dataset.read_csv('titanic_train.csv')

In [3]: len(titanic_data)

Out[3]: 891

In [4]: titanic_data.head()

Out[4]:

	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

In [5]: titanic_data.index

Out[5]: RangeIndex(start=0, stop=891, step=1)

In [6]: titanic_data.columns

Out[6]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
dtype='object')

In [7]: titanic_data.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 [8]: titanic_data.dtypes

Out[8]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

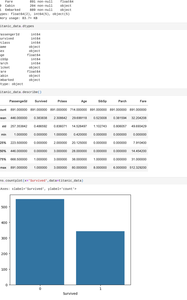
In [9]: titanic_data.describe()

Out[9]:

	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 [10]: sns.countplot('Survived',data=titanic_data)

Out[10]: <bokeh: ('Survived', plabel='count')>



In [11]: sns.countplot('Survived',data=titanic_data,hue='Sex')

Out[11]: <bokeh: ('Survived', plabel='count')>

In [12]: titanic_data.isnull()

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

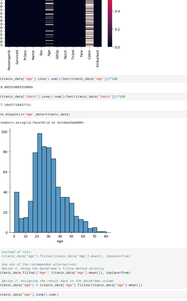
In [13]: titanic_data.isnull.sum()

Out[13]:

```
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 [14]: sns.heatmap(titanic_data.isnull())

Out[14]: <bokeh: >



In [15]: titanic_data['Age'].isnull.sum()/len(titanic_data['Age'])*100

Out[15]: 19.865319865319865

In [16]: titanic_data['Cabin'].isnull.sum()/len(titanic_data['Cabin'])*100

Out[16]: 77.104377104377

In [17]: sns.distplot(df['Age',dataf=titanic_data)

Out[17]: <seaborn.axisgrid.FacetGrid at 0x1d8ad5a8d90>

In [18]: # Instead of this:
# titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
# Use one of the recommended alternatives:
# titanic_data['Age'] = titanic_data['Age'].fillna(...)
titanic_data.fillna({'Age': titanic_data['Age'].mean()}, inplace=True)
# Or:
# Option 2: Assigning the result back to the dataframe column
titanic_data['Age'] = titanic_data['Age'].fillna(titanic_data['Age'].mean)

In [19]: titanic_data['Age'].isnul.sum()

Out[19]: 0

In [20]: sns.heatmap(titanic_data.isnu())

Out[20]: <bokeh: >

In [21]: titanic_data.drop('Cabin',axis=1,inplace=True)

In [22]: titanic_data.head()

Out[22]:

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

In [23]: titanic_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7 KB
```

In [24]: titanic_data.dtypes

Out[24]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Embarked        object
dtype: object
```

In [25]: gender=pd.get_dummies(titanic_data['Sex'],drop_first=True)

In [26]: titanic_data['Gender']=gender

In [27]: titanic_data.head()

Out[27]:

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

In [28]: titanic_data.drop(['Name','Sex','Ticket','Embarked'],axis=1,inplace=True)

In [29]: titanic_data.head()

Out[29]:

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

In [30]: titanic_data.drop('PassengerId',axis=1,inplace=True)
# Or:
# titanic_data = titanic_data[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Gender']]
y=titanic_data['Survived']

In [31]:
y

Out[31]:

```
0    0
1    1
2    1
3    1
4    0
    ..
886  0
887  1
888  0
889  1
890  0
Name: Survived, Length: 891, dtype: int64
```

In [32]: from sklearn.model_selection import train_test_split

In [33]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)

In [34]: from sklearn.linear_model import LogisticRegression

In [35]: lrlogisticRegression()

```
C:\Users\Jatin\Anaconda3\Python\Python37\site-packages\sklearn\linear_model\_logistic.py:469: 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 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(
```

Out[35]:

```
LogisticRegression()
```

In [36]: lrlogisticRegression()

Out[36]:

```
LogisticRegression()
```

In [37]: predictlr= predictlr.fit()

In [38]: from sklearn.metrics import confusion_matrix

In [39]: pd.DataFrame(confusion_matrix(y_test,predictlr),columns=['Predicted No','Predicted Yes'],index=['Actual No','Actual Yes'])

Out[39]:

	Predicted No	Predicted Yes
Actual No	117	14
Actual Yes	39	52

In [40]: from sklearn.metrics import classification_report

In [41]: print(classification_report(y_test,predictlr))

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.83       131
           1       0.79      0.57      0.66        91

    accuracy                           0.78       222
   macro avg       0.79      0.73      0.74       222
weighted avg       0.78      0.78      0.776      222
```

In [42]: from sklearn.preprocessing import StandardScaler
#setting train set small (Survived[:300])
#setting test set small (Survived[300:])
scaler=StandardScaler()
x_train=scaler.fit_transform(x)
y_train=y.to_numpy

In [43]: from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
clf = RandomForestClassifier()
param_grid={'n_estimators':[10, 100, 200, 300],'max_depth':[None, 5, 10],'min_samples_split':[2,5,8]}

y_data = x.shape
y_data = y.shape(y_data, y_data)

grid_search=GridSearchCV(clf, param_grid, cv=3, scoring="accuracy", return_train_score=True)
grid_search.fit(x_data, y_data)

In [44]: from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

1. Prepare the data
Assuming x_data and y_data are already defined
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

2. Create and train the model
rfc = RandomForestClassifier(n_estimators=100, random_state=42)
rfc.fit(x_train, y_train)

# 3. Evaluate the model
# y_pred = rfc.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

4. (Optional) Hyperparameter Tuning
You can perform hyperparameter tuning to improve the model's performance

5. (Optional) Deploy the model
Once satisfied with the model's performance, you can deploy it for making predictions on new data

Import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

Instantiate the classifier

rfc = RandomForestClassifier(random_state=0)

Fit the model

rfc.fit(x_train, y_train)

Predict the test set results

y_pred = rfc.predict(x_test)

Check accuracy score

from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

Model accuracy score with 10 decision-trees : 0.8227

In [45]: rfc_100 = RandomForestClassifier(n_estimators=100, random_state=0)

Fit the model to the training set

rfc_100.fit(x_train, y_train)

Predict on the test set results

y_pred_100 = rfc_100.predict(x_test)

Check accuracy score

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred_100)))

Model accuracy score with 100 decision-trees : 0.8227

In [46]: # create the classifier with n_estimators = 100

clf = RandomForestClassifier(n_estimators=100, random_state=0)

Fit the model to the training set

clf.fit(x_train, y_train)

Out[46]:

```
RandomForestClassifier(random_state=0)
```

In [47]: feature_scores = pd.Series(clf.feature_importances_, index=x_train.columns).sort_values(ascending=False)

```
Out[47]: Feature_scores

Gender      0.244842
Age         0.239104
PassengerID 0.200422
Age         0.178679
Fare        0.100000
SibSp       0.040953
```

```
Kurch          0.000000
dtype: float64
```