Machine learning
한눈에 보기

MSP Korea, Kyeongwan Kang

2018. 07. 20.

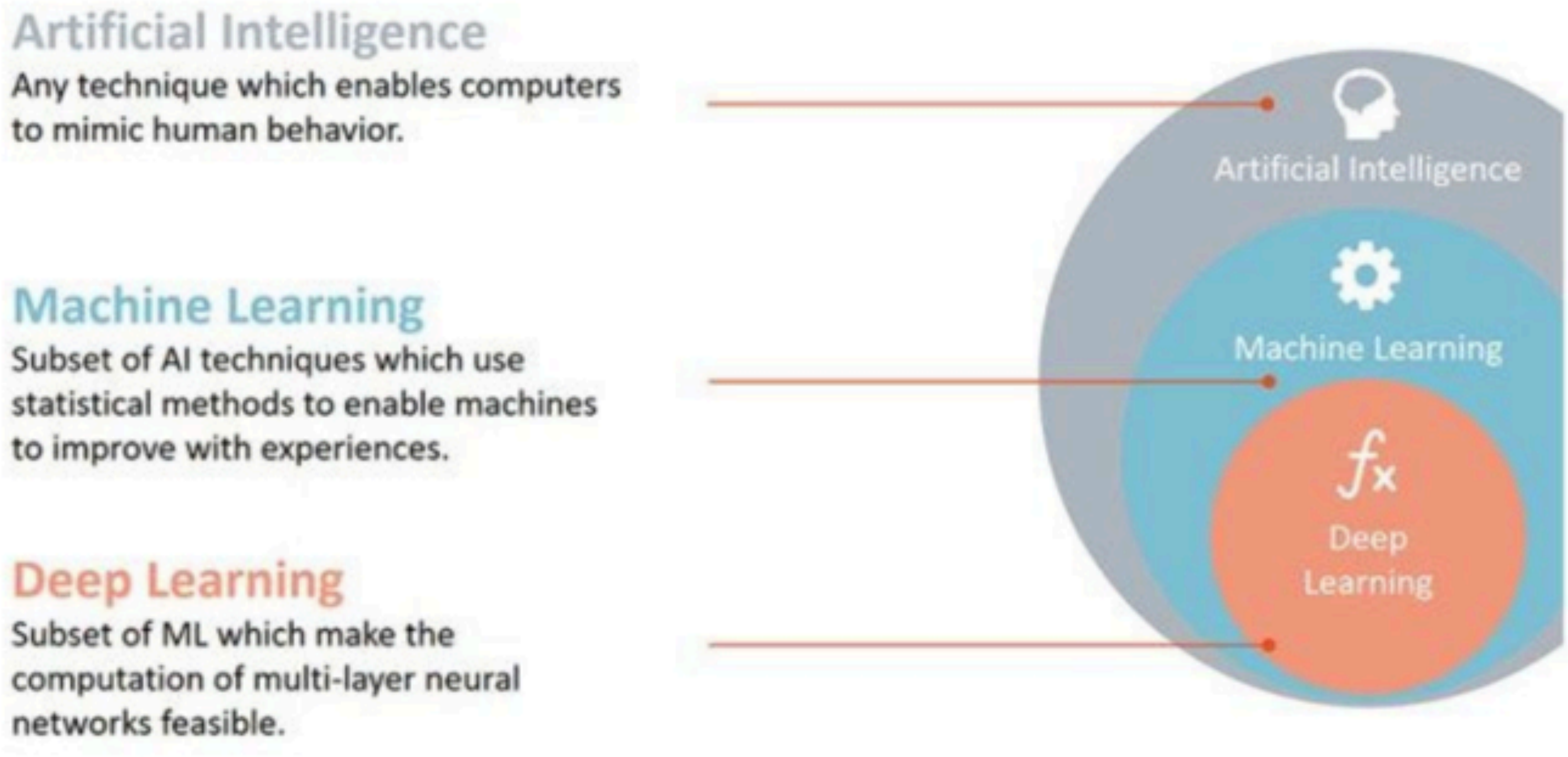
Microsoft
Microsoft Student Partner

WISET
한국여성과학기술인지원센터

1. 머신러닝

https://memegenerator.net/instance/72049399/matrix-architect-i-am-the-architect

1. 머신러닝



Artificial Intelligence
Any technique which enables computers to mimic human behavior.

Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.

https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html

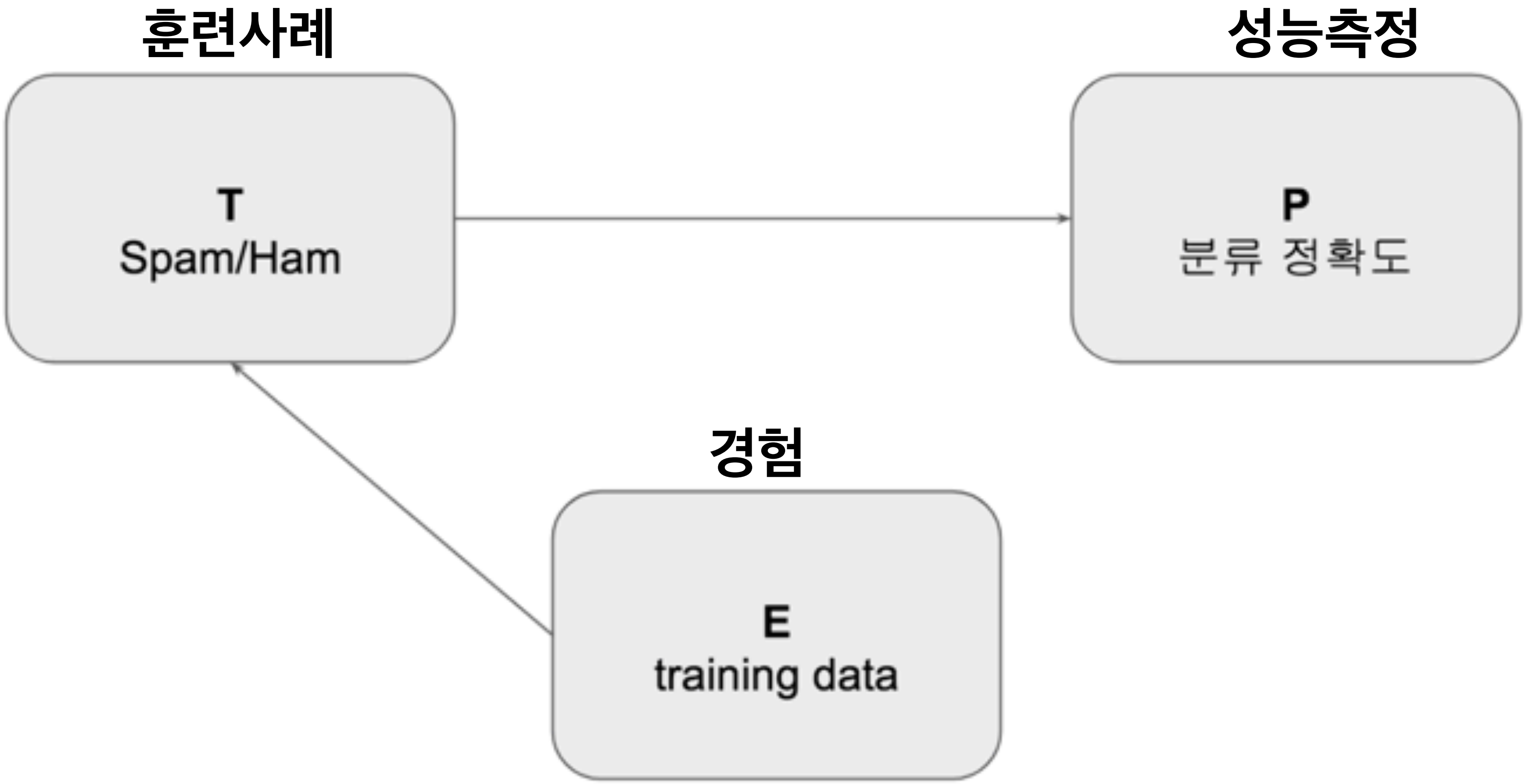
‘머신러닝은 명시적인 프로그래밍 없이
컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야다.’

–아서 아무엘, *1959*

‘어떤 작업 **T**에 대한 컴퓨터 프로그램의 성능을 **P**로 측정되었을 때
경험 **E**로 인해 성능이 향상됐다면,
이 컴퓨터 프로그램은 작업 **T**와 성능 측정 **P**에 대해 경험 **E**로 학습한 것이다.’

–톰 미첼*, 1997*

1. 머신러닝



훈련사례

T
Spam/Ham

성능측정

P
분류 정확도

경험

E
training data

1. 머신러닝

- 기존 솔루션은 많은 수동 조정과 규칙이 필요 -> 하나의 머신러닝 모델이 코드를 간단히 더 잘 수행할 수 있음

- 전통적인 방법으로 불가능 -> 머신러닝은 가능

- 유동적인 환경

- 복잡한 문제와 대량의 데이터 통찰 얻기

2. 머신러닝 시스템

- Supervised Learning (지도학습)
- Unsupervised Learning(비지도학습)
- Semi-supervised Learning (준지도학습)
- Reinforcement Learning (강화학습)
- Batch Learning (배치학습)
- Online Learning (온라인학습)
- Instance-based Learning (사례기반학습)
- Model-based Learning (모델 기반 학습)

2. 지도학습

- 알고리즘에서 사용하는 training data 에 label(=class) 라는 답이 있음 (java class 아님 주의)
- Classification and Regression (분류와 회귀)
- 신경망 (Neural Network)
- Logistic Regression
- Linear Regression

2. 지도학습

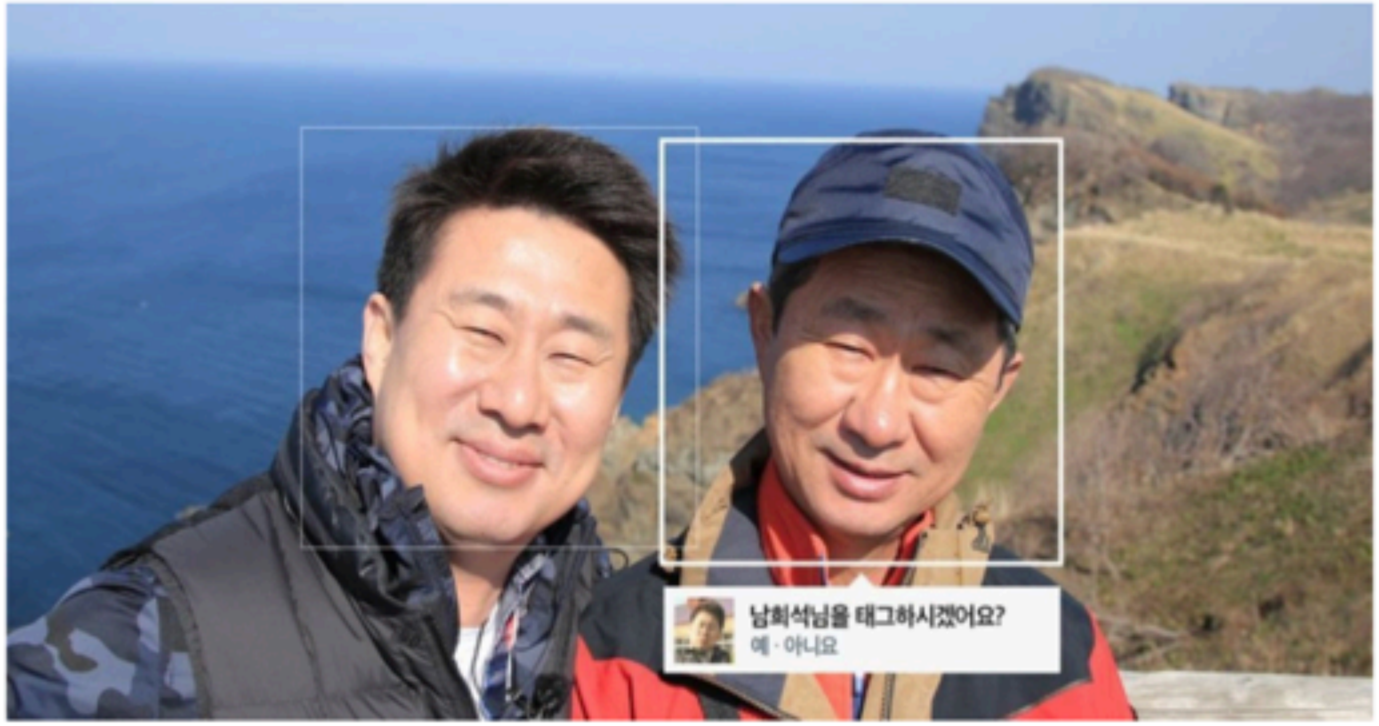
- Classification and Regression (분류와 회귀)
- 분류
    - 분류 작업이 훈련되어야 함
    - 스팸인지 아닌지 분류함
- 회귀
    - 특성(feature)
    - 분류하는 것이 아니라 예측

2. 비지도학습

- Training data에 label이 없음
- 군집화 (Clustering)
- 시각화 등
- ex) 블로그 방문자 데이터가 많이 있다
  -> 비슷한 방문자를 묶고 싶다
  -> 레이블이 없다…
  -> 알고리즘이 스스로 연결고리를 찾음

2. 준지도학습

- label이 있긴 있는데 조금, 대부분은 없음
- 지도학습과 비지도학습의 조합
- 예 : Facebook, Google Photo의 얼굴 태그



남희석님을 태그하시겠어요?

출처: 남희석 Facebook

2. 강화학습

- 알고리즘 스스로가 환경을 관찰하여 행동
- 행동을 실행한 댓가로 보상이나 벌을 받음
- 보상을 많이 받기 위한 정책을 스스로 학습
- ex) 알파고

2. 배치학습

- 가지고 있는 데이터를 모두 사용하여 훈련 시키는 방법
    - Offline Learning
    - 새로운 데이터에 점진적으로 학습 불가
    - 새로운 데이터를 학습하려면 전체 데이터를 사용하여 처음부터 다시 훈련
    - 많은 리소스가 필요, 데이터가 아주 크다면 유지도 불가능
    - 리소스가 제한된 시스템에서는 사용불가 또는 심각한 문제 발생

2. 온라인 학습

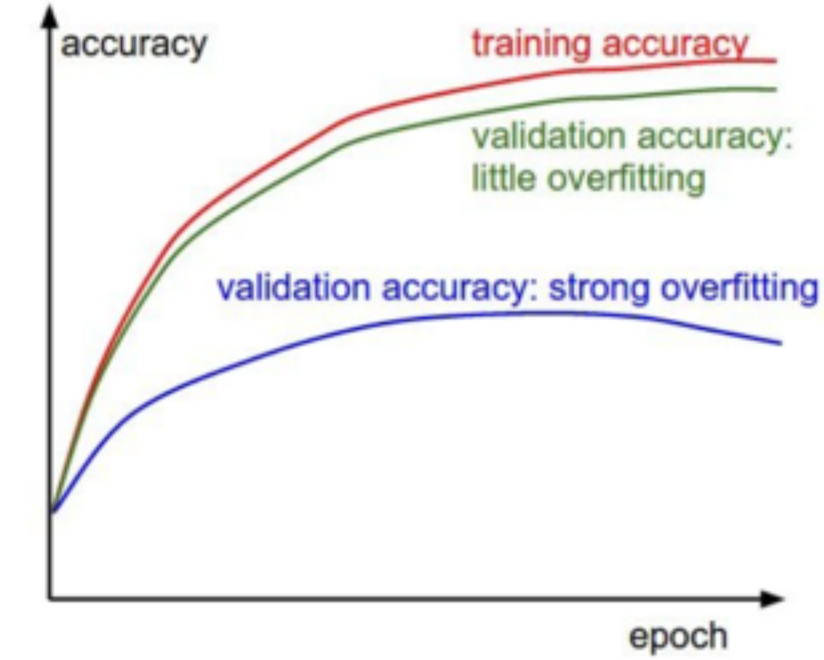
- 데이터를 순차적으로 훈련
- 새로운 데이터를 학습할 때 전체 데이터가 아닌 새로운 것만 학습하면 됨
- 문제점 : 새로운 데이터로 나쁜 데이터가 주입되었을 때 성능이 떨어짐
  ex) 챗봇의 인종차별 문제

3. 머신러닝이 풀어야 할 과제 (데이터)

- 낮은 품질의 데이터
  - 일부 샘플이 outlier를 가진다면 무시하거나 수동으로 수정 (전처리 과정)
  - 일부 샘플에 feature가 빠져있다면 무시하거나 값을 수정 (전처리 과정)
- 관련 없는 특성
  - 훈련에 사용할 좋은 특성 찾기

3. 머신러닝이 풀어야 할 과제 (알고리즘)

- Overfitting
    - 모델이 data에 너무 잘 맞아서 오히려 일반성이 떨어진다
      ex) 소개팅 어플의 결제 확률 모델
      -> 모두 결제안함 으로 예측하면 정확도가 90% 이상
    - 더 많은 data 를 집어넣고 outlier 를 제거
- Underfitting
    - 모델이 너무 단순하여 제대로 학습하지 못함



http://cs231n.github.io/neural-networks-3/

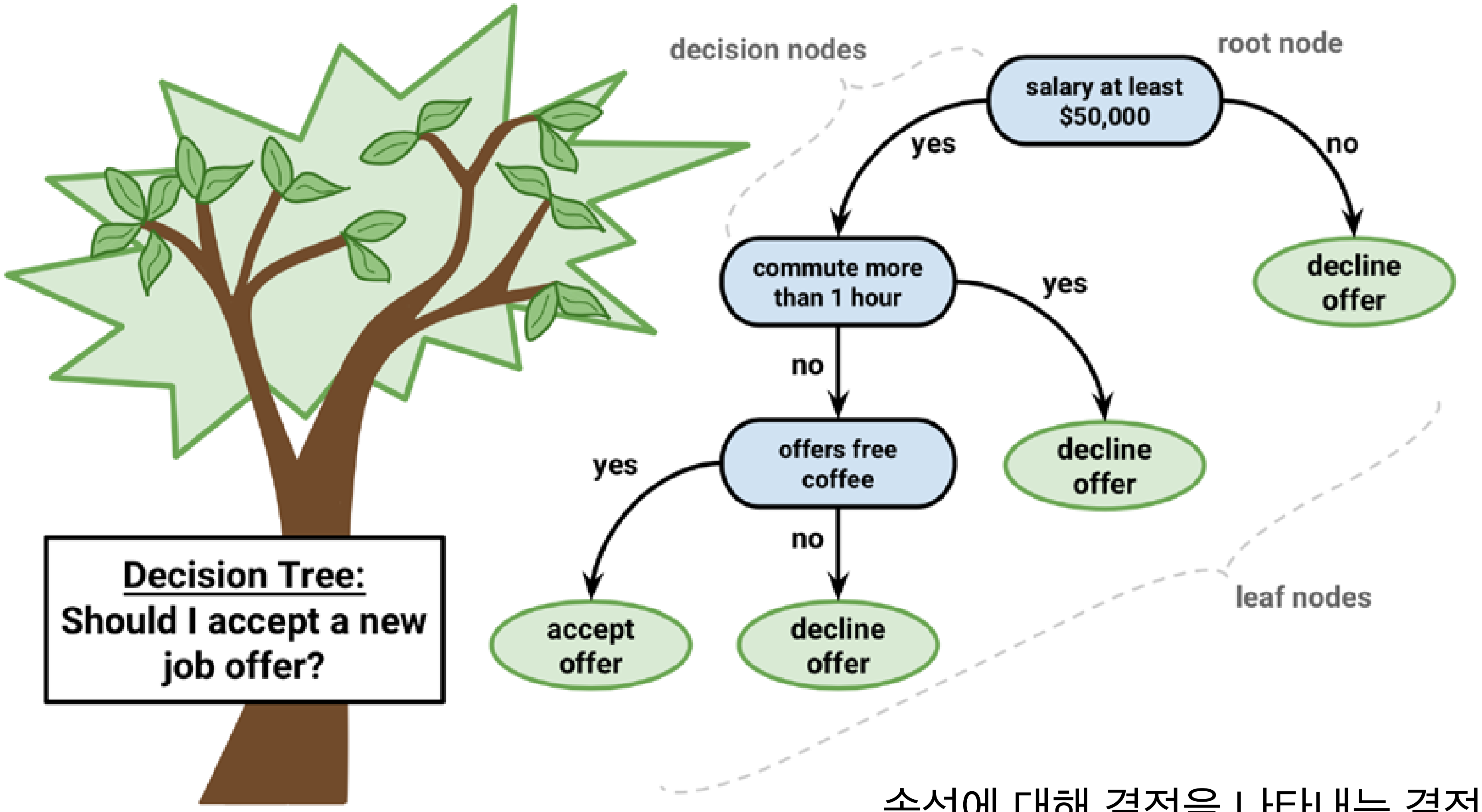
3. 테스트와 검증

- Training Set and Test Set
  - 모델이 얼마나 잘 일반화 되는지 검증
  - training set과 test set 으로 데이터를 나누어 training set으로 학습 후 test set으로 검증

그래서 오늘 실습은

결정 트리 (지도학습)

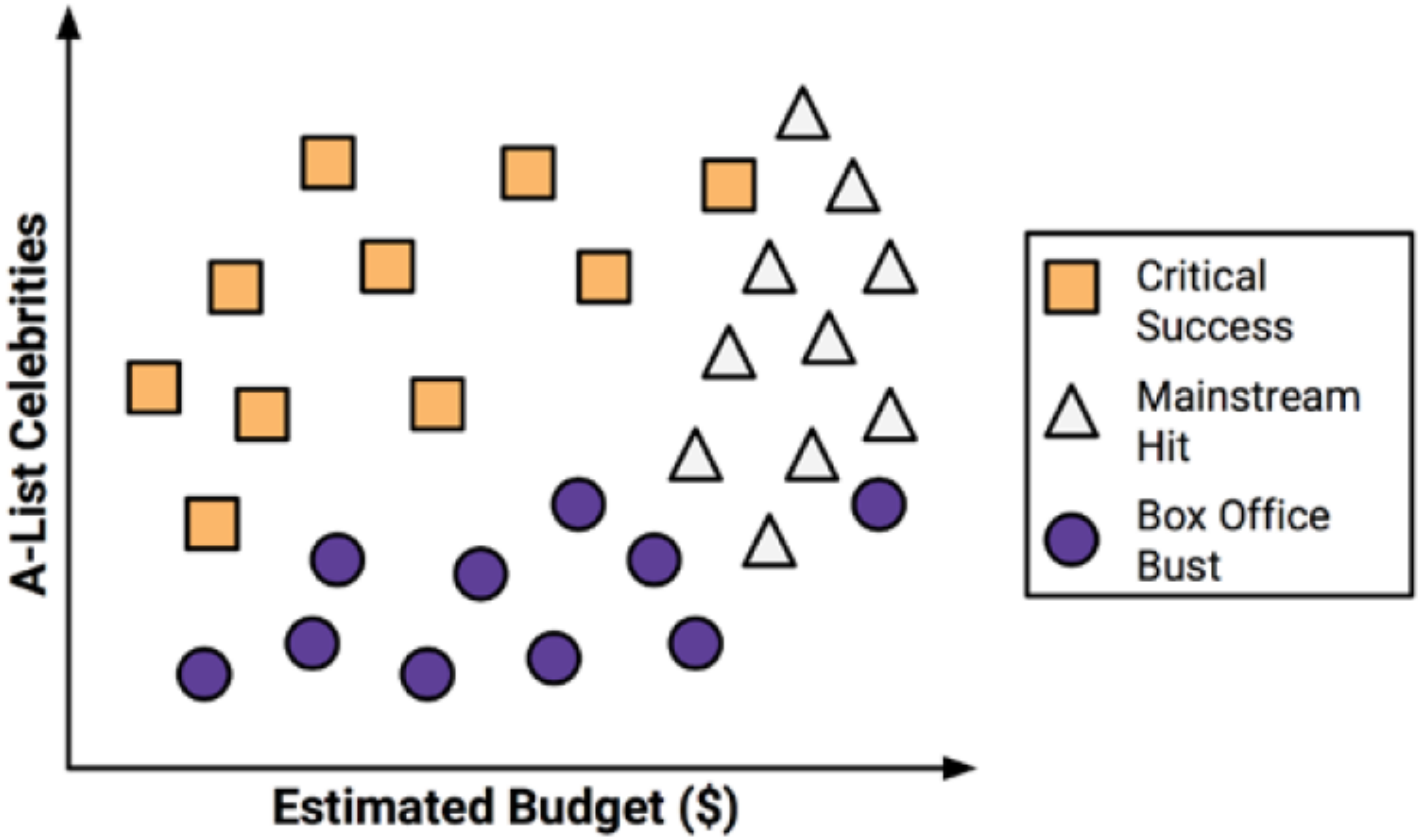
결정 트리의 형태

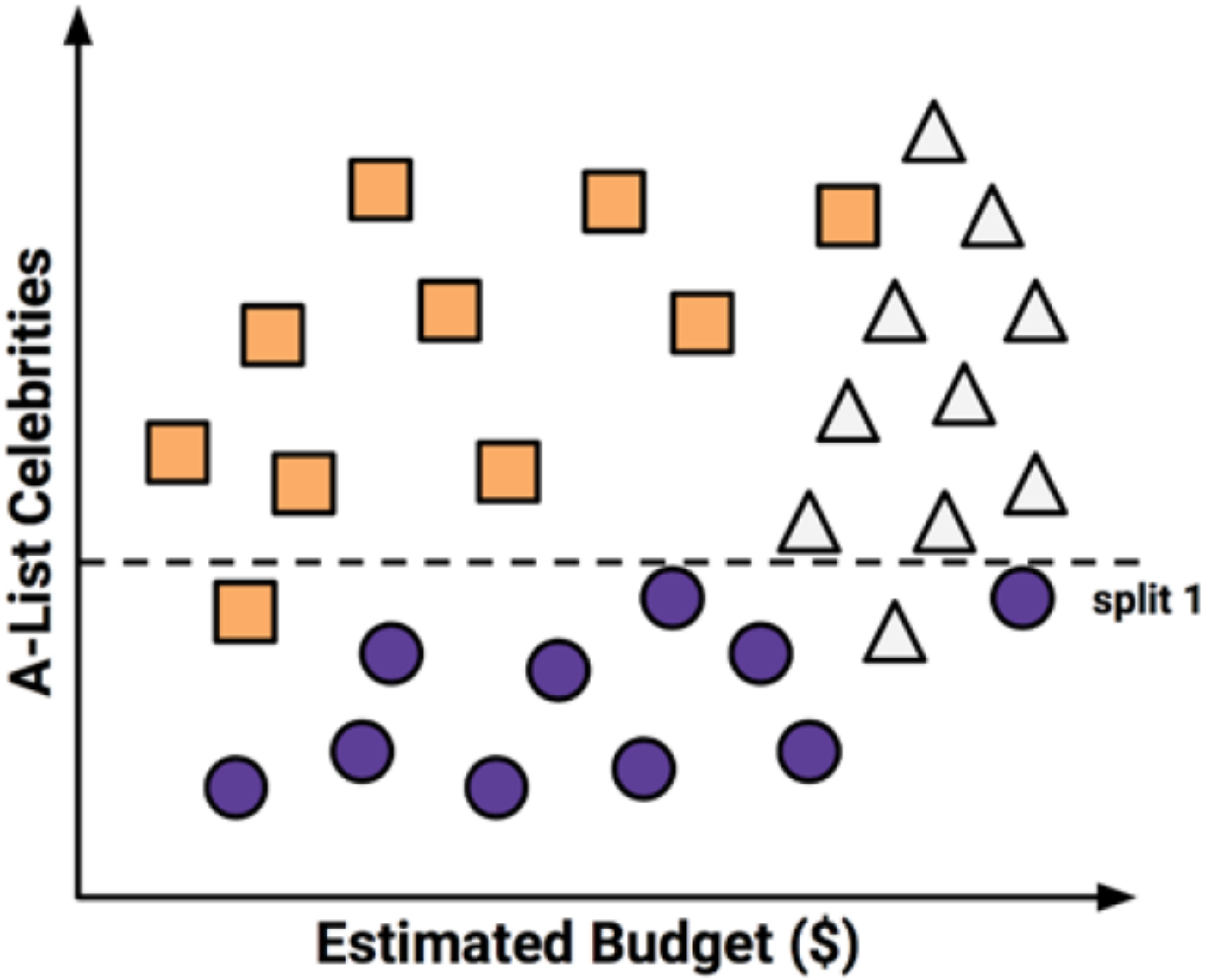


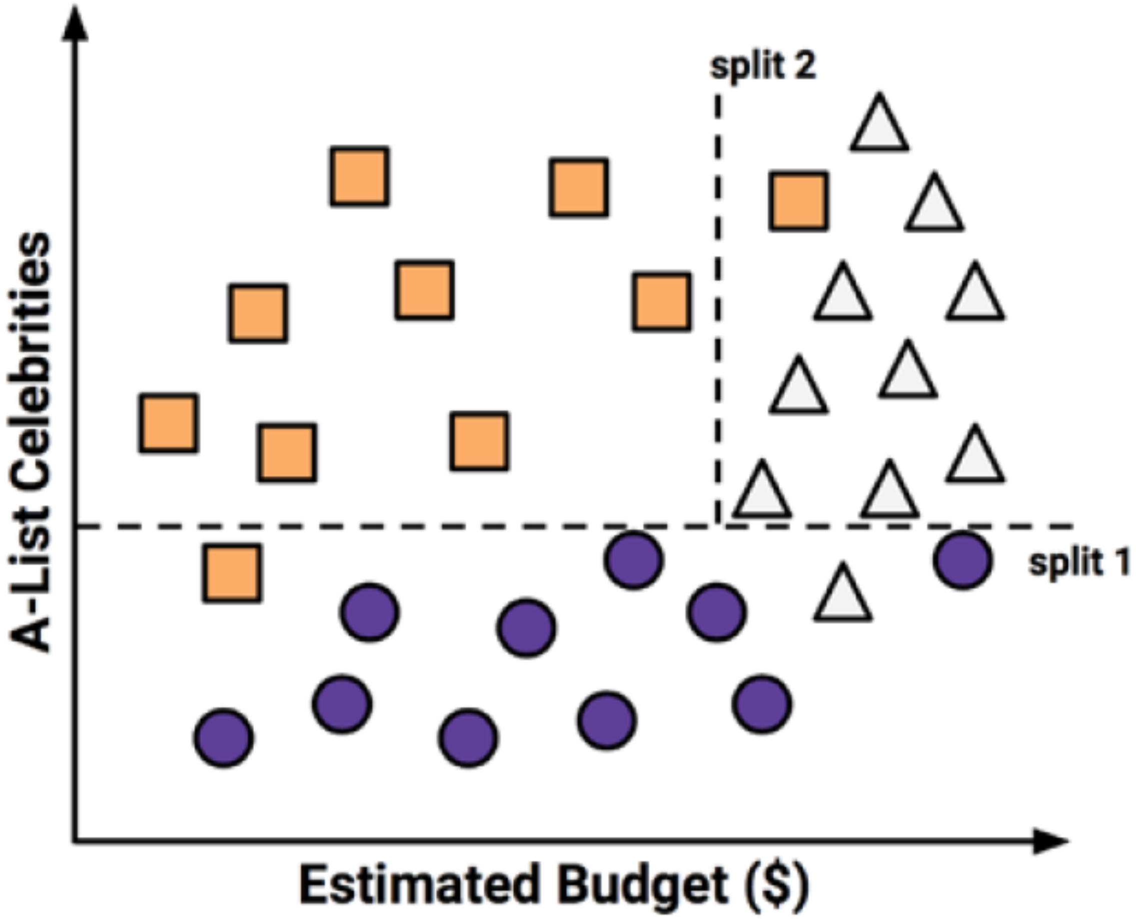
Decision Tree:
Should I accept a new job offer?

속성에 대해 결정을 나타내는 결정 노드 (decision node)로 이루어지며, 결정 노드는 속성에 따라 결정하는 가지들(branches)로 나눠짐

분할정복 (Divide and Conquer)


A-List Celebrities
Estimated Budget ($)
Critical Success
Mainstream Hit
Box Office Bust


A-List Celebrities
split 1
Estimated Budget ($)


split 2
split 1
A-List Celebrities
Estimated Budget ($)

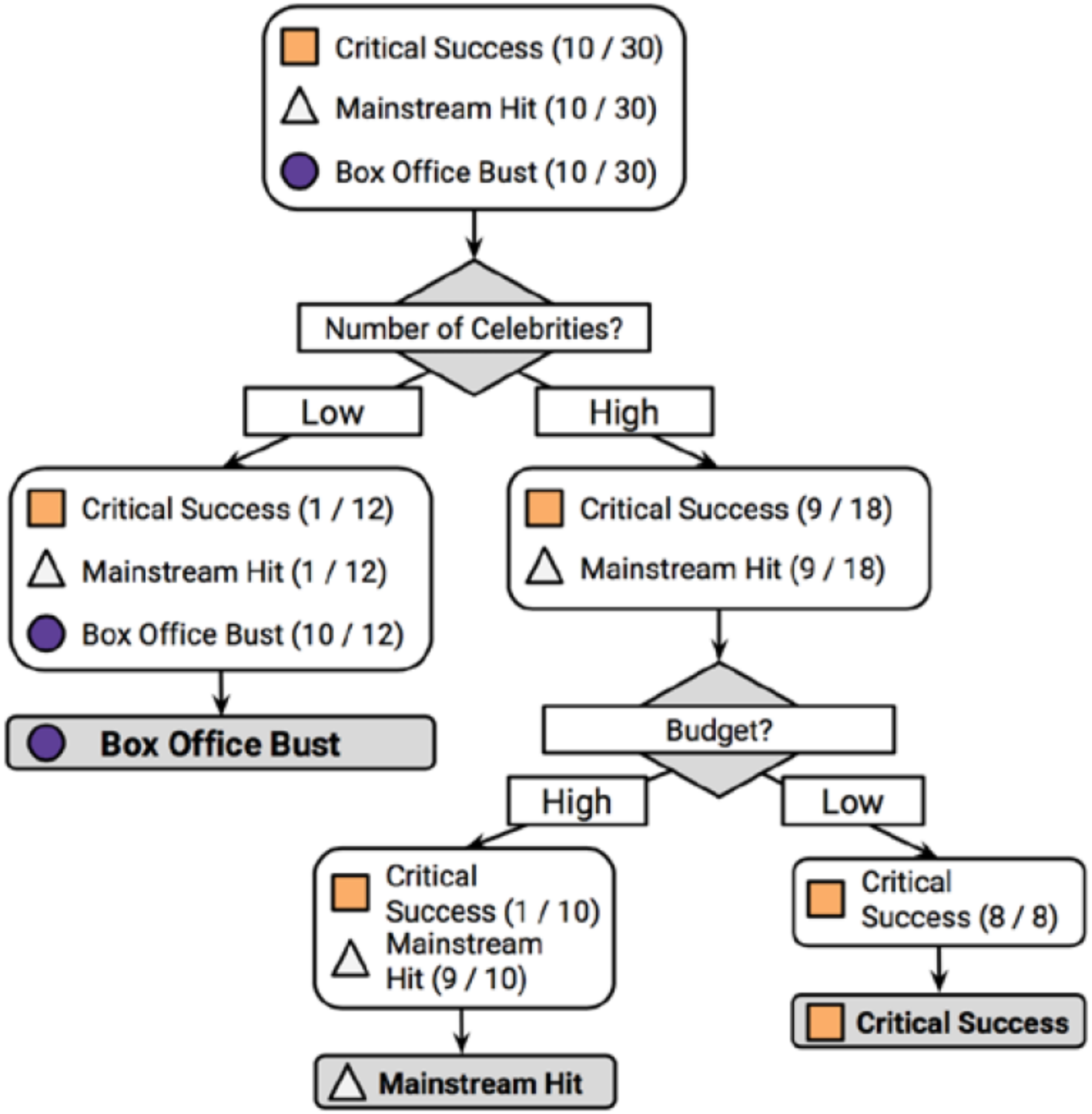
여기까지 데이터를 3개 그룹으로 구별함

왼쪽 위 그룹은 비평가의 평가가 좋은 영호이며, 많은 영화배우와 상대적으로 적은 예산으로 구별됨

오른쪽 위에는 많은 예산이 들어갔으며, 많은 수의 A급 영화배우가 나오는 흥행 영화

마지막 그룹은 적은 스타 배우와 예산과 상관없이 망해버린 영화

원한다면 더 잘못된 분류된 값 까지 다양한 특정 범위의 예산과 영화배우 수를 바탕으로 계속 나눌 수 있음



🟧 Critical Success (10 / 30)
△ Mainstream Hit (10 / 30)
● Box Office Bust (10 / 30)

Number of Celebrities?

Low

High

🟧 Critical Success (1 / 12)
△ Mainstream Hit (1 / 12)
● Box Office Bust (10 / 12)

● **Box Office Bust**

🟧 Critical Success (9 / 18)
△ Mainstream Hit (9 / 18)

Budget?

High

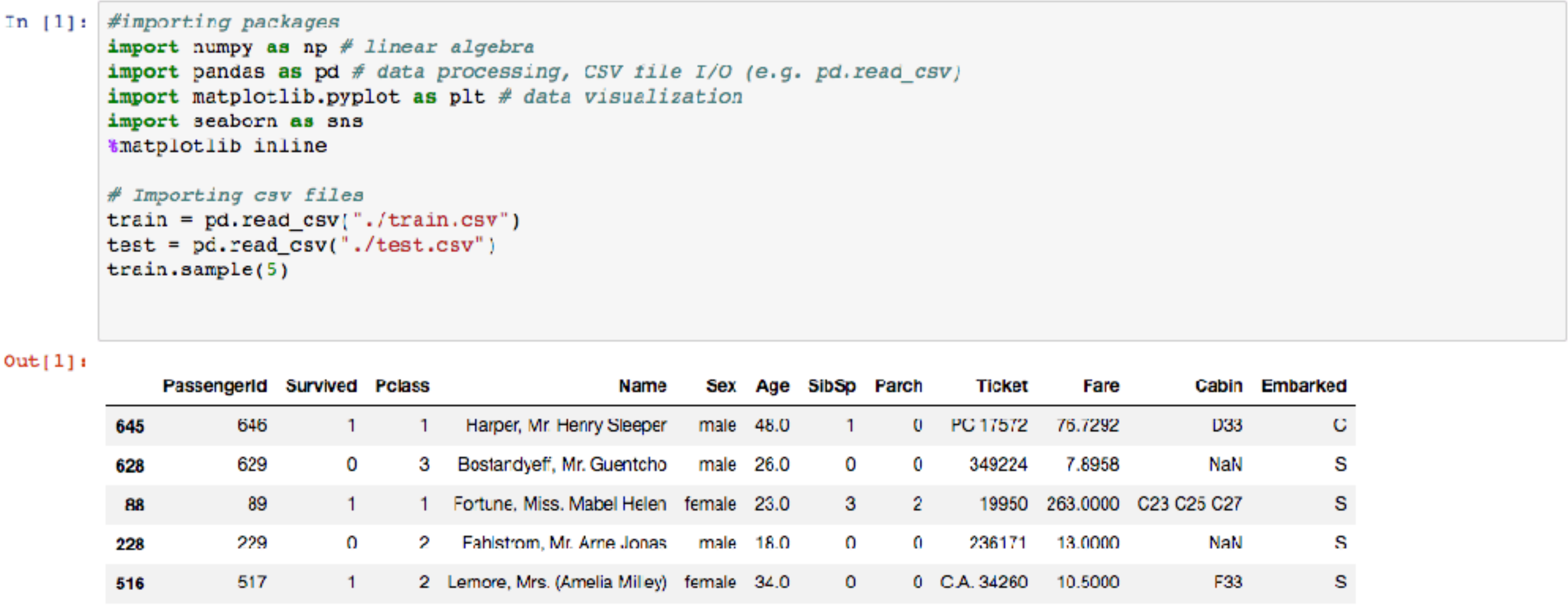
Low

🟧 Critical Success (1 / 10)
△ Mainstream Hit (9 / 10)

△ **Mainstream Hit**

🟧 Critical Success (8 / 8)

🟧 **Critical Success**

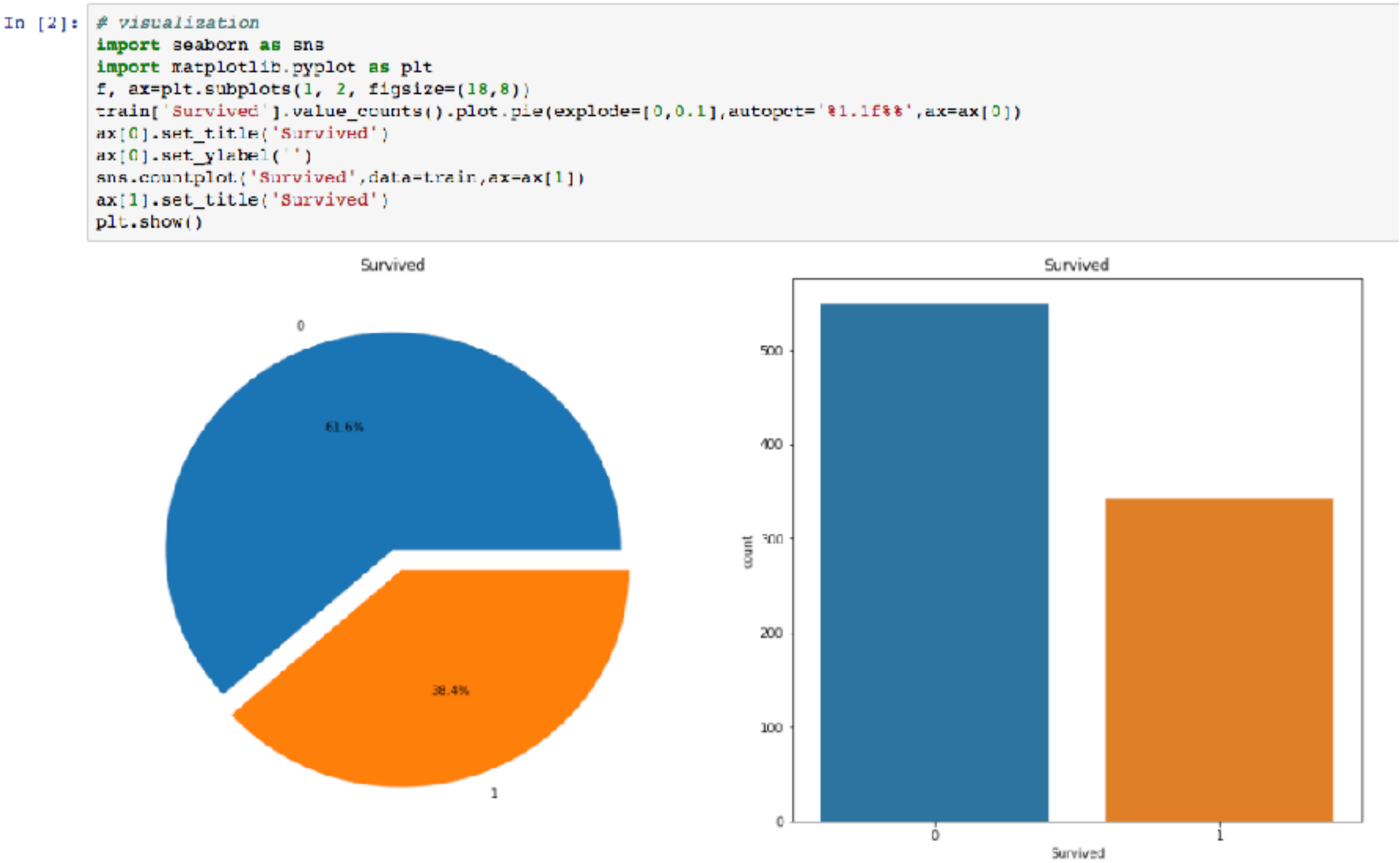


In [1]:

```python
#importing packages
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns
%matplotlib inline

# Importing csv files
train = pd.read_csv("./train.csv")
test = pd.read_csv("./test.csv")
train.sample(5)
```

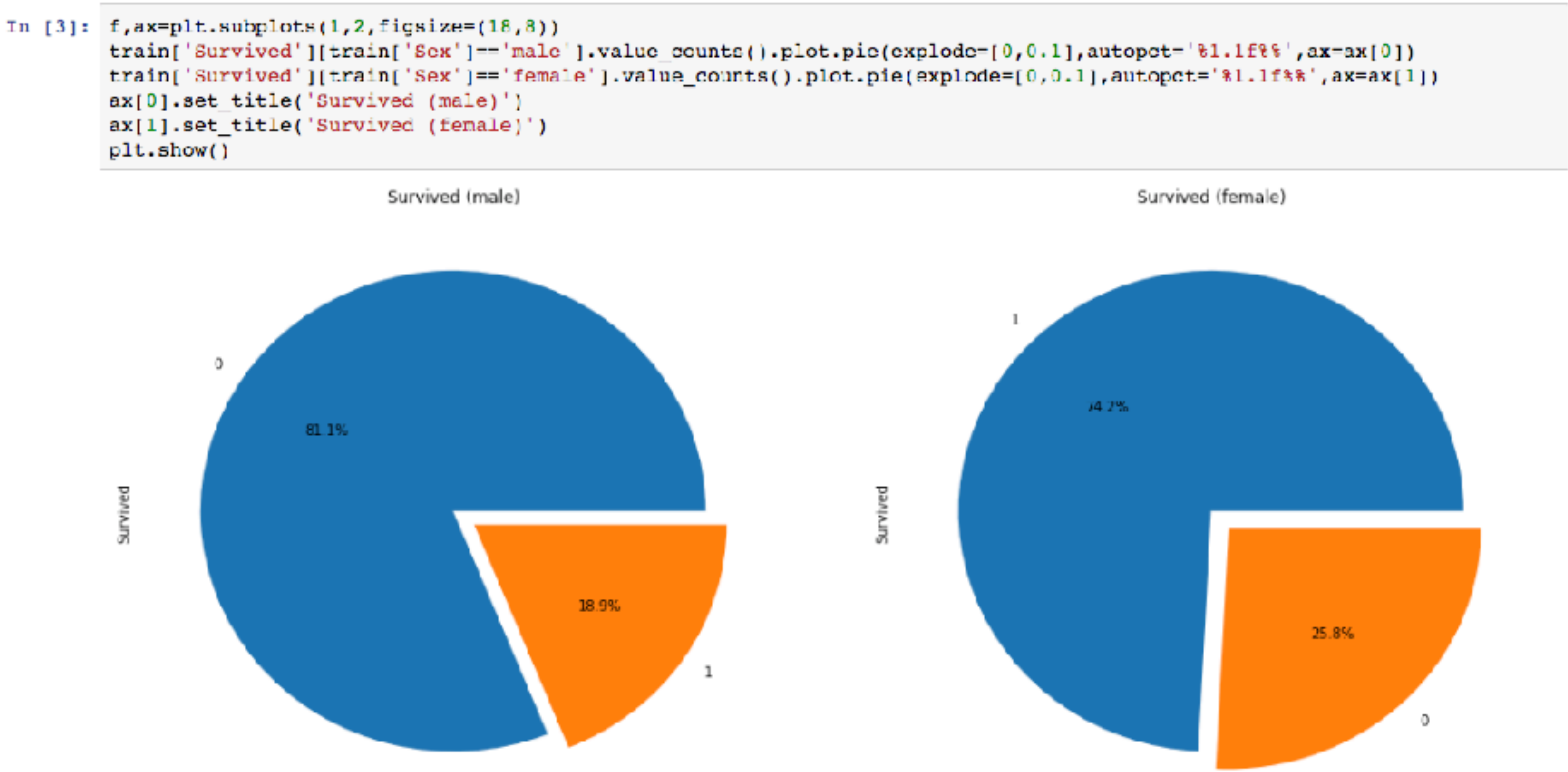
Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
645	646	1	1	Harper, Mr. Henry Sleeper	male	48.0	1	0	PC 17572	76.7292	D33	C
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224	7.8958	NaN	S
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
228	229	0	2	Fahlstrom, Mr. Arne Jonas	male	18.0	0	0	236171	13.0000	NaN	S
516	517	1	2	Lemore, Mrs. (Amelia Milley)	female	34.0	0	0	C.A. 34260	10.5000	F33	S

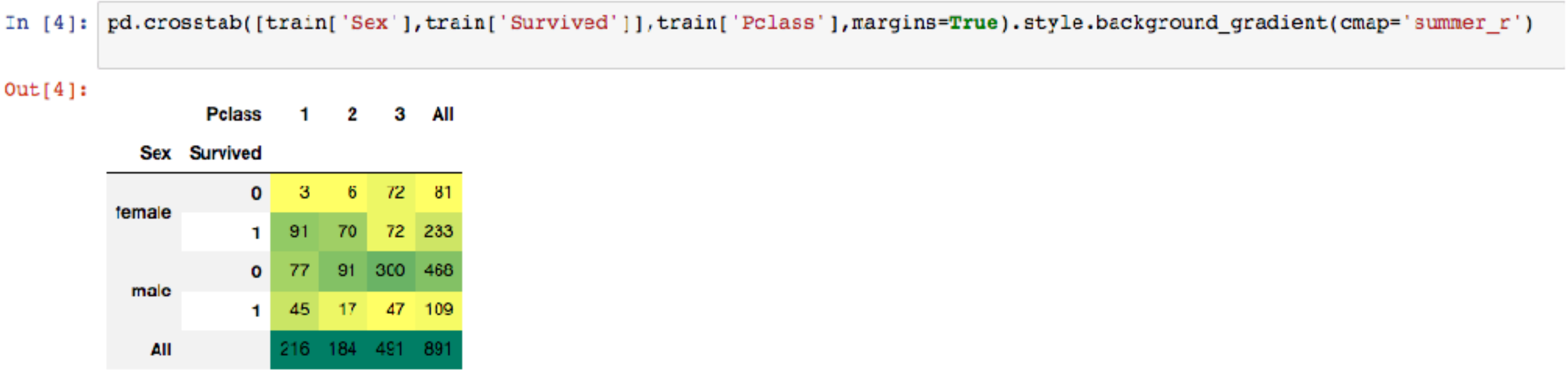


In [2]:

```
# visualization
import seaborn as sns
import matplotlib.pyplot as plt
f, ax=plt.subplots(1, 2, figsize=(18,8))
train['Survived'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0])
ax[0].set_title('Survived')
ax[0].set_ylabel('')
sns.countplot('Survived',data=train,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```



```
In [3]: f,ax=plt.subplots(1,2,figsize=(18,8))
        train['Survived'][train['Sex']=='male'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0])
        train['Survived'][train['Sex']=='female'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[1])
        ax[0].set_title('Survived (male)')
        ax[1].set_title('Survived (female)')
        plt.show()
```

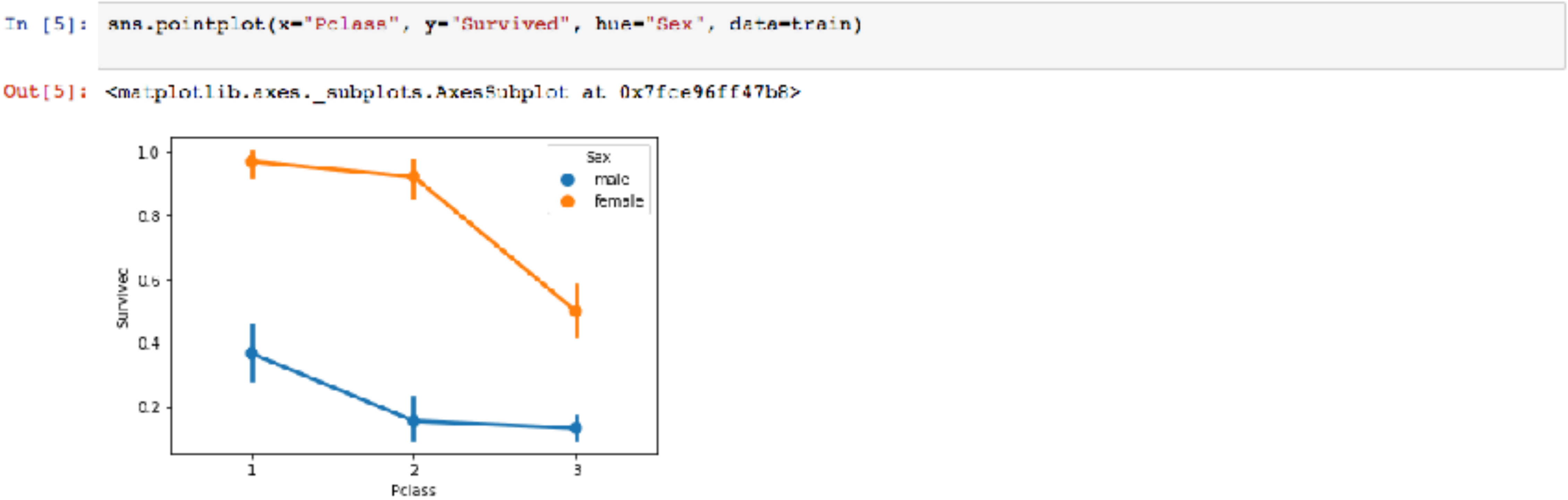


In [4]:

```
pd.crosstab([train['Sex'],train['Survived']],train['Pclass'],margins=True).style.background_gradient(cmap='summer_r')
```

Out[4]:

	Pclass	1	2	3	All
Sex	Survived				
female	0	3	6	72	81
	1	91	70	72	233
male	0	77	91	300	468
	1	45	17	47	109
All		216	184	491	891

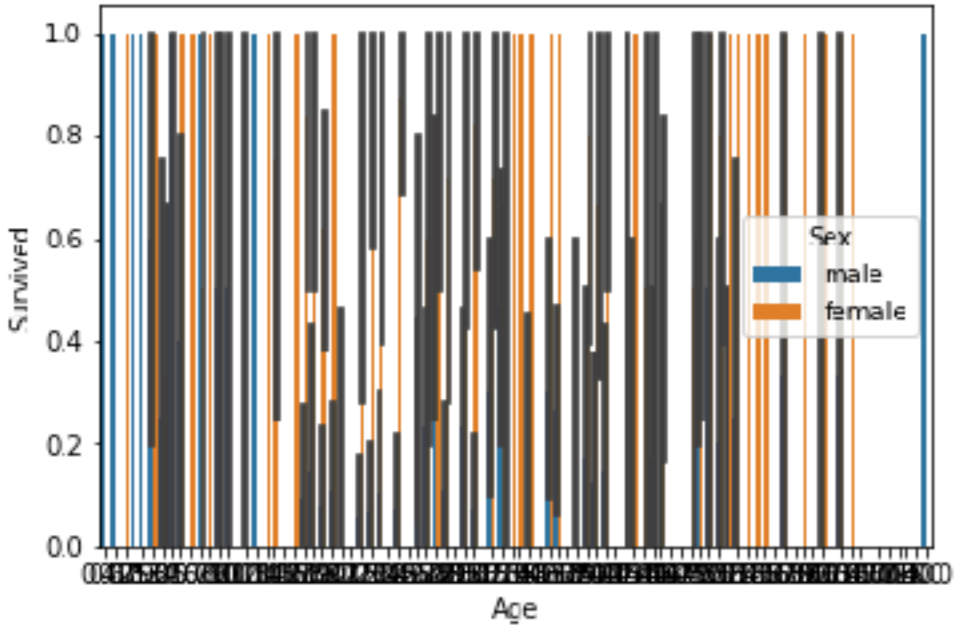


```
In [5]: sns.pointplot(x="Pclass", y="Survived", hue="Sex", data=train)
```

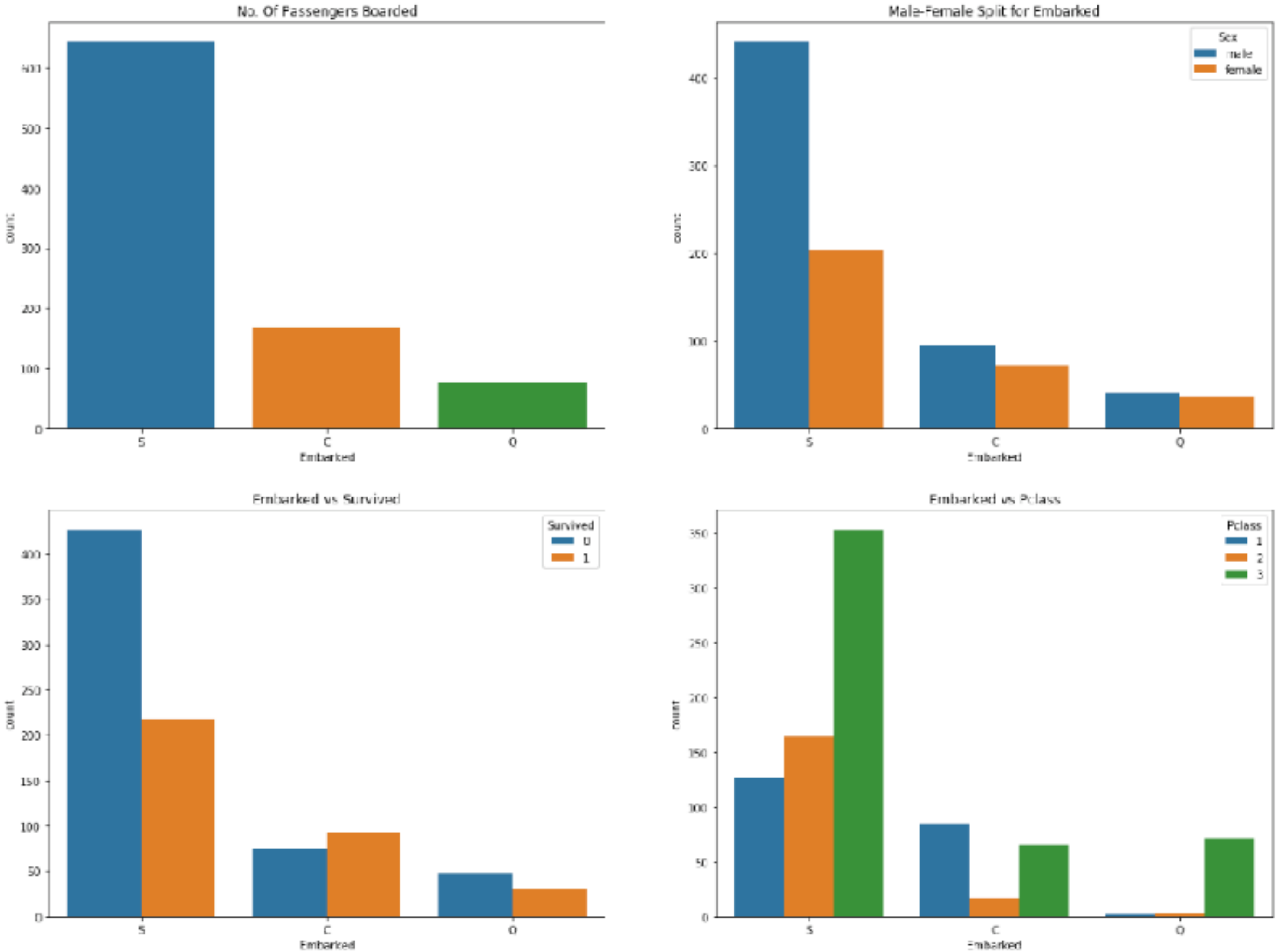
```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce96ff47b8>
```

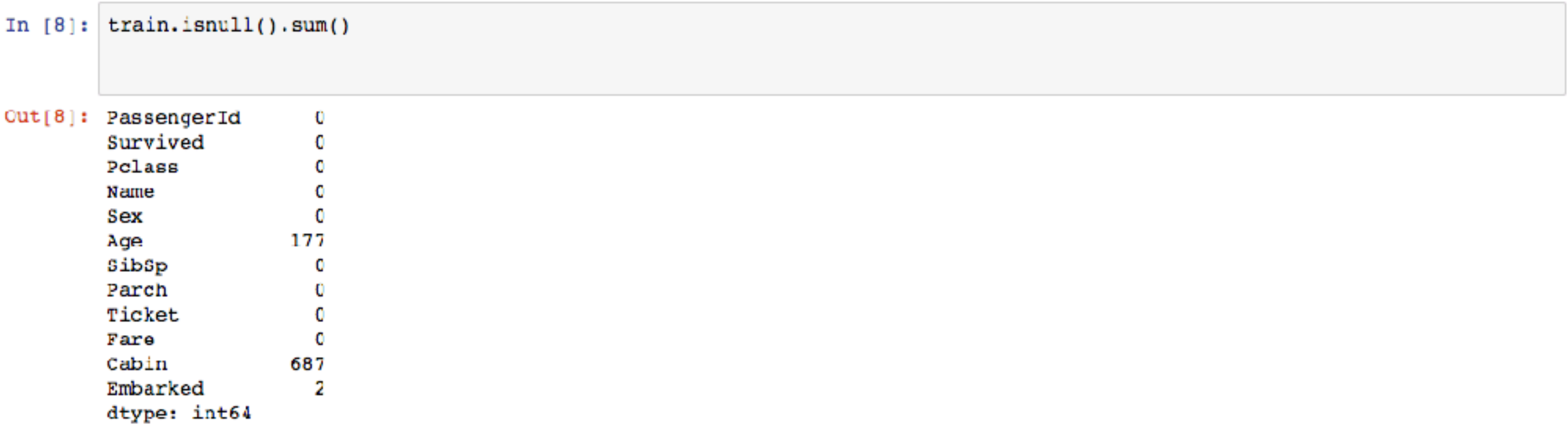
```
In [6]: sns.barplot(x="Age", y="Survived", hue="Sex", data=train)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc96f78128>
```



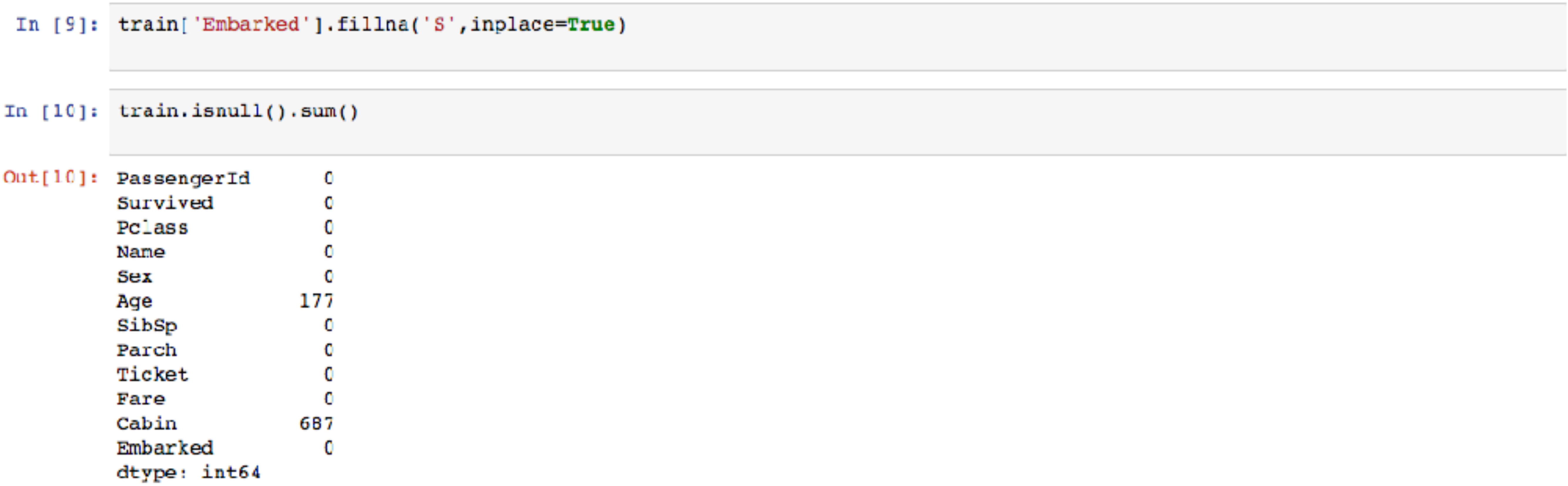
In [7]:
```
f, ax = plt.subplots(2, 2, figsize=(20,15))
sns.countplot('Embarked', data=train,ax=ax[0,0])
ax[0,0].set_title('No. Of Passengers Boarded')
sns.countplot('Embarked',hue='Sex',data=train,ax=ax[0,1])
ax[0,1].set_title('Male-Female Split for Embarked')
sns.countplot('Embarked',hue='Survived',data=train,ax=ax[1,0])
ax[1,0].set_title('Embarked vs Survived')
sns.countplot('Embarked',hue='Pclass',data=train,ax=ax[1,1])
ax[1,1].set_title('Embarked vs Pclass')
plt.show()
```





```
In [8]: train.isnull().sum()
```

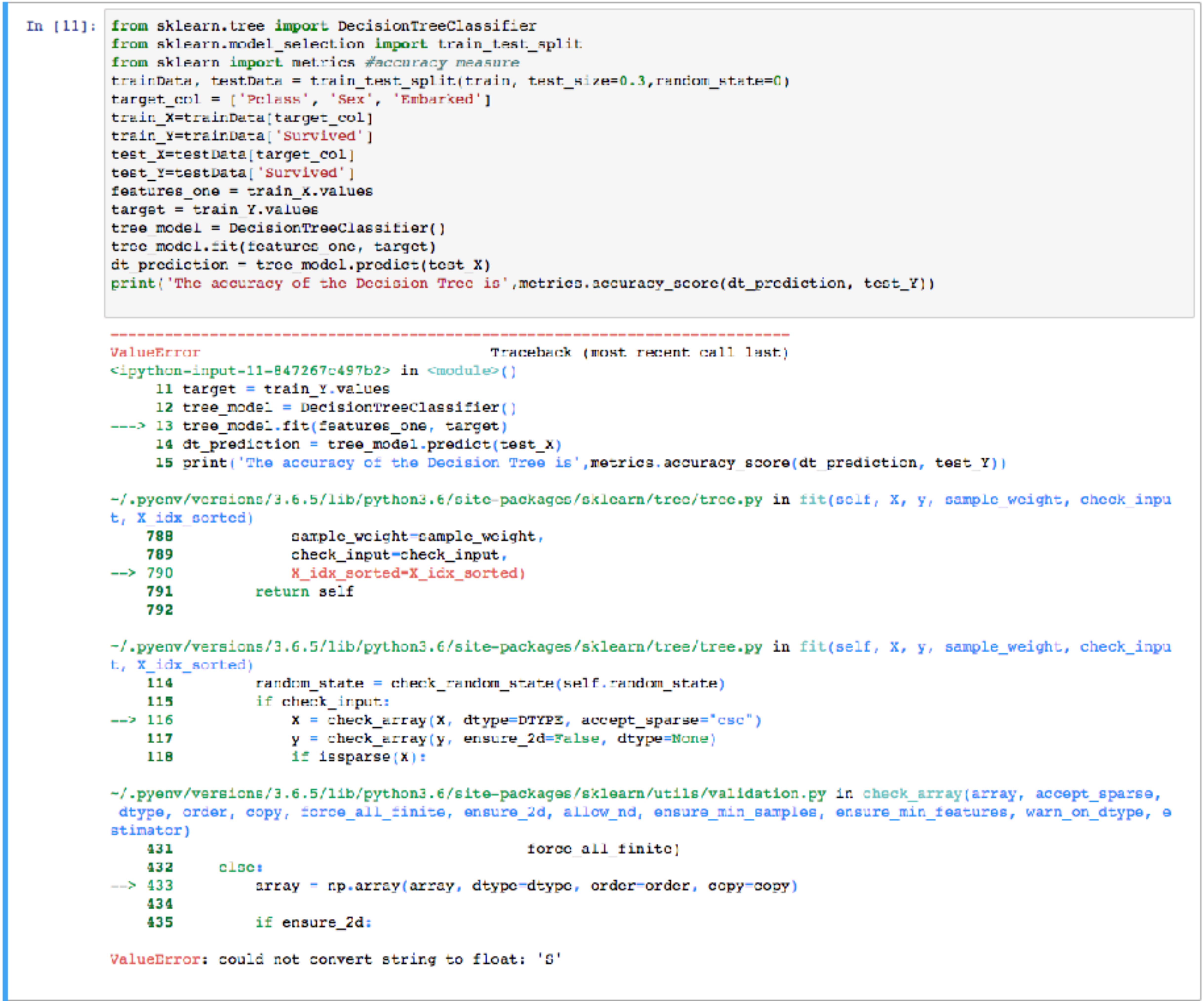
```
Out[8]: PassengerId      0
        Survived         0
        Pclass           0
        Name             0
        Sex              0
        Age            177
        SibSp            0
        Parch            0
        Ticket           0
        Fare             0
        Cabin          687
        Embarked         2
        dtype: int64
```



```
In [9]: train['Embarked'].fillna('S',inplace=True)
```

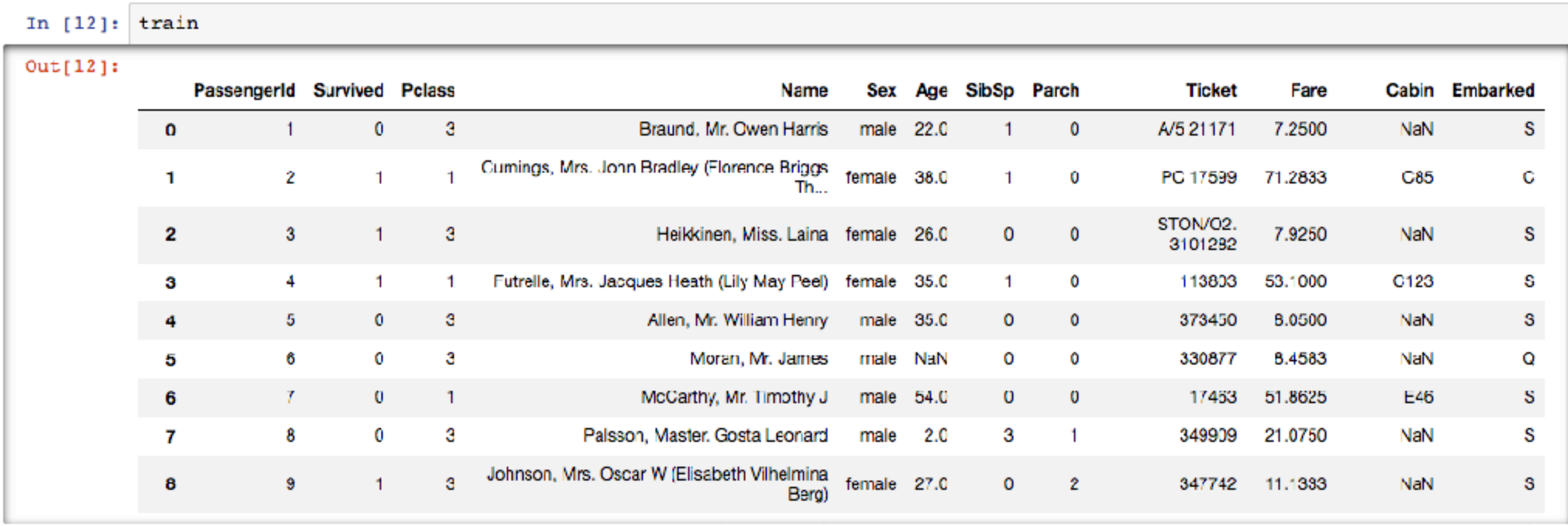
```
In [10]: train.isnull().sum()
```

```
Out[10]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         0
         dtype: int64
```



```
In [11]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn import metrics #accuracy measure
         trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
         target_col = ['Pclass', 'Sex', 'Embarked']
         train_X=trainData[target_col]
         train_Y=trainData['Survived']
         test_X=testData[target_col]
         test_Y=testData['Survived']
         features_one = train_X.values
         target = train_Y.values
         tree_model = DecisionTreeClassifier()
         tree_model.fit(features_one, target)
         dt_prediction = tree_model.predict(test_X)
         print('The accuracy of the Decision Tree is',metrics.accuracy_score(dt_prediction, test_Y))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-847267c497b2> in <module>()
     11 target = train_Y.values
     12 tree_model = DecisionTreeClassifier()
---> 13 tree_model.fit(features_one, target)
     14 dt_prediction = tree_model.predict(test_X)
     15 print('The accuracy of the Decision Tree is',metrics.accuracy_score(dt_prediction, test_Y))

~/.pyenv/versions/3.6.5/lib/python3.6/site-packages/sklearn/tree/tree.py in fit(self, X, y, sample_weight, check_inpu
t, X_idx_sorted)
    788             sample_weight=sample_weight,
    789             check_input=check_input,
--> 790             X_idx_sorted=X_idx_sorted)
    791         return self
    792 

~/.pyenv/versions/3.6.5/lib/python3.6/site-packages/sklearn/tree/tree.py in fit(self, X, y, sample_weight, check_inpu
t, X_idx_sorted)
    114         random_state = check_random_state(self.random_state)
    115         if check_input:
--> 116             X = check_array(X, dtype=DTYPE, accept_sparse="csc")
    117             y = check_array(y, ensure_2d=False, dtype=None)
    118             if issparse(X):

~/.pyenv/versions/3.6.5/lib/python3.6/site-packages/sklearn/utils/validation.py in check_array(array, accept_sparse,
 dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_on_dtype, e
stimator)
    431                                       force_all_finite)
    432     else:
--> 433         array = np.array(array, dtype=dtype, order=order, copy=copy)
    434 
    435         if ensure_2d:

ValueError: could not convert string to float: 'S'
```

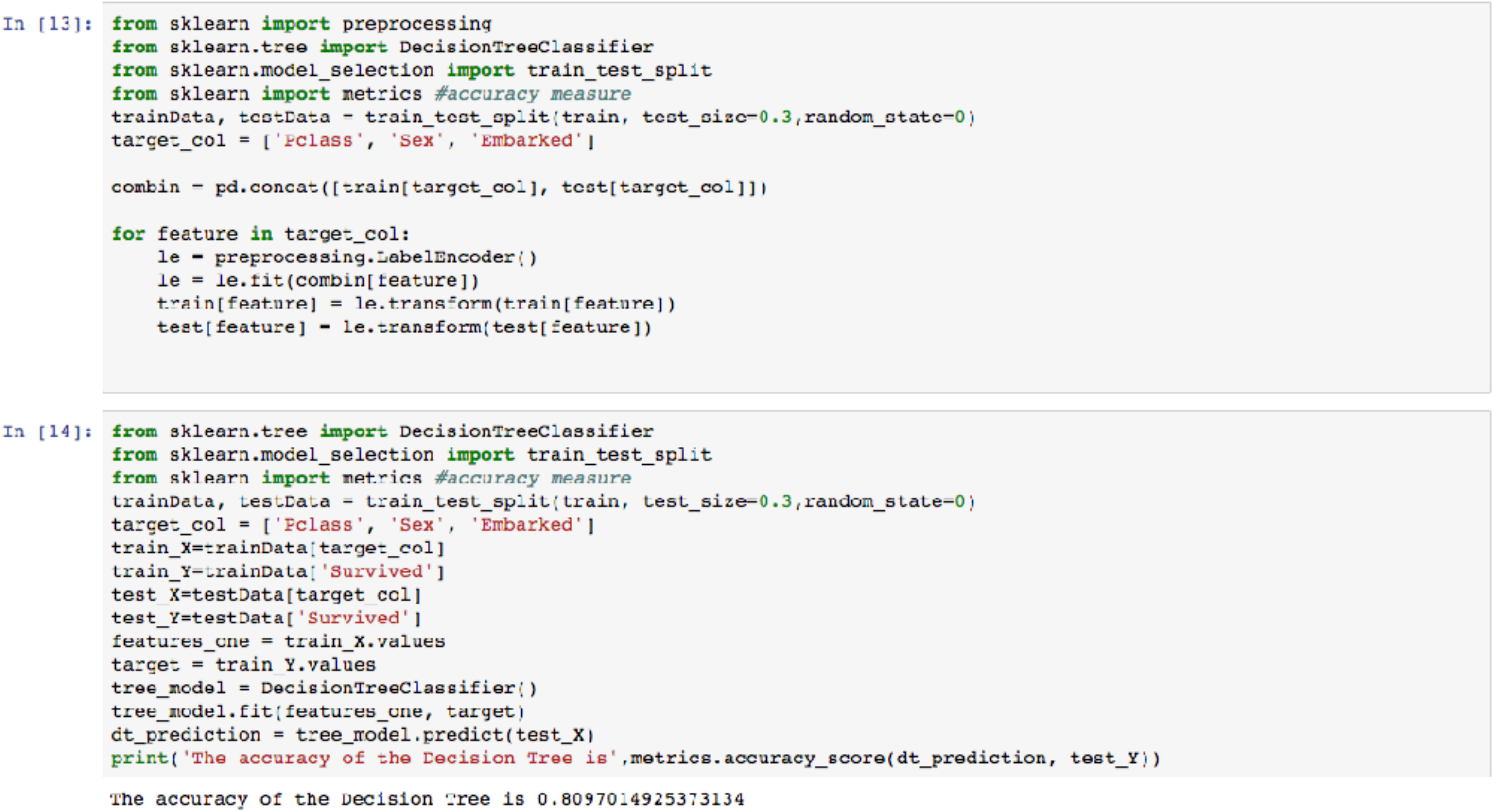


In [12]:

```
train
```

Out[12]:

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S



```
In [13]: from sklearn import preprocessing
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn import metrics #accuracy measure
         trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
         target_col = ['Pclass', 'Sex', 'Embarked']

         combin = pd.concat([train[target_col], test[target_col]])

         for feature in target_col:
             le = preprocessing.LabelEncoder()
             le = le.fit(combin[feature])
             train[feature] = le.transform(train[feature])
             test[feature] = le.transform(test[feature])
```

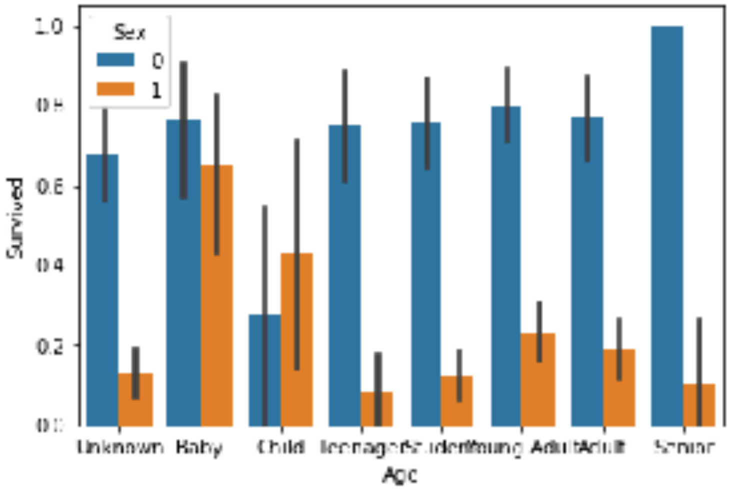
```
In [14]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn import metrics #accuracy measure
         trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
         target_col = ['Pclass', 'Sex', 'Embarked']
         train_X=trainData[target_col]
         train_Y=trainData['Survived']
         test_X=testData[target_col]
         test_Y=testData['Survived']
         features_one = train_X.values
         target = train_Y.values
         tree_model = DecisionTreeClassifier()
         tree_model.fit(features_one, target)
         dt_prediction = tree_model.predict(test_X)
         print('The accuracy of the Decision Tree is',metrics.accuracy_score(dt_prediction, test_Y))

         The accuracy of the Decision Tree is 0.8097014925373134
```

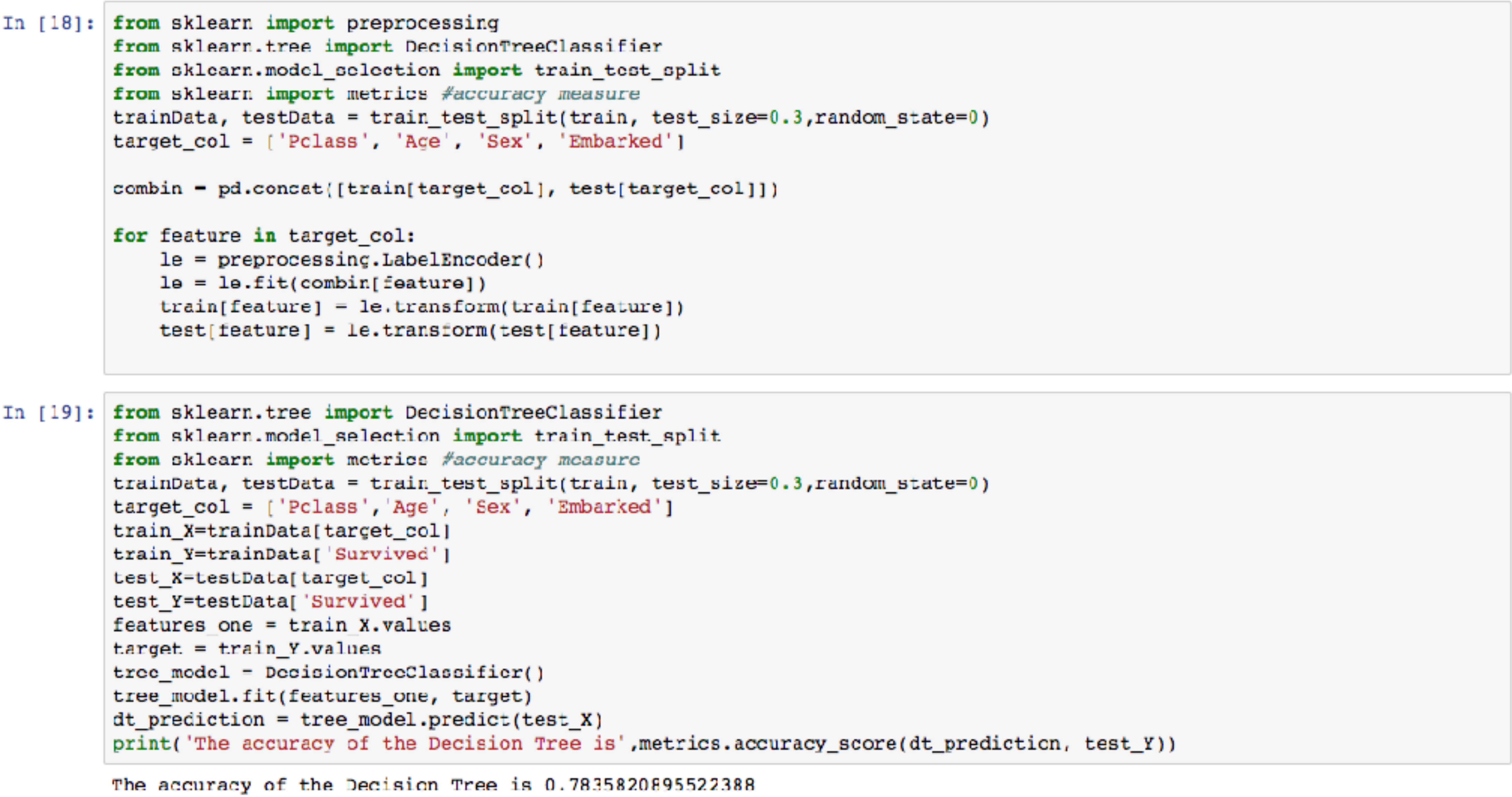
```
In [15]: test.Age = test.Age.fillna(-0.5)
         train.Age = train.Age.fillna(-0.5)
```

```
In [16]: bins = (-1, 0, 5, 12, 18, 25, 35, 60, 120)
         group_names = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Senior']
         categoriesTR = pd.cut(train.Age, bins, labels=group_names)
         categoriesTE = pd.cut(test.Age, bins, labels=group_names)
         train.Age = categoriesTR
         test.Age = categoriesTE
```

```
In [17]: sns.barplot(x="Age", y="Survived", hue="Sex", data=train)
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce96404ba8>
```





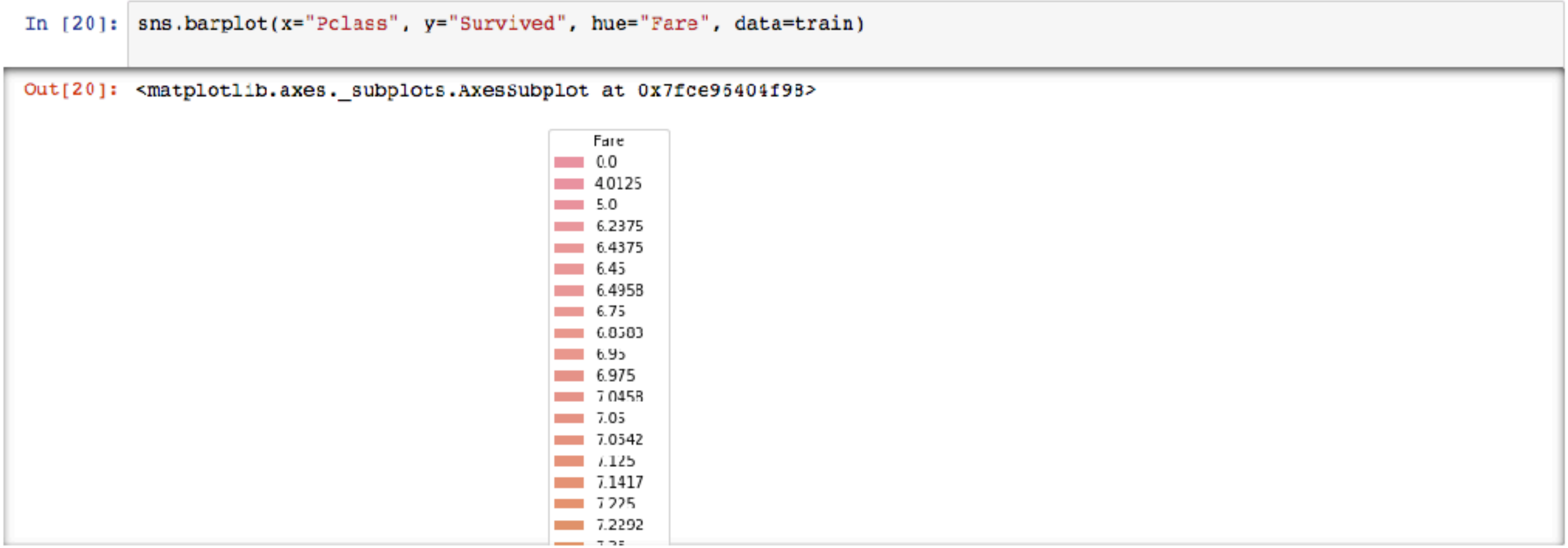
In [18]:

```
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics #accuracy measure
trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
target_col = ['Pclass', 'Age', 'Sex', 'Embarked']

combin = pd.concat([train[target_col], test[target_col]])

for feature in target_col:
    le = preprocessing.LabelEncoder()
    le = le.fit(combin[feature])
    train[feature] = le.transform(train[feature])
    test[feature] = le.transform(test[feature])
```

In [19]:

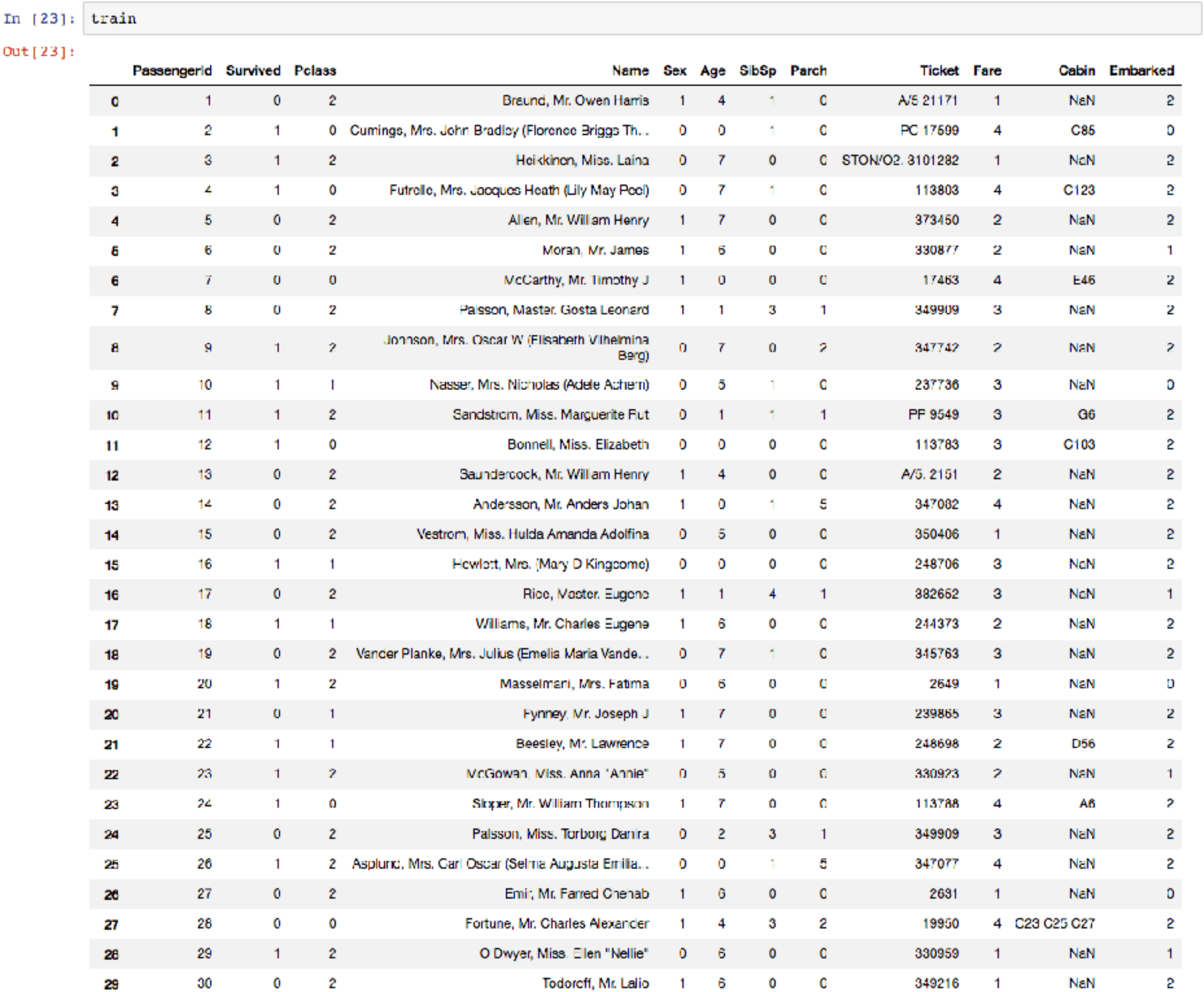
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics #accuracy measure
trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
target_col = ['Pclass','Age', 'Sex', 'Embarked']
train_X=trainData[target_col]
train_Y=trainData['Survived']
test_X=testData[target_col]
test_Y=testData['Survived']
features_one = train_X.values
target = train_Y.values
tree_model = DecisionTreeClassifier()
tree_model.fit(features_one, target)
dt_prediction = tree_model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(dt_prediction, test_Y))
```

```
The accuracy of the Decision Tree is 0.7835820895522388
```



```
In [20]: sns.barplot(x="Pclass", y="Survived", hue="Fare", data=train)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce95404f98>
```

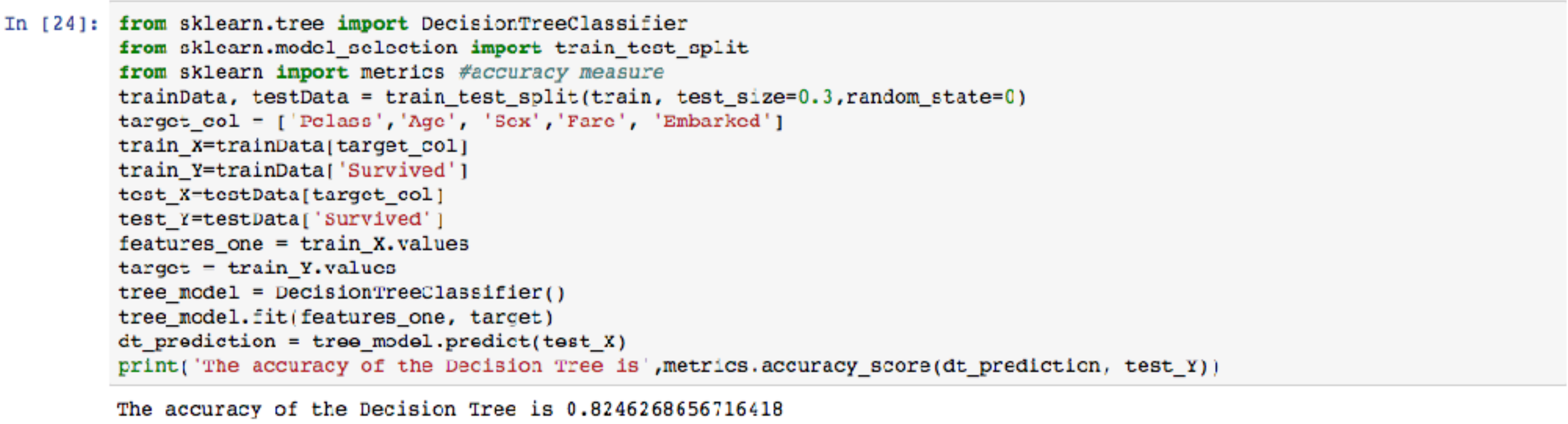
```
In [21]: train.Fare.isnull().sum()
         test.Fare.isnull().sum()

Out[21]: 1
```



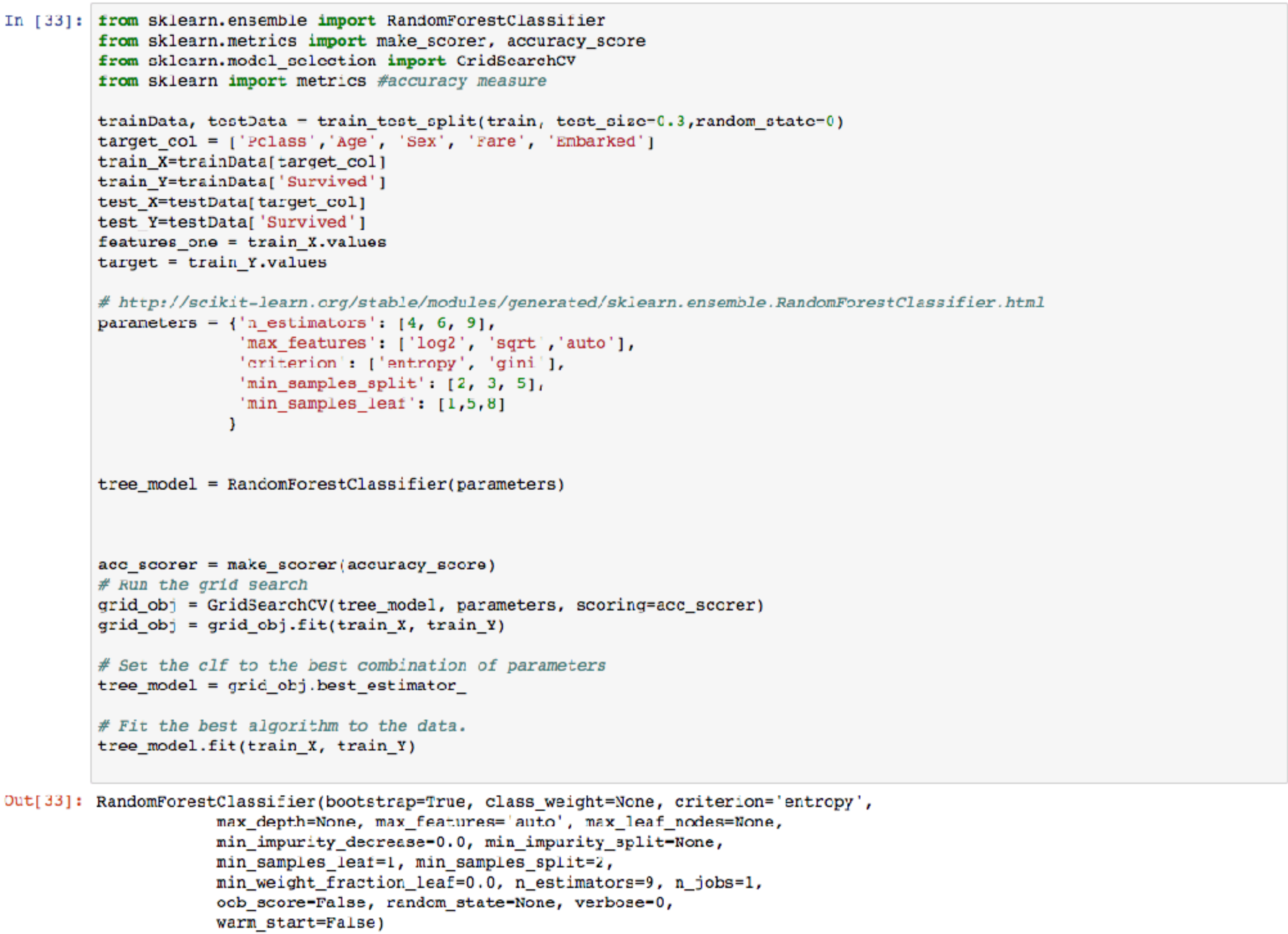
```
In [23]: train
```

```
Out[23]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	2	Braund, Mr. Owen Harris	1	4	1	0	A/5 21171	1	NaN	2
1	2	1	0	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	0	1	0	PC 17599	4	C85	0
2	3	1	2	Heikkinen, Miss. Laina	0	7	0	0	STON/O2. 3101282	1	NaN	2
3	4	1	0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	7	1	0	113803	4	C123	2
4	5	0	2	Allen, Mr. William Henry	1	7	0	0	373450	2	NaN	2
5	6	0	2	Moran, Mr. James	1	6	0	0	330877	2	NaN	1
6	7	0	0	McCarthy, Mr. Timothy J	1	0	0	0	17463	4	E46	2
7	8	0	2	Palsson, Master. Gosta Leonard	1	1	3	1	349909	3	NaN	2
8	9	1	2	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	0	7	0	2	347742	2	NaN	2
9	10	1	1	Nasser, Mrs. Nicholas (Adele Achem)	0	5	1	0	237736	3	NaN	0
10	11	1	2	Sandstrom, Miss. Marguerite Rut	0	1	1	1	PP 9549	3	G6	2
11	12	1	0	Bonnell, Miss. Elizabeth	0	0	0	0	113783	3	C103	2
12	13	0	2	Saundercock, Mr. William Henry	1	4	0	0	A/5. 2151	2	NaN	2
13	14	0	2	Andersson, Mr. Anders Johan	1	0	1	5	347082	4	NaN	2
14	15	0	2	Vestrom, Miss. Hulda Amanda Adolfina	0	5	0	0	350406	1	NaN	2
15	16	1	1	Hewlett, Mrs. (Mary D Kingcome)	0	0	0	0	248706	3	NaN	2
16	17	0	2	Rice, Master. Eugene	1	1	4	1	382652	3	NaN	1
17	18	1	1	Williams, Mr. Charles Eugene	1	6	0	0	244373	2	NaN	2
18	19	0	2	Vander Planke, Mrs. Julius (Emelia Maria Vande...	0	7	1	0	345763	3	NaN	2
19	20	1	2	Masselmani, Mrs. Fatima	0	6	0	0	2649	1	NaN	0
20	21	0	1	Fynney, Mr. Joseph J	1	7	0	0	239865	3	NaN	2
21	22	1	1	Beesley, Mr. Lawrence	1	7	0	0	248698	2	D56	2
22	23	1	2	McGowan, Miss. Anna "Annie"	0	5	0	0	330923	2	NaN	1
23	24	1	0	Sloper, Mr. William Thompson	1	7	0	0	113788	4	A6	2
24	25	0	2	Palsson, Miss. Torborg Danira	0	2	3	1	349909	3	NaN	2
25	26	1	2	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	0	0	1	5	347077	4	NaN	2
26	27	0	2	Emir, Mr. Farred Chehab	1	6	0	0	2631	1	NaN	0
27	28	0	0	Fortune, Mr. Charles Alexander	1	4	3	2	19950	4	C23 C25 C27	2
28	29	1	2	O Dwyer, Miss. Ellen "Nellie"	0	6	0	0	330959	1	NaN	1
29	30	0	2	Todoroff, Mr. Lalio	1	6	0	0	349216	1	NaN	2

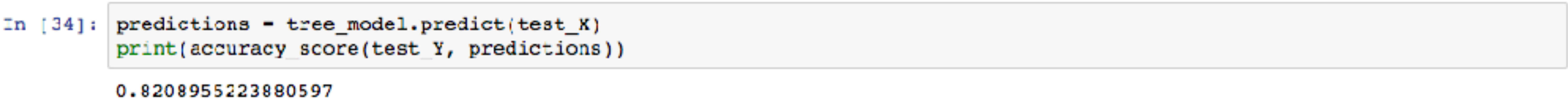


```
In [24]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn import metrics #accuracy measure
         trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
         target_col = ['Pclass','Age', 'Sex','Fare', 'Embarked']
         train_X=trainData[target_col]
         train_Y=trainData['Survived']
         test_X=testData[target_col]
         test_Y=testData['Survived']
         features_one = train_X.values
         target = train_Y.values
         tree_model = DecisionTreeClassifier()
         tree_model.fit(features_one, target)
         dt_prediction = tree_model.predict(test_X)
         print('The accuracy of the Decision Tree is',metrics.accuracy_score(dt_prediction, test_Y))

         The accuracy of the Decision Tree is 0.8246268656716418
```



In [33]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn import metrics #accuracy measure

trainData, testData = train_test_split(train, test_size=0.3,random_state=0)
target_col = ['Pclass','Age', 'Sex', 'Fare', 'Embarked']
train_X=trainData[target_col]
train_Y=trainData['Survived']
test_X=testData[target_col]
test_Y=testData['Survived']
features_one = train_X.values
target = train_Y.values

# http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
parameters = {'n_estimators': [4, 6, 9],
              'max_features': ['log2', 'sqrt','auto'],
              'criterion': ['entropy', 'gini'],
              'min_samples_split': [2, 3, 5],
              'min_samples_leaf': [1,5,8]
             }


tree_model = RandomForestClassifier(parameters)



acc_scorer = make_scorer(accuracy_score)
# Run the grid search
grid_obj = GridSearchCV(tree_model, parameters, scoring=acc_scorer)
grid_obj = grid_obj.fit(train_X, train_Y)

# Set the clf to the best combination of parameters
tree_model = grid_obj.best_estimator_

# Fit the best algorithm to the data.
tree_model.fit(train_X, train_Y)
```

Out[33]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=9, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```



In [34]:

```
predictions = tree_model.predict(test_X)
print(accuracy_score(test_Y, predictions))
```

```
0.8208955223880597
```