

به نام یکتای بی همتا

نام: امیرحسین سهراب بیگ

شماره ی دانشجویی: ۹۳۳۱۰۶۵

درس: دادکاوی

استاد: دکتر ناظر فرد

موضوع: تمرین سری اول

آرائه ی مدلی برای پیش بینی زنده ماندن یا کشته شدن مسافران کشتی تایتانیک

نام کابری در وب سایت **Kaggle.com**: Sohrabbeig

بهترین درصد در وب سایت **Kaggle.com**: 79.904

/python/1.py:

```
# Import the Pandas library
import pandas as pd
# Load the train and test datasets to create two DataFrames
train_url =
'http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv'
train = pd.read_csv(train_url)
test_url =
'http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv'
test = pd.read_csv(test_url)
# Passengers that survived vs passengers that passed away
print("Passengers that survived vs passengers that passed away")
print(train["Survived"].value_counts())
print("-----")
# As proportions
print("Passengers that survived vs passengers that passed away as proportion")
print(train["Survived"].value_counts(normalize = True))
print("-----")
# Males that survived vs males that passed away
print("Males that survived vs males that passed away")
print(train["Survived"][train["Sex"] == 'male'].value_counts())
print("-----")
# Females that survived vs Females that passed away
print("Females that survived vs Females that passed away")
print(train["Survived"][train["Sex"] == 'female'].value_counts())
print("-----")
# Normalized male survival
print("Normalized male survival")
print(train["Survived"][train["Sex"] == 'male'].value_counts(normalize = True))
print("-----")
# Normalized female survival
print("Normalized female survival")
print(train["Survived"][train["Sex"] == 'female'].value_counts(normalize = True))
print("-----")
```

در ابتدا پکیج pandas را import می‌کنیم سپس فایل های test.csv و train.csv را دانلود کرده سپس با استفاده از تابع value\_counts تعداد افرادی که زنده مانده اند را در مرحله ی اول، و در مرحله ی دوم به تفکیک جنسیت، تعداد نجات یافتگان را پیدا میکنیم.  
که نتایج زیر بدست می‌آید :

Passengers that survived vs passengers that passed away

```
0    549
```

```
1    342
```

```
Name: Survived, dtype: int64
```

-----

Passengers that survived vs passengers that passed away as proportion

```
0    0.616162
```

```
1    0.383838
```

```
Name: Survived, dtype: float64
```

-----

Males that survived vs males that passed away

```
0    468
```

```
1    109
```

```
Name: Survived, dtype: int64
```

-----

Females that survived vs Females that passed away

```
1    233
```

```
0     81
```

```
Name: Survived, dtype: int64
```

-----

Normalized male survival

```
0    0.811092
```

1 0.188908

Name: Survived, dtype: float64

-----

Normalized female survival

1 0.742038

0 0.257962

Name: Survived, dtype: float64

-----




(۱)

یک نتیجه گیری ابتدایی که از اطلاعات بالا میتوان گرفت این است که با توجه به اینکه حدوداً ۶۲ درصد افرادی که حضور داشته اند مرده اند، پس تمام داده های تست ما نیز خواهند مرد. که مشخصاً تحلیل خوبی نیست

**/python/2.py:**

```
import pandas as pd
import numpy as np
# Load the train and test datasets to create two DataFrames
train_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)
test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
my_prediction = np.zeros((test.shape[0],), dtype=np.int)
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(my_prediction, PassengerId, columns = ["Survived"])
my_solution.to_csv("my_solution_one.csv", index_label = ["PassengerId"])
```

در کد بالا پکیج numpy را نیز import کردیم زیرا در تولید آرایه‌ی صفر برای ما کاربرد داشت

4738	▼ 516	JoshinSmith		0.75598	3	7d
4739	new	AmirhosseinSohrabbeig		0.75598	4	~10s
4740	new	HassamSolano		0.75598	1	7d

به هر حال پس از آپلود کردن فایل نتیجه در سایت Kaggle به درستی ۷۵ درصد مواقع پیش بینی کرده بود





(۲)

با توجه به اطلاعاتی که مرحله‌ی قبل بدست آوردیم به نظر میرسد که خانم‌ها شانس بیشتری برای زنده ماندن نسبت به آقایان دارند. لذا پیش بینی ما برای داده‌ی تست میتواند به این صورت باشد که هر کسی که زن بود زنده می‌ماند و هرکسی که مرد بود می‌میرد

**/python/3.py:**

```
import pandas as pd
import numpy as np
test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
# Initialize a Survived column to 0
test["Survived"] = 0
# Set Survived to 1 if Sex equals "female" and print the 'Survived' column from
'test_one'
test["Survived"][test["Sex"] == "female"] = 1
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(test[["Survived"]].as_matrix(), PassengerId, columns
= ["Survived"])
my_solution.to_csv("my_solution_two.csv", index_label = ["PassengerId"])
```

نتیجه‌ی بدست آمده در سایت کگل دقتی در حدود ۷۷ درصد را نشان میدهد:

4372	new	Anat M		0.76555	1	44m
4373	▼ 148	AmirhosseinSohrabbeig		0.76555	5	now
<b>Your Best Entry ↑</b>						
Your submission scored 0.76555, which is an improvement of your previous score of 0.75598. Great job!				 <b>Tweet this!</b>		
4374	▼ 502	Disha lindal		0.76077	2	2mo


(۳)

با توجه به اینکه احتمالا بچه‌ها زودتر نجات داده میشوند یک حدس این میتواند باشد که بچه‌ها یعنی افرادی که سن بالاتر از ۱۸ دارند نجات پیدا میکنند و افرادی که سنی بالاتر از ۱۸ دارند خواهند مرد. لذا می‌توان فیلد نجات یافتن را بر اساس سن پرکرد که کد آن به صورت زیر می‌باشد

/python/4.py:

```
import pandas as pd
import numpy as np
test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
# Initialize a Survived column to 0
test["Survived"] = 0
# Set Survived to 1 if age is less than 18 and to 0 if age is greater equal than
18
test["Survived"][test["Age"] < 18] = 1
test["Survived"][test["Age"] >= 18] = 0
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(test["Survived"].as_matrix(), PassengerId, columns
= ["Survived"])
my_solution.to_csv("my_solution_three.csv", index_label = ["PassengerId"])
```

نتیجه ی آپلود فایل حاصل از کد بالا به صورت زیر می باشد:

4372	▼ 148	AmirhosseinSohrabbeig		0.76555	6	now
<b>Your Best Entry</b> ↑						
Your submission scored 0.63158, which is not an improvement of your best score. Keep trying!						

(4

درخت تصمیم:

/python/5.py:

```
# Import the Pandas library
import pandas as pd
# Import the Numpy library
import numpy as np
# Import 'tree' from scikit-learn library
from sklearn import tree
# Load the train and test datasets to create two DataFrames
train_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)
```

```

test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
# Convert the male and female groups to integer form
train["Sex"][train["Sex"] == "male"] = 0
train["Sex"][train["Sex"] == "female"] = 1
test["Sex"][test["Sex"] == "male"] = 0
test["Sex"][test["Sex"] == "female"] = 1
# Impute the Embarked variable
train["Embarked"] = train["Embarked"].fillna("S")
# Impute the Age variable
train["Age"] = train["Age"].fillna(train["Age"].median())
test["Age"] = test["Age"].fillna(test["Age"].median())
# Convert the Embarked classes to integer form
train["Embarked"][train["Embarked"] == "S"] = 0
train["Embarked"][train["Embarked"] == "C"] = 1
train["Embarked"][train["Embarked"] == "Q"] = 2
# Create the target and features numpy arrays: target, features_one
target = train["Survived"].values
features_one = train[["Pclass", "Sex", "Age", "Fare"]].values
# to understand if there is Nan in the arrays
# print(np.isnan(target).any())
# print(pd.isnull(features_one).any())
# Fit your first decision tree: my_tree_one
my_tree_one = tree.DecisionTreeClassifier()
my_tree_one = my_tree_one.fit(features_one, target)
# Impute the missing value with the median
test.Fare[152] = test["Fare"].median()
# Extract the features from the test set: Pclass, Sex, Age, and Fare.
test_features = test[["Pclass", "Sex", "Age", "Fare"]].values
# Make your prediction using the test set
my_prediction = my_tree_one.predict(test_features)
# Create a data frame with two columns: PassengerId & Survived. Survived
contains your predictions
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(my_prediction, PassengerId, columns = ["Survived"])
my_solution.to_csv("my_solution_four.csv", index_label = ['PassengerId'])

```


در مراحل قبل ما به صورت دستی در مورد زنده ماندن یا نماندن هر یک از مسافران کشتی تایتانیک تصمیم می‌گرفتیم. یک راه ساده تر استفاده از درخت تصمیم است. در درخت تصمیم همه ی داده ها در ابتدا در ریشه قرار دارند. سپس مشخص میشود که در هر مرحله بر اساس چه ویژگی ای داده ها را تقسیم کنیم. پس از هر تقسیم یک لایه پایین میرویم و همین طور ادامه داده تا به گره های نهایی برسیم. در گره ی نهایی مشخص میشود که داده هایی که در این گره قرار میگیرند زنده می‌مانند یا خیر

برای ساختن درخت کلاس `tree` را از پکیج `sklearn` ، `import` میکنیم.

دو نکته وجود دارد. اول اینکه درخت تصمیم ما با مقادیر غیر عددی کار نمیکند لذا باید این به هر یک از این مقادیر یک عدد نسبت بدهیم. مثلا به جای مقدار `s` برای ویژگی `embarked` عدد یک و ... استفاده میکنیم از طرف دیگر برخی از مسافران دارای مقادیر نامشخص یا `NaN` برای برخی ویژگی های خود هستند که باید به جای این مقادیر نامشخص مقدار بگذاریم. که به این عمل `imputation` گفته میشود

```
my_tree_one = tree.DecisionTreeClassifier()
my_tree_one = my_tree_one.fit(features_one,target)
```

در دو خط بالا هم درخت تصمیم را بر اساس آرایه ی `features_one` و آرایه ی `target` میسازیم در انتها با استفاده از درخت ساخته شده زنده ماندن یا نماندن مسافران موجود در دیتافریم `test` را پیش بینی می‌کنیم. دقت این پیش بینی در وب سایت کگل ۷۲ درصد ارزیابی شده است.

4320	▼ 105	AmirhosseinSohrabbeig		0.76555	8	now
<b>Your Best Entry</b> ⬆						
Your submission scored 0.72249, which is not an improvement of your best score. Keep trying!						

(5)

`/python/6.py:`

```
# Import the Pandas library
import pandas as pd
# Import the Numpy library
import numpy as np
# Import 'tree' from scikit-learn library
from sklearn import tree
# Load the train and test datasets to create two DataFrames
train_url =
'http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv'
train = pd.read_csv(train_url)
```




```

test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
# Convert the male and female groups to integer form
train["Sex"][train["Sex"] == "male"] = 0
train["Sex"][train["Sex"] == "female"] = 1
test["Sex"][test["Sex"] == "male"] = 0
test["Sex"][test["Sex"] == "female"] = 1
# Impute the Embarked variable
train["Embarked"] = train["Embarked"].fillna("S")
# Impute the Age variable
train["Age"] = train["Age"].fillna(train["Age"].median())
test["Age"] = test["Age"].fillna(test["Age"].median())
# Convert the Embarked classes to integer form
train["Embarked"][train["Embarked"] == "S"] = 0
train["Embarked"][train["Embarked"] == "C"] = 1
train["Embarked"][train["Embarked"] == "Q"] = 2
test["Embarked"][test["Embarked"] == "S"] = 0
test["Embarked"][test["Embarked"] == "C"] = 1
test["Embarked"][test["Embarked"] == "Q"] = 2
# Create the target and features numpy arrays: target, features_two
target = train["Survived"].values
# Create a new array with the added features: features_two
features_two = train[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch",
"Embarked"]].values
# Control overfitting by setting "max_depth" to 10 and "min_samples_split" to 5 :
my_tree_two
max_depth = 10
min_samples_split = 5
my_tree_two = tree.DecisionTreeClassifier(max_depth = max_depth,
min_samples_split = min_samples_split, random_state = 1)
my_tree_two = my_tree_two.fit(features_two, target)
# Impute the missing value with the median
test.Fare[152] = test["Fare"].median()
# Extract the features from the test set: Pclass, Age, Sex, Fare, SibSp, Parch,
Embarked
test_features = test[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch",
"Embarked"]].values
# Make your prediction using the test set
my_prediction = my_tree_two.predict(test_features)

```

```
# Create a data frame with two columns: PassengerId & Survived. Survived
contains your predictions
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(my_prediction, PassengerId, columns = ["Survived"])
my_solution.to_csv("my_solution_five.csv", index_label = ['PassengerId'])
```

نتیجه ی بدست آمده از مدل قبل نشان میدهد که انگار مدل ما به خوبی عمل نمیکند زیرا حتی از مدل ساده ای که بر اساس تفکیک جنسیت ساختیم نتیجه ی بدتری گرفت. به نظر میرسید که مدل ما بیشتر از پیش بینی حفظ میکند یا به عبارت دیگر دچار **overfitting** شده است برای جلوگیری از این مسئله از دو آرگومان **max\_depth** و **min\_sample\_split** که در حالت قبل به صورت پیش فرض مقدار **none** داشتند استفاده می کنیم. پارامتر **max\_depth** حداکثر عمق درخت را مشخص میکند و اجازه نمیدهد که درخت تصمیم ساخته شده از یک حدی که ما به عنوان ورودی به تابع میدهم، بزرگ تر شود. پارامتر **min\_sample\_split** به نحو دیگری مانع از **overfitting** میشود به این صورت که به یک گره اجازه ی تقسیم شدن نمیدهد در حالیکه تعداد نمونه های موجود در آن گره از یک مقدار مشخصی کمتر باشد به این صورت دیگر درخت ما به داده ها حساس نخواهد بود و **overfitting** رخ نخواهد داد آپلود فایل حاصل از کد بالا نتیجه ی زیر را داشت که دقت بالاتری نسبت به حالت قبل دارد

4320	▼ 104	AmirhosseinSohrabbeig		0.76555	9	now
<b>Your Best Entry ↑</b> Your submission scored 0.76077, which is not an improvement of your best score. Keep trying!						

(6)

**/python/7.py:**

```
# Import the Pandas library
import pandas as pd
# Import the Numpy library
import numpy as np
# Import 'tree' from scikit-learn library
from sklearn import tree
# Load the train and test datasets to create two DataFrames
train_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)
test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
```

```


test = pd.read_csv(test_url)
# Convert the male and female groups to integer form
train["Sex"][train["Sex"] == "male"] = 0
train["Sex"][train["Sex"] == "female"] = 1
test["Sex"][test["Sex"] == "male"] = 0
test["Sex"][test["Sex"] == "female"] = 1
# Impute the Embarked variable
train["Embarked"] = train["Embarked"].fillna("S")
# Impute the Age variable
train["Age"] = train["Age"].fillna(train["Age"].median())
test["Age"] = test["Age"].fillna(test["Age"].median())
# Convert the Embarked classes to integer form
train["Embarked"][train["Embarked"] == "S"] = 0
train["Embarked"][train["Embarked"] == "C"] = 1
train["Embarked"][train["Embarked"] == "Q"] = 2
test["Embarked"][test["Embarked"] == "S"] = 0
test["Embarked"][test["Embarked"] == "C"] = 1
test["Embarked"][test["Embarked"] == "Q"] = 2
# Create the target and features numpy arrays: target, features_two
target = train["Survived"].values
# Create train_two with the newly defined feature
train["family_size"] = train["SibSp"] + train["Parch"] + 1
test["family_size"] = test["SibSp"] + test["Parch"] + 1
# Create a new feature set and add the new feature
features_three = train[["Pclass", "Sex", "Age", "Fare", "SibSp", "Parch",
"family_size"]].values
# Define the tree classifier, then fit the model
my_tree_three = tree.DecisionTreeClassifier()
my_tree_three = my_tree_three.fit(features_three, target)
# Impute the missing value with the median
test.Fare[152] = test["Fare"].median()
# Extract the features from the test set: Pclass, Age, Sex, Fare, SibSp, Parch,
Embarked
test_features = test[["Pclass", "Sex", "Age", "Fare", "SibSp", "Parch",
"family_size"]].values
# Make your prediction using the test set
my_prediction = my_tree_three.predict(test_features)
# Create a data frame with two columns: PassengerId & Survived. Survived
contains your predictions
PassengerId = np.array(test["PassengerId"]).astype(int)

```

```
my_solution = pd.DataFrame(my_prediction, PassengerId, columns = ["Survived"])
my_solution.to_csv("my_solution_six.csv", index_label = ['PassengerId'])
```

به نظر میرسد که تعداد افراد یک خانواده در نجات یافتن اعضای آن تاثیر داشته باشد. زیرا هرچه خانواده بزرگتر باشد امکان جمع شدن افراد در کنار یکدیگر و تلاش برای نجات سخت تر خواهد بود. به همین دلیل این ویژگی اندازه ی خانواده میتواند یک عامل مهم در درخت تصمیم ما باشد. لذا این ویژگی که از قبل وجود نداشت را به دیتافریم اضافه میکنیم. و درخت تصمیم را میسازیم.

نتیجه ی کار زنده ماندن افراد را با دقت ۶۲ درصد پیش بینی میکند

4318	▼ 104	AmirhosseinSohrabbeig		0.76555	10	now
------	-------	-----------------------	---	---------	----	-----

**Your Best Entry ↑**

Your submission scored 0.62201, which is not an improvement of your best score. Keep trying!

(7)

جنگل تصادفی:

`/python/8.py:`

```
# Import the Pandas library
import pandas as pd
# Import the Numpy library
import numpy as np
# Import the 'RandomForestClassifier'
from sklearn.ensemble import RandomForestClassifier
# Load the train and test datasets to create two DataFrames
train_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)
test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)
# Convert the male and female groups to integer form
train["Sex"][train["Sex"] == "male"] = 0
train["Sex"][train["Sex"] == "female"] = 1
test["Sex"][test["Sex"] == "male"] = 0
test["Sex"][test["Sex"] == "female"] = 1
# Impute the Embarked variable
```

```

train["Embarked"] = train["Embarked"].fillna("S")
# Impute the Age variable
train["Age"] = train["Age"].fillna(train["Age"].median())
test["Age"] = test["Age"].fillna(test["Age"].median())
# Convert the Embarked classes to integer form
train["Embarked"][train["Embarked"] == "S"] = 0
train["Embarked"][train["Embarked"] == "C"] = 1
train["Embarked"][train["Embarked"] == "Q"] = 2
test["Embarked"][test["Embarked"] == "S"] = 0
test["Embarked"][test["Embarked"] == "C"] = 1
test["Embarked"][test["Embarked"] == "Q"] = 2
# Create the target and features numpy arrays: target, features_two
target = train["Survived"].values
# Impute the missing value with the median
test.Fare[152] = test["Fare"].median()
# We want the Pclass, Age, Sex, Fare, SibSp, Parch, and Embarked variables
features_forest = train[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch",
"Embarked"]].values
# Building and fitting my_forest
forest = RandomForestClassifier(max_depth = 10, min_samples_split=2, n_estimators
= 100, random_state = 1)
my_forest = forest.fit(features_forest, target)
# Compute predictions on our test set features then print the length of the
prediction vector
test_features = test[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch",
"Embarked"]].values
pred_forest = my_forest.predict(test_features)
# Create a data frame with two columns: PassengerId & Survived. Survived
contains your predictions
PassengerId = np.array(test["PassengerId"]).astype(int)
my_solution = pd.DataFrame(pred_forest, PassengerId, columns = ["Survived"])
my_solution.to_csv("my_solution_seven.csv", index_label = ["PassengerId"])

```

برای رفع مشکل **overfitting** که در استفاده از مدل درخت تصمیم رخ میداد میتوان از مدل دیگری به اسم جنگل تصادفی استفاده کرد. به این صورت که برای **feature** های مشخص روی داده های تمرین درخت های تصمیم متفاوت و بسیار عمیقی ساخته میشود. و با هریک زنده ماندن یا مردن سرنشینان پیش بینی میشود. نتیجهی هر درخت به عنوان یک رای در نظر گرفته میشود و در نهایت مقداری که برای زنده ماندن فرد انتخاب میکنیم مقداری است که رای بیشتری بیاورد.

```
forest = RandomForestClassifier(max_depth = 10, min_samples_split=2, n_estimators  
= 100, random_state = 1)
```

تکه کد بالا این جنگل تصادفی را برای ما میسازد. پارامتر آخر تعداد درخت های این جنگل را مشخص میکند.

4317 ▼ 103 AmirhosseinSohrabbeig



0.76555

11

now

**Your Best Entry** ↑

Your submission scored 0.75120, which is not an improvement of your best score. Keep trying!

:R

این قسمت دقیقاً از لحاظ مفهومی مانند قسمت بالا است که با زبان R مدل‌ها پیاده‌سازی شده‌اند  
در هر قسمت آدرسی که فایل در آن قرار داده گرفته است مشخص شده است

*/other/1.R:*

**# Import the training set: train**

**train\_url <-**

**"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"**

**train <- read.csv(train\_url)**

**# Import the testing set: test**

**test\_url <-**

**"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"**

**test <- read.csv(test\_url)**

**# Print train and test to the console**

**str(test)**

**str(train)**

**# Survival rates in absolute numbers**

**print(table(train\$Survived))**

**# Survival rates in proportions**

**print(prop.table(table(train\$Survived)))**

**# Two-way comparison: Sex and Survived**

**print(table(train\$Sex, train\$Survived))**

**# Two-way comparison: row-wise proportions**

**print(prop.table(table(train\$Sex, train\$Survived), margin = 1))**

```
0    1
549 342
```

```
      0      1
0.6161616 0.3838384
```

```
      0    1
female 81 233
male   468 109
```

```
      0      1
female 0.2579618 0.7420382
male   0.8110919 0.1889081
```

(1

*/other/2.R:*

**# Import the training set: train**

**train\_url <-**

**"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"**

**train <- read.csv(train\_url)**

**# Import the testing set: test**

**test\_url <-**

**"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"**

**test <- read.csv(test\_url)**


**PassengerId <- as.vector(test\$PassengerId)**

**test\_one <- data.frame(PassengerId)**

**test\_one\$Survived <- 0**

**write.csv(test\_one, file = "first\_prediction.csv", row.names=FALSE)**



4318	▼ 106	AmirhosseinSohrabbeig		0.76555	12	now
------	-------	-----------------------	---	---------	----	-----

**Your Best Entry** ↑


Your submission scored 0.62679, which is not an improvement of your best score. Keep trying!

(2)

*/other/3.R:*

```
# Import the testing set: test
test_url <-
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test <- read.csv(test_url)

PassengerId <- as.vector(test$PassengerId)
test_one <- data.frame(PassengerId)
test_one$Survived <- 0
test_one$Survived[test$Sex == "female"] <- 1
write.csv(test_one, file = "second_prediction.csv", row.names=FALSE)
```

4318	▼ 105	AmirhosseinSohrabbeig		0.76555	13	now
------	-------	-----------------------	---	---------	----	-----

**Your Best Entry** ↑

Your submission scored 0.76555, which is not an improvement of your best score. Keep trying!

(3)

*/other/4.R:*

```
# Import the training set: train
```

```

train_url <-
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train <- read.csv(train_url)


# Create the column child, and indicate whether child or no child
train$Child <- NA
train$Child[train$Age < 18] <- 1
train$Child[train$Age >= 18] <- 0

# Two-way comparison
prop.table(table(train$Child,train$Survived),1)

# Import the testing set: test
test_url <-
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test <- read.csv(test_url)

PassengerId <- as.vector(test$PassengerId)
test_one <- data.frame(PassengerId)
test_one$Survived <- 0
test_one$Survived[test$Age < 18] <- 1
write.csv(test_one, file = "third_prediction.csv",row.names=FALSE)

```

4319	▼ 106	AmirhosseinSohrabbeig		0.76555	14	now
<p><b>Your Best Entry</b> ↑</p> <p>Your submission scored 0.63158, which is not an improvement of your best score. Keep trying!</p>						

**# Load in the R package**

**library(rpart)**

**# Build the decision tree**

**my\_tree\_two <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +  
Embarked, data = train, method = "class")**

**# Visualize the decision tree using plot() and text()**

**plot(my\_tree\_two)**

**text(my\_tree\_two)**

**# Load in the packages to build a fancy plot**

**library(rattle)**

**library(rpart.plot)**

**library(RColorBrewer)**

**# Time to plot your fancy tree**

**fancyRpartPlot(my\_tree\_two)**

**# Make predictions on the test set**

**my\_prediction <- predict(my\_tree\_two, test, type = "class")**

**# Finish the data.frame() call**

**my\_solution <- data.frame(PassengerId = test\$PassengerId, Survived =  
my\_prediction)**

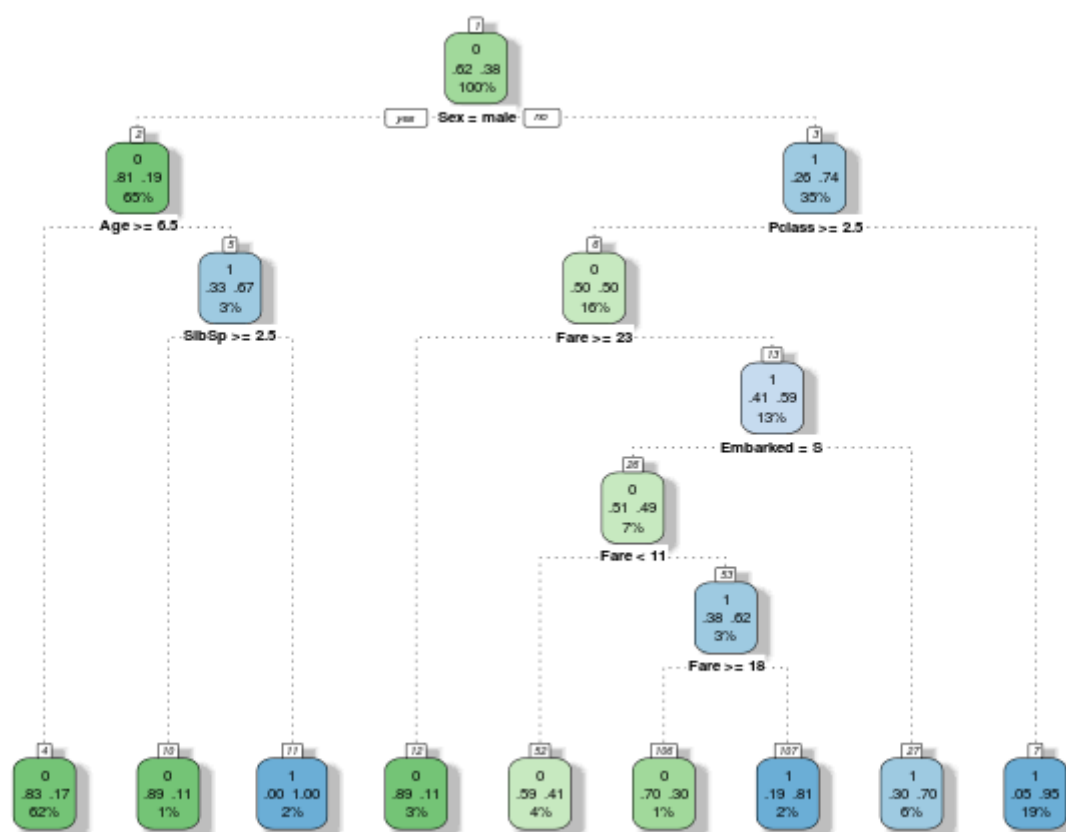
**# Use nrow() on my\_solution**

**nrow(my\_solution)**

**# Finish the write.csv() call**

```
write.csv(my_solution, file = "fourth_prediction.csv", row.names = FALSE)
```

/other/plots/Rplot01.png:



Rattle 2017-Feb-23 21:49:36 rstudio

2347 ▲ 18... AmirhosseinSohrabbeig



0.78469

15

now

Your Best Entry ▲

You advanced 1,869 places on the leaderboard!

Your submission scored 0.78469, which is an improvement of your previous score of 0.76555. Great job!



Tweet this!

*/other/6.R:*

**# Load in the R package**

**library(rpart)**

**my\_tree\_three <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +  
Embarked,**

**data = train, method = "class", control = rpart.control(minsplit =  
50, cp = 0))**

**# Visualize the decision tree using plot() and text()**

**plot(my\_tree\_two)**

**text(my\_tree\_two)**

**# Load in the packages to build a fancy plot**

**library(rattle)**

**library(rpart.plot)**

**library(RColorBrewer)**

**# Visualize my\_tree\_three**

**fancyRpartPlot(my\_tree\_three)**

**# Make predictions on the test set**

**my\_prediction <- predict(my\_tree\_three, test, type = "class")**

**# Finish the data.frame() call**

**my\_solution <- data.frame(PassengerId = test\$PassengerId, Survived =  
my\_prediction)**

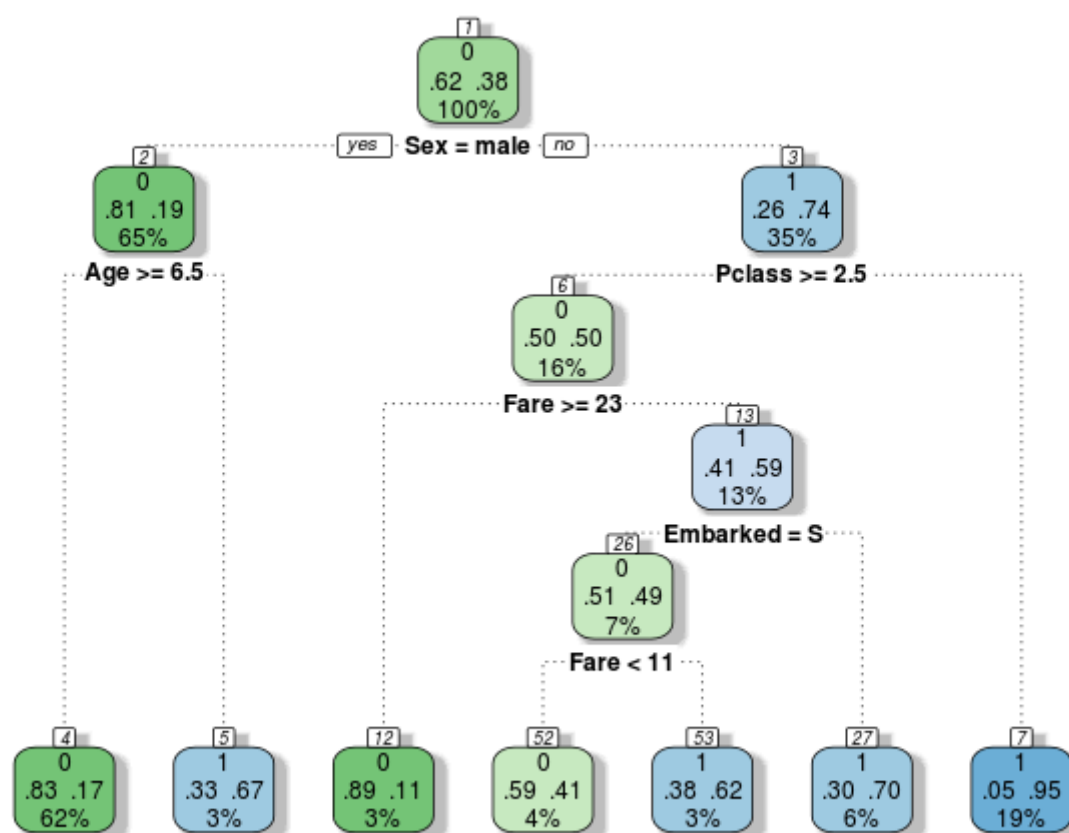
```
# Use nrow() on my_solution
```

```
nrow(my_solution)
```

```
# Finish the write.csv() call
```

```
write.csv(my_solution, file = "5th_prediction.csv", row.names = FALSE)
```

/other/plots/Rplot01.png:



Rattle 2017-Feb-23 22:07:58 rstudio

2347 ▲ 18... AmirhosseinSohrabbeig



0.78469

16

now

Your Best Entry ↑

Your submission scored 0.77990, which is not an improvement of your best score. Keep trying!

*/other/7.R:*

```
# Create train_two
train_two <- train
train_two$family_size <- train_two$SibSp + train_two$Parch + 1

test_two <- test
test_two$family_size <- test_two$SibSp + test_two$Parch + 1

# Finish the command
my_tree_four <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +
Embarked + family_size,
                      data = train_two, method = "class")

# Visualize your new decision tree
fancyRpartPlot(my_tree_four)

# Make predictions on the test set
my_prediction <- predict(my_tree_four, test_two, type = "class")

# Finish the data.frame() call
my_solution <- data.frame(PassengerId = test$PassengerId, Survived =
my_prediction)

# Use nrow() on my_solution
nrow(my_solution)

# Finish the write.csv() call
write.csv(my_solution, file = "6th_prediction.csv", row.names = FALSE)
```

*/other/plots/Rplot01.png:*



Rattle 2017-Feb-23 22:23:28 rstudio

2346
▲18...
AmirhosseinSohrabbeig
0.78469
17
now

Your Best Entry ↑  
Your submission scored 0.78469, which is not an improvement of your best score. Keep trying!

(7

/other/8.R:

```
load(file = "~/all_data.RData")
```

```
# Load in the package
```

```
library(randomForest)
```

```
# Passenger on row 62 and 830 do not have a value for embarkment.
```

```
# Since many passengers embarked at Southampton, we give them the value S.
```

```
all_data$Embarked[c(62, 830)] <- "S"
```

```
# Factorize embarkment codes.
```

```
all_data$Embarked <- factor(all_data$Embarked)
```

```
# Passenger on row 1044 has an NA Fare value. Let's replace it with the median fare value.
```



```
all_data$Fare[1044] <- median(all_data$Fare, na.rm = TRUE)
```

```
# How to fill in missing Age values?
```

```
# We make a prediction of a passengers Age using the other variables and a  
decision tree model.
```

```
# This time you give method = "anova" since you are predicting a continuous  
variable.
```

```
library(rpart)
```

```
predicted_age <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked  
+ Title + family_size,
```

```
data = all_data[!is.na(all_data$Age),], method = "anova")
```

```
all_data$Age[is.na(all_data$Age)] <- predict(predicted_age,
```

```
all_data[is.na(all_data$Age),])
```

```
# Split the data back into a train set and a test set
```

```
train <- all_data[1:891,]
```

```
test <- all_data[892:1309,]
```

```
# Set seed for reproducibility
```

```
set.seed(111)
```

```
# Apply the Random Forest Algorithm
```

```
my_forest <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp +  
Parch + Fare + Embarked + Title,
```

```
data = train, importance = TRUE, ntree = 1000)
```

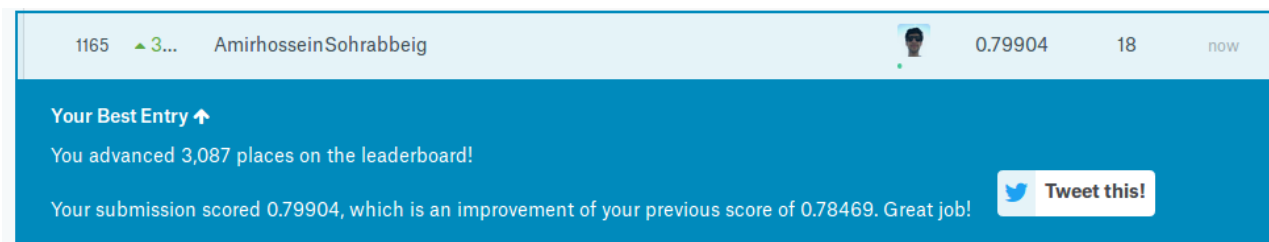
```
# Make your prediction using the test set
```

```
my_prediction <- predict(my_forest, test)
```

# Create a data frame with two columns: PassengerId & Survived. Survived contains your predictions

```
my_solution <- data.frame(PassengerId = test$PassengerId, Survived = my_prediction)
```

# Write your solution away to a csv file with the name my\_solution.csv  
write.csv(my\_solution, file = "last\_prediction.csv", row.names = FALSE)



## سوالات

### 1) DataFrame چیست؟

دیتا فریم (dataFrame) یک ساختمان داده‌ی دوبعدی برچسب گذاری شده است با ستون هایی که می‌توانند از انواع (type) های متفاوت باشند. می‌توان به آن به عنوان یک جدول در زبان اسکیوال یا یک دیشکنری از series نگاه کرد. دیتافریم رایج ترین شی در pandas است. مثل series ها دیتافریم ها هم میتوانند انواع مختلف ای از ورودی ها را بپذیرند

- Dict of 1D ndarrays, lists, dicts, or Series
- 2-D numpy.ndarray
- [Structured or record](#) ndarray
- A Series
- Another DataFrame

### 2) مقصود از Imputation چیست؟

در آمار به عمل جایگزینی داده‌های گمشده با داده‌های جایگزین imputation گفته می‌شود. سه مشکل اصلی وجود دارد که داده‌های گمشده باعث می‌شوند: ۱. داده‌های گشده می‌توانند مقادیر قابل توجهی bias ایجاد کنند ۲. می‌توانند کنترل کردن و تحلیل داده‌ها را سخت تر کنند. ۳. بهره‌وری را کاهش دهند

### 3) normalize کردن داده‌ها به چه منظوری صورت می‌گیرد؟

معمولا داده ها در رنج های متفاوتی قرار گرفته اند که همین باعث میشود مقایسه و کارکردن با آنها مشکل شود برای جلوگیری از این مسئله میتواند داده ها را نرمال کرد. به این صورت که آنها را به رنج ۰ تا ۱ میبریم تا همه ی داده ها در یک رنج قرار بگیرند. و کار کردن با آنها ساده تر شود. به این عمل **normalization** گفته میشود

(4)

آیا در درخت تصمیم تمام ویژگی های مسئله دارای اهمیت یکسانی هستند؟  
خیر اهمیت ویژگی های مختلف مسئله در درخت تصمیم یکسان نیست. قرار دادن کد

```
print(my_tree_one.feature_importances_)
```

در مدل شماره ی چهار که با زبان پایتون نوشته شد نتیجه ی زیر را میدهد:

```
[ 0.12901815  0.31274009  0.223413   0.33482875]
```

که هر عدد نشان از اهمیت ویژگی هایی میدهد که در ساخت درخت استفاده شده. با توجه به اعداد بالا ویژگی Fare اهمیت بالاتری نسبت به بقیه دارد.

برخی از ویژگی ها بهتر از بقیه می توانند در تفکیک داده ها عمل کنند

(5)

دو پارامتر **max\_depth** و **min\_sample\_split** در الگوریتم جنگل تصادفی بیانگر چه چیزی هستند؟

بیشینه عمقی که درختان میتوانند داشته باشند با پارامتر **max\_depth** نشان داده میشود. برای مثال اگر این پارامتر برابر ده باشد هیچ درختی با عمق بیش از ده نمیتواند در جنگل وجود داشته باشد.

کمینه تعداد داده هایی که هر گره داخلی (node هایی که برگ نیستند) میتواند داشته باشد را با **min\_sample\_split** مشخص میکنیم.