Import Modules

```
# Import necessary modules

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
```

Reading Dataset

```
# Import the titanic train dataset as a Pandas dataframe

df = pd.read_csv("titanic_train.csv")
df.head()
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	bo
0	1216	3	Smyth, Miss. Julia	female	NaN	0	0	335432	7.7333	NaN	Q	13	N
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625	NaN	S	NaN	N
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	1	345773	24.1500	NaN	S	NaN	N

```
# Since the model takes the input as the person's age, gender and fare: discard the rest of the attributes

df.drop(['passenger_id','pclass','name','sibsp','parch','ticket','cabin','embarked','boat','body','home.dest'],axis='columns',inplace
df
```

	sex	age	fare	survived
0	female	NaN	7.7333	1
1	male	38.0	8.6625	0
2	female	30.0	24.1500	0
3	female	54.0	23.0000	1
4	male	40.0	13.0000	0
...	...	...	...	...
845	male	55.0	50.0000	0
846	male	58.0	29.7000	0
847	female	24.0	26.0000	1
848	female	3.0	13.7750	0
849	male	52.0	13.0000	0

850 rows × 4 columns

Data Preprocessing for Training Dataset

> Since the training dataset has string values and empty values, we need to format it before using it to build our model

```
# For the input dataframe, convert the string values to integer.
# (could also be done using LabelEncoder)
```

```
inputs = df.drop('survived', axis='columns')
inputs.sex = inputs.sex.map({'male': 1, 'female': 2})
inputs
```

	sex	age	fare
0	2	NaN	7.7333
1	1	38.0	8.6625
2	2	30.0	24.1500
3	2	54.0	23.0000
4	1	40.0	13.0000
...	...	...	...
845	1	55.0	50.0000
846	1	58.0	29.7000
847	2	24.0	26.0000
848	2	3.0	13.7750
849	1	52.0	13.0000

850 rows × 3 columns

```
# Check whether there are null values present

inputs.isna().sum()
```

```
sex       0
age     174
fare      1
dtype: int64
```

```
# Since there are null values are present for age and fare,
# fill the empty values with the mean of the existing data.

inputs.age = inputs.age.fillna(inputs.age.mean())
inputs.fare = inputs.fare.fillna(inputs.fare.mean())
inputs
```

	sex	age	fare
0	2	29.519847	7.7333
1	1	38.000000	8.6625
2	2	30.000000	24.1500
3	2	54.000000	23.0000
4	1	40.000000	13.0000
...	...	...	...
845	1	55.000000	50.0000
846	1	58.000000	29.7000
847	2	24.000000	26.0000
848	2	3.000000	13.7750
849	1	52.000000	13.0000

850 rows × 3 columns

```
# Create the target dataframe

target = df.survived
target
```

```
0      1
1      0
2      0
```

```
3      1
4      0
      ..
845    0
846    0
847    1
848    0
849    0
Name: survived, Length: 850, dtype: int64
```

Building the Model

```
# Split the dataset into training dataset and test data for building the model

X_train, X_test, y_train, y_test = train_test_split(inputs.values,target,test_size=0.2)
```

```
#Create the model using the Decision Tree Classifier

model = tree.DecisionTreeClassifier()
```

```
# To display the settings of the classifier function

from sklearn import set_config
set_config(print_changed_only=False)
```

```
# Fit the dataset onto the model

model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       random_state=None, splitter='best')
```

Prediction and Accuracy

```
# Predict a sample data entry

model.predict([[2,30,24.15]])
```

```
array([0])
```

```
# Display the score of the model

model.score(X_test, y_test)
```

```
0.788235294117647
```

Testing of given Test Data

> Since the test dataset has string values and empty values, we need to format it before using it to test our model

```
# Import the test data as a Pandas dataframe

df_test = pd.read_csv("titanic_test.csv")
df_test
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	b
0	295	1	Thayer, Mr. John Borland Jr	male	17.0	0	2	17421	110.8833	C70	C	
1	1150	3	Risien, Mr. Samuel Beard	male	NaN	0	0	364498	14.5000	NaN	S	
2	89	1	Davidson, Mr. Thornton	male	31.0	1	0	F.C. 12750	52.0000	B71	S	
3	1063	3	Nirva, Mr. Iisakki Antino Aijo	male	41.0	0	0	SOTON/O2 3101272	7.1250	NaN	S	
4	1020	3	Minkoff, Mr. Lazar	male	21.0	0	0	349211	7.8958	NaN	S	

```
# Create a separate dataframe for the given Passenger IDs

p_id = df_test['passenger_id']
p_id
```

```
0       295
1      1150
2        89
3      1063
4      1020
       ...
454    1194
455     403
456     108
457     510
458    1265
Name: passenger_id, Length: 459, dtype: int64
```

```
# Drop the unnecessary attributes from the Test Dataframe; keep age, gender, fare

df_test.drop(['passenger_id','pclass','name','sibsp','parch','ticket','cabin','embarked','boat','body','home.dest'],axis='columns',i
df_test
```

	sex	age	fare
0	male	17.0	110.8833
1	male	NaN	14.5000
2	male	31.0	52.0000
3	male	41.0	7.1250
4	male	21.0	7.8958
...	...	...	...
454	male	NaN	7.8958
455	male	23.0	13.0000
456	female	NaN	110.8833
457	male	16.0	10.5000
458	female	10.0	24.1500

459 rows × 3 columns

Data Preprocessing for Test Dataset

```
# Convert the string values to integers

df_test.sex = df_test.sex.map({'male': 1, 'female': 2})
df_test
```

	sex	age	fare
0	1	17.0	110.8833
1	1	NaN	14.5000
2	1	31.0	52.0000
3	1	41.0	7.1250
4	1	21.0	7.8958
...	...	...	...
454	1	NaN	7.8958
455	1	23.0	13.0000
456	2	NaN	110.8833
457	1	16.0	10.5000
458	2	10.0	24.1500

459 rows × 3 columns

```
# Check for null values present

df_test.isna().sum()
```

```
sex      0
age     89
fare     0
dtype: int64
```

```
# Since there are null values are present for age,
# fill the empty values with the mean of the existing data.

df_test.age = df_test.age.fillna(df_test.age.mean())
df_test
```

	sex	age	fare
0	1	17.000000	110.8833
1	1	30.541216	14.5000
2	1	31.000000	52.0000
3	1	41.000000	7.1250
4	1	21.000000	7.8958
...	...	...	...
454	1	30.541216	7.8958
455	1	23.000000	13.0000
456	2	30.541216	110.8833
457	1	16.000000	10.5000
458	2	10.000000	24.1500

459 rows × 3 columns

Prediction using Test Data

```
# Run the model on the given test data and see the predicted output

pred = model.predict(df_test.values)
pred
```

```
array([0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
```

```
0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1])
```

```
# Store the predicted result as a dataframe

df_result = pd.DataFrame(pred, columns = ['survived_status'])
df_result
```

	survived_status
0	0

```
# Create a separate column to store the predicted
# result against the respective Passenger IDs

df_result['passenger_id'] = p_id
df_result
```

	survived_status	passenger_id
0	0	295
1	0	1150
2	0	89
3	0	1063
4	0	1020
...	...	...
454	0	1194
455	0	403
456	1	108
457	0	510
458	1	1265

459 rows × 2 columns

Accuracy of the Model

```
# Import the test dataset target attribute as a Pandas dataframe

df_check = pd.read_csv("gender_baseline.csv")
df_check
```

	passenger_id	survived
0	295	0
1	1150	0
2	89	0
3	1063	0
4	1020	0
...	...	...
454	1194	0
455	403	0
456	108	1
457	510	0
458	1265	1

459 rows × 2 columns

```
# Display the final accuracy of the model

print("Accuracy:", accuracy_score(df_check.survived, df_result.survived_status))
```

```
Accuracy: 0.8017429193899782
```

✓ 0s completed at 12:58 AM