**Predict Titanic Survival**

The RMS Titanic set sail on its maiden voyage in 1912, crossing the Atlantic from Southampton, England to New York City. The ship never completed the voyage, sinking to the bottom of the Atlantic Ocean after hitting an iceberg, bringing down 1,502 of 2,224 passengers onboard.

In this project you will create a Logistic Regression model that predicts which passengers survived the sinking of the Titanic, based on features like age and class.

The data we will be using for training our model is provided by Kaggle. Feel free to make the model better on your own and submit it to the [Kaggle Titanic competition](https://www.kaggle.com/c/titanic)!

If you get stuck during this project, check out the **project walkthrough video** which can be found at the bottom of the page after the final step of the project.

Mark the tasks as complete by checking them off

**Load the Data**

1.

The file passengers.csv contains the data of 892 passengers onboard the Titanic when it sank that fateful day. Let's begin by loading the data into a pandas DataFrame named passengers. Print passengers and inspect the columns. What features could we use to predict survival?

**Clean the Data**

2.

Given the saying, "women and children first," Sex and Age seem like good features to predict survival. Let's map the text values in the Sex column to a numerical value. Update Sex such that all values female are replaced with 1 and all values male are replaced with 0.

3.

Let's take a look at Age. Print passengers['Age'].values. You can see we have multiple missing values, or nans. Fill all the empty Age values in passengers with the mean age.

4.

Given the strict class system onboard the Titanic, let's utilize the Pclass column, or the passenger class, as another feature. Create a new column named FirstClass that stores 1 for all passengers in first class and 0 for all other passengers.

5.

Create a new column named SecondClass that stores 1 for all passengers in second class and 0 for all other passengers.

Print passengers and inspect the DataFrame to ensure all the updates have been made.

**Select and Split the Data**

6.

Now that we have cleaned our data, let's select the columns we want to build our model on. Select columns Sex, Age, FirstClass, and SecondClass and store them in a variable named features. Select column Survived and store it a variable named survival.

7.

Split the data into training and test sets using sklearn's train\_test\_split() method. We'll use the training set to train the model and the test set to evaluate the model.

**Normalize the Data**

8.

Since sklearn's Logistic Regression implementation uses *Regularization*, we need to scale our feature data. Create a StandardScaler object, .fit\_transform() it on the training features, and .transform() the test features.

**Create and Evaluate the Model**

9.

Create a LogisticRegression model with sklearn and .fit() it on the training data.

Fitting the model will perform gradient descent to find the feature coefficients that minimize the log-loss for the training data.

10.

.score() the model on the training data and print the training score.

Scoring the model on the training data will run the data through the model and make final classifications on survival for each passenger in the training set. The score returned is the percentage of correct classifications, or the accuracy.

11.

.score() the model on the test data and print the test score.

Similarly, scoring the model on the testing data will run the data through the model and make final classifications on survival for each passenger in the test set.

How well did your model perform?

12.

Print the feature coefficients determined by the model. Which feature is most important in predicting survival on the sinking of the Titanic?

**Predict with the Model**

13.

Let's use our model to make predictions on the survival of a few fateful passengers. Provided in the code editor is information for 3rd class passenger Jack and 1st class passenger Rose, stored in NumPy arrays. The arrays store 4 feature values, in the following order:

* Sex, represented by a 0 for male and 1 for female
* Age, represented as an integer in years
* FirstClass, with a 1 indicating the passenger is in first class
* SecondClass, with a 1 indicating the passenger is in second class

A third array, You, is also provided in the code editor with empty feature values. Update the array You with your information, or for some fictitious passenger. Make sure to enter all values as floats with a .!

14.

Combine Jack, Rose, and You into a single NumPy array named sample\_passengers.

15.

Since our Logistic Regression model was trained on scaled feature data, we must also scale the feature data we are making predictions on. Using the StandardScaler object created earlier, apply its .transform() method to sample\_passengers and save the result to sample\_passengers.

Print sample\_passengers to view the scaled features.

16.

Who will survive, and who will sink? Use your model's .predict() method on sample\_passengers and print the result to find out.

Want to see the probabilities that led to these predictions? Call your model's .predict\_proba() method on sample\_passengers and print the result. The 1st column is the probability of a passenger perishing on the Titanic, and the 2nd column is the probability of a passenger surviving the sinking (which was calculated by our model to make the final classification decision).

17.

If you are stuck on the project or would like to see an experienced developer work through the project, watch the following project walkthrough video! <https://youtu.be/sfwUktfwFzQ>

import codecademylib3\_seaborn  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LogisticRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
  
# Load the passenger data  
passengers = pd.read\_csv('passengers.csv')  
print(passengers.head())  
# PassengerId | Sex | Survived | Pclass | Age | SibSp | Parch | Fare | Embarked | Cabin  
  
# Update sex column to numerical  
sex\_mapping = {'male': 0, 'female': 1}  
passengers['Sex'] = passengers['Sex'].map(sex\_mapping)  
  
# Fill the nan values in the age column  
passengers['Age'].fillna(value=passengers['Age'].mean(), inplace=True)  
print(passengers['Age'].values)  
  
# Create a first class column  
passengers['FirstClass'] = passengers['Pclass'].apply(lambda x: 1 if x == 1 else 0)  
  
# Create a second class column  
passengers['SecondClass'] = passengers['Pclass'].apply(lambda x: 1 if x == 2 else 0)  
print(passengers.head())  
  
# Select the desired features  
features = passengers[['Sex', 'Age', 'FirstClass', 'SecondClass']]  
survial = passengers['Survived']  
  
# Perform train, test, split  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, survial, train\_size = 0.8, test\_size = 0.2, random\_state=1)  
  
  
# Scale the feature data so it has mean = 0 and standard deviation = 1  
scaler = StandardScaler()  
train\_features = scaler.fit\_transform(x\_train)  
test\_features = scaler.fit\_transform(x\_test)  
  
# Create and train the model  
model = LogisticRegression()  
model.fit(train\_features, y\_train)  
  
# Score the model on the train data  
print(model.score(train\_features, y\_train)) # 0.797752808988764  
  
# Score the model on the test data  
print(model.score(test\_features, y\_test)) # 0.7877094972067039  
  
# Analyze the coefficients  
coefficients = model.coef\_  
coefficients = coefficients.tolist()[0]  
print(coefficients)  
# [1.2494638306305015, -0.45605733478432503, 1.0266136315364254, 0.5510979638113181]  
feats = ['Sex', 'Age', 'FirstClass', 'SecondClass']  
plt.bar(list(range(0,4)),coefficients)  
plt.xticks(list(range(0,4)),feats, rotation='vertical')  
plt.title("Visualizing a Feature's Contribution towards Survival")  
plt.xlabel('Feature')  
plt.ylabel('Coefficient')  
plt.show()  
  
# Sample passenger features  
Jack = np.array([0.0,20.0,0.0,0.0])  
Rose = np.array([1.0,17.0,1.0,0.0])  
Jeff = np.array([0.0,31.0,0.0,1.0])  
Lindsay = np.array([1.0,33.0,0.0,1.0])  
  
# Combine passenger arrays  
combined\_arrays = np.array([Jack, Rose, Jeff, Lindsay])  
  
# Scale the sample passenger features  
sample\_passengers = scaler.fit\_transform(combined\_arrays)  
print(sample\_passengers)  
  
# Make survival predictions!  
predicted\_survival = model.predict(sample\_passengers)  
print(predicted\_survival)  
  
proba\_survival = model.predict\_proba(sample\_passengers)  
print(proba\_survival)

