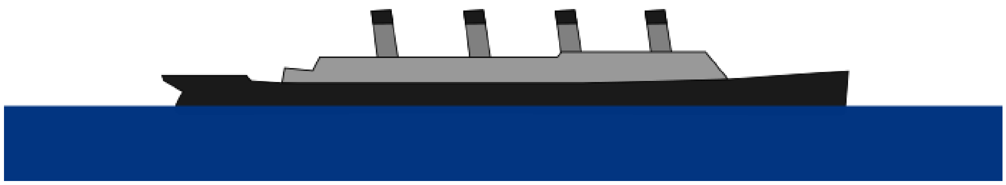
Titanic - a Machine Learning Case Study-Solutions

August 25, 2017

1 Titanic: a Machine Learning Case Study

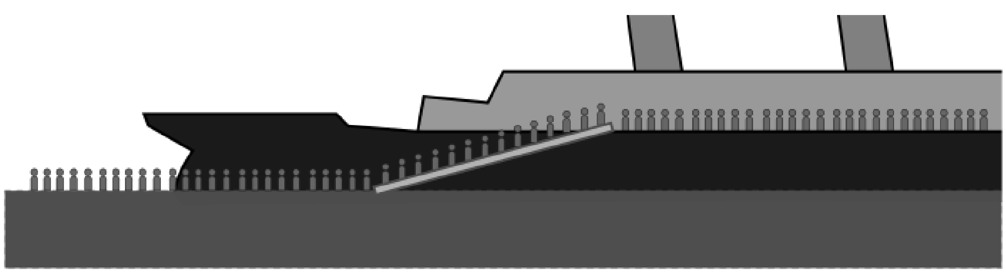


by Dr. Kristian Rother

1.1 Goal

We would like to utilize passenger data to predict whether or not they will survive a trip on the Titanic.

1.2 Part 1: Boarding



1.2.1 1.1 Importing Python Libraries

Import a few Python libraries typically used in Machine Learning:

```
In [1]: import pandas as pd  # handling of tabular data
        import numpy as np   # number crunching
        import pylab as plt  # plotting
```

```
In [2]: %matplotlib inline
```

1.2.2 1.2. Load passenger data

Use `pandas` to load the file `train.csv`.

```
In [3]: df = pd.read_csv('train.csv')
```

You can find a detailed documentation of the dataset on www.kaggle.com/c/titanic.

1.2.3 1.3. Inspect the data

Show the contents of the pandas DataFrame.

```
In [4]: df
```

```
Out[4]:      PassengerId  Survived  Pclass  \
        0              1         0       3
        1              2         1       1
        2              3         1       3
        3              4         1       1
        4              5         0       3
        5              6         0       3
        6              7         0       1
        7              8         0       3
        8              9         1       3
        9             10         1       2
        10            11         1       3
        11            12         1       1
        12            13         0       3
        13            14         0       3
        14            15         0       3
        15            16         1       2
        16            17         0       3
        17            18         1       2
        18            19         0       3
        19            20         1       3
        20            21         0       2
        21            22         1       2
        22            23         1       3
        23            24         1       1
        24            25         0       3
```

```
25            26         1       3
26            27         0       3
27            28         0       1
28            29         1       3
29            30         0       3
..           ...       ...     ...
861          862         0       2
862          863         1       1
863          864         0       3
864          865         0       2
865          866         1       2
866          867         1       2
867          868         0       1
868          869         0       3
869          870         1       3
870          871         0       3
871          872         1       1
872          873         0       1
873          874         0       3
874          875         1       2
875          876         1       3
876          877         0       3
877          878         0       3
878          879         0       3
879          880         1       1
880          881         1       2
881          882         0       3
882          883         0       3
883          884         0       2
884          885         0       3
885          886         0       3
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
5                                     Moran, Mr. James    male   NaN      0
6                              McCarthy, Mr. Timothy J    male  54.0      0
7                       Palsson, Master. Gosta Leonard    male   2.0      3
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                  Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
```

10	Sandstrom, Miss. Marguerite Rut	female	4.0	1
11	Bonnell, Miss. Elizabeth	female	58.0	0
12	Saundercock, Mr. William Henry	male	20.0	0
13	Andersson, Mr. Anders Johan	male	39.0	1
14	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0
15	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0
16	Rice, Master. Eugene	male	2.0	4
17	Williams, Mr. Charles Eugene	male	NaN	0
18	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1
19	Masselmani, Mrs. Fatima	female	NaN	0
20	Fynney, Mr. Joseph J	male	35.0	0
21	Beesley, Mr. Lawrence	male	34.0	0
22	McGowan, Miss. Anna "Annie"	female	15.0	0
23	Sloper, Mr. William Thompson	male	28.0	0
24	Palsson, Miss. Torborg Danira	female	8.0	3
25	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1
26	Emir, Mr. Farred Chehab	male	NaN	0
27	Fortune, Mr. Charles Alexander	male	19.0	3
28	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0
29	Todoroff, Mr. Lalio	male	NaN	0
..	...	...	...	...
861	Giles, Mr. Frederick Edward	male	21.0	1
862	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0
863	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8
864	Gill, Mr. John William	male	24.0	0
865	Bystrom, Mrs. (Karolina)	female	42.0	0
866	Duran y More, Miss. Asuncion	female	27.0	1
867	Roebling, Mr. Washington Augustus II	male	31.0	0
868	van Melkebeke, Mr. Philemon	male	NaN	0
869	Johnson, Master. Harold Theodor	male	4.0	1
870	Balkic, Mr. Cerin	male	26.0	0
871	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1
872	Carlsson, Mr. Frans Olof	male	33.0	0
873	Vander Cruyssen, Mr. Victor	male	47.0	0
874	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1
875	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0
876	Gustafsson, Mr. Alfred Ossian	male	20.0	0
877	Petroff, Mr. Nedelio	male	19.0	0
878	Laleff, Mr. Kristo	male	NaN	0
879	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0
880	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0
881	Markun, Mr. Johann	male	33.0	0
882	Dahlberg, Miss. Gerda Ulrika	female	22.0	0
883	Banfield, Mr. Frederick James	male	28.0	0
884	Sutehall, Mr. Henry Jr	male	25.0	0
885	Rice, Mrs. William (Margaret Norton)	female	39.0	0
886	Montvila, Rev. Juozas	male	27.0	0
887	Graham, Miss. Margaret Edith	female	19.0	0

888	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1
889	Behr, Mr. Karl Howell	male	26.0	0
890	Dooley, Mr. Patrick	male	32.0	0

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S
5	0	330877	8.4583	NaN	Q
6	0	17463	51.8625	E46	S
7	1	349909	21.0750	NaN	S
8	2	347742	11.1333	NaN	S
9	0	237736	30.0708	NaN	C
10	1	PP 9549	16.7000	G6	S
11	0	113783	26.5500	C103	S
12	0	A/5. 2151	8.0500	NaN	S
13	5	347082	31.2750	NaN	S
14	0	350406	7.8542	NaN	S
15	0	248706	16.0000	NaN	S
16	1	382652	29.1250	NaN	Q
17	0	244373	13.0000	NaN	S
18	0	345763	18.0000	NaN	S
19	0	2649	7.2250	NaN	C
20	0	239865	26.0000	NaN	S
21	0	248698	13.0000	D56	S
22	0	330923	8.0292	NaN	Q
23	0	113788	35.5000	A6	S
24	1	349909	21.0750	NaN	S
25	5	347077	31.3875	NaN	S
26	0	2631	7.2250	NaN	C
27	2	19950	263.0000	C23 C25 C27	S
28	0	330959	7.8792	NaN	Q
29	0	349216	7.8958	NaN	S
..	...	...	...	...	...
861	0	28134	11.5000	NaN	S
862	0	17466	25.9292	D17	S
863	2	CA. 2343	69.5500	NaN	S
864	0	233866	13.0000	NaN	S
865	0	236852	13.0000	NaN	S
866	0	SC/PARIS 2149	13.8583	NaN	C
867	0	PC 17590	50.4958	A24	S
868	0	345777	9.5000	NaN	S
869	1	347742	11.1333	NaN	S
870	0	349248	7.8958	NaN	S
871	1	11751	52.5542	D35	S
872	0	695	5.0000	B51 B53 B55	S

```
873       0          345765    9.0000          NaN         S
874       0         P/PP 3381   24.0000         NaN         C
875       0            2667    7.2250          NaN         C
876       0            7534    9.8458          NaN         S
877       0          349212    7.8958          NaN         S
878       0          349217    7.8958          NaN         S
879       1           11767   83.1583          C50         C
880       1          230433   26.0000          NaN         S
881       0          349257    7.8958          NaN         S
882       0            7552   10.5167          NaN         S
883       0   C.A./SOTON 34068  10.5000         NaN         S
884       0     SOTON/OQ 392076  7.0500         NaN         S
885       5          382652   29.1250          NaN         Q
886       0          211536   13.0000          NaN         S
887       0          112053   30.0000          B42         S
888       2       W./C. 6607   23.4500          NaN         S
889       0          111369   30.0000         C148         C
890       0          370376    7.7500          NaN         Q

[891 rows x 12 columns]
```

```
In [5]: df['Survived'].value_counts()
```

```
Out[5]: 0    549
        1    342
        Name: Survived, dtype: int64
```

1.2.4 Challenge

Examine the distribution of values in two other columns of the dataset using the `value_counts()` function.

```
In [6]: df['Sex'].value_counts()
```

```
Out[6]: male      577
        female    314
        Name: Sex, dtype: int64
```

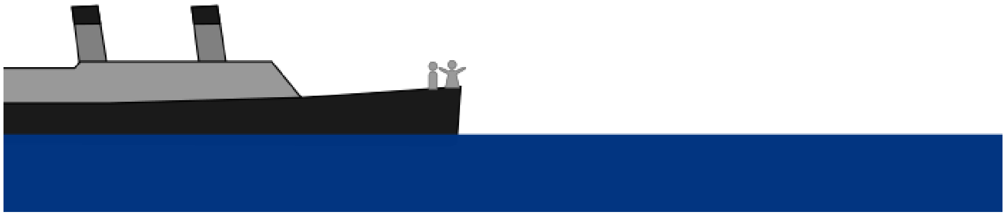
```
In [7]: df['Pclass'].value_counts()
```

```
Out[7]: 3    491
        1    216
        2    184
        Name: Pclass, dtype: int64
```

1.3 Part 2: The Beauty of the Sea

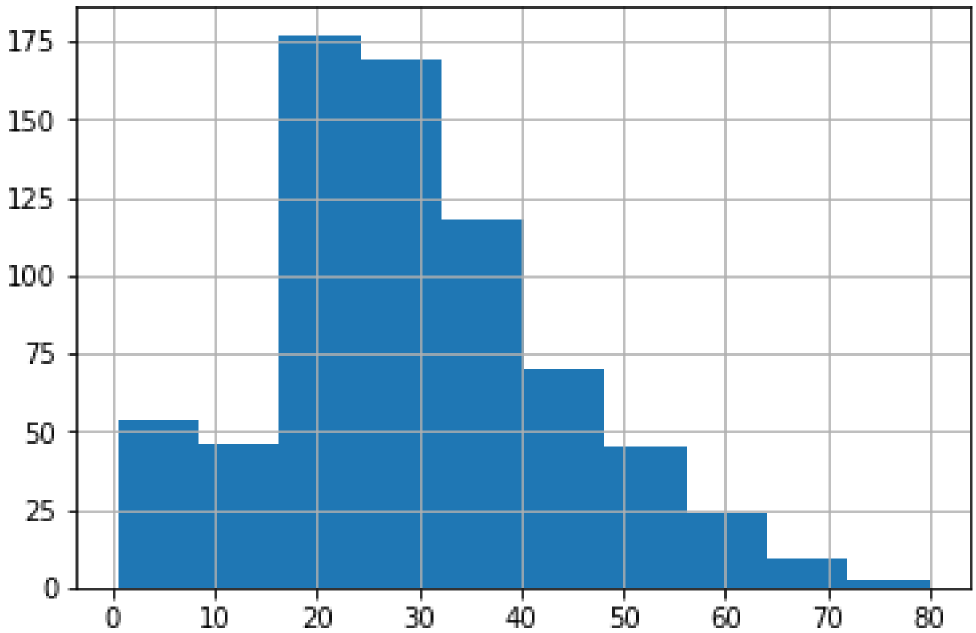
1.3.1 2.1 Draw a histogram

Create a histogram grouping the passengers by age:



```
In [8]: df['Age'].hist()
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8f7610ea90>
```

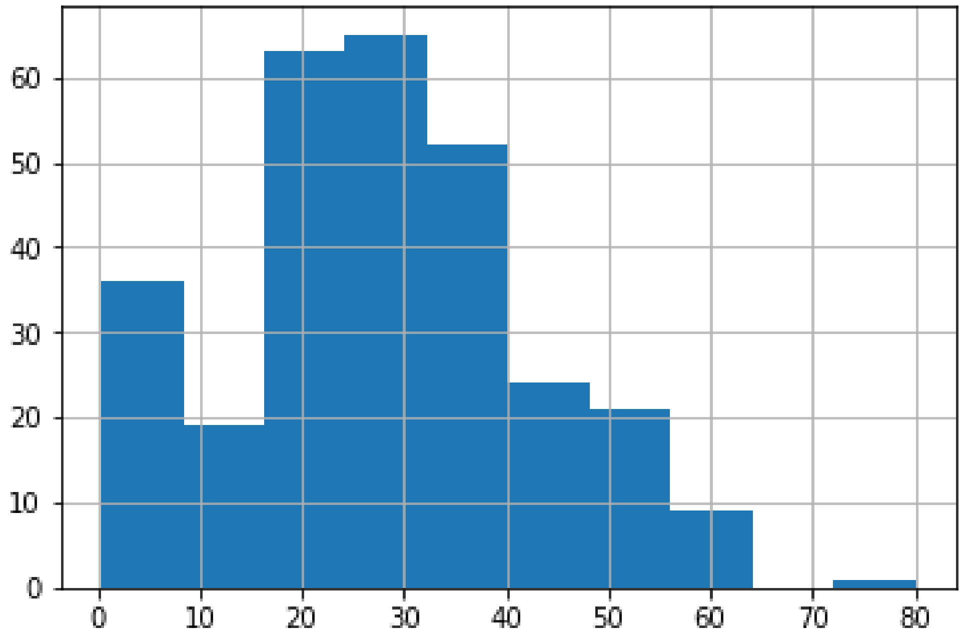


1.3.2 Challenge

Explain the following line.

```
In [9]: df[df['Survived']==1]['Age'].hist()
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8f73970128>
```

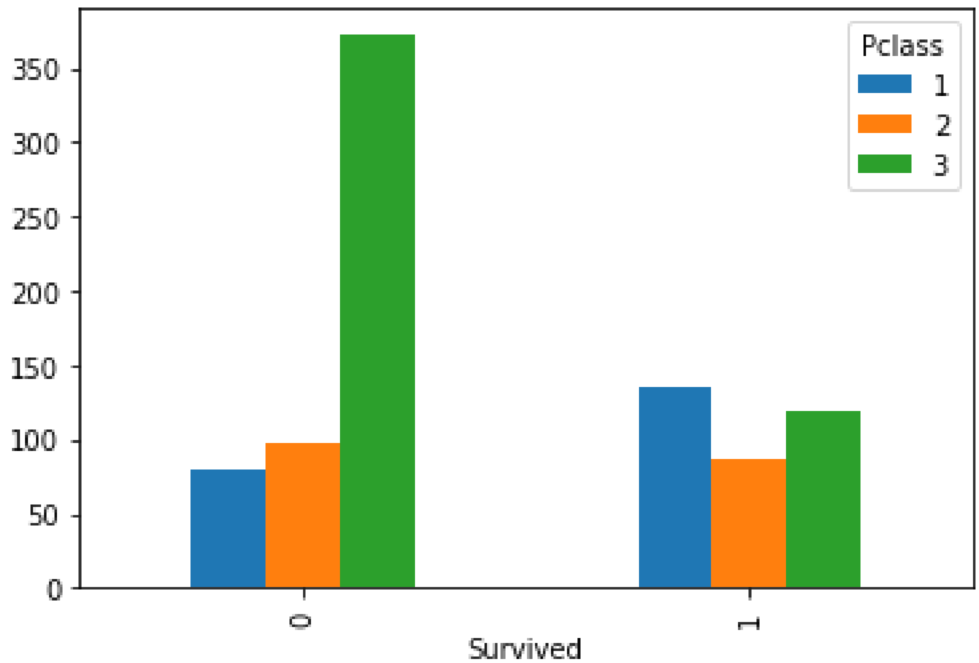


1.3.3 2.2 Bar plot

Create a bar plot that groups the passenger class by survival:

```
In [10]: g = df.groupby(['Survived', 'Pclass'])
         g = g['Name'].count()
         g = g.unstack()
         g.plot.bar()
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8f738965c0>
```

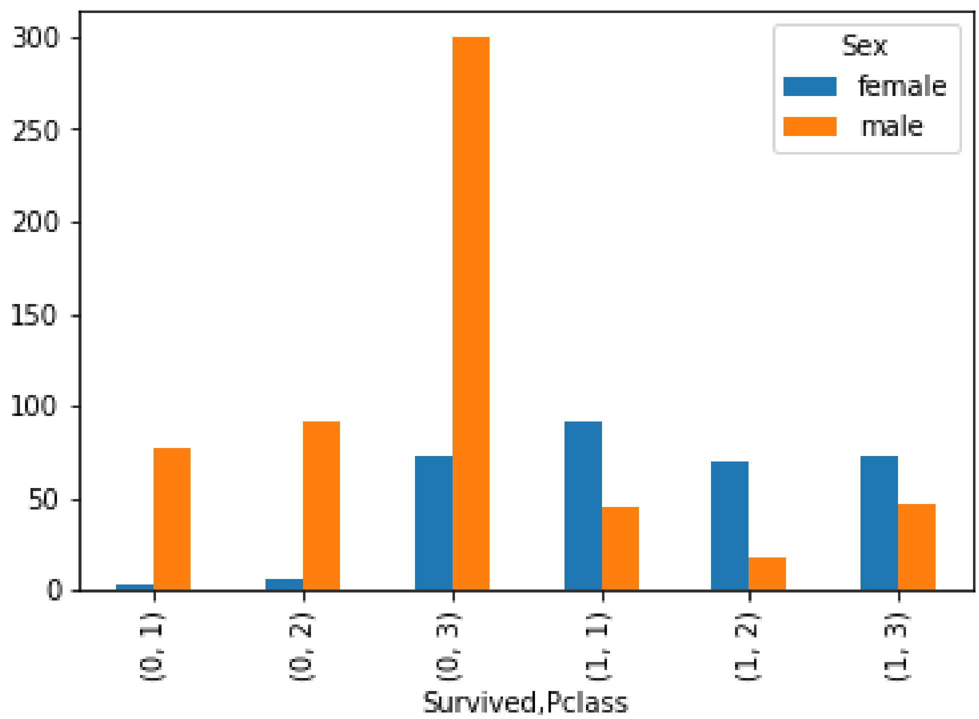


1.3.4 Challenge

Create another bar plot, this time group the bars by gender.

```
In [11]: g = df.groupby(['Survived', 'Pclass', 'Sex'])
         g = g['Name'].count()
         g = g.unstack()
         g.plot.bar()
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8f73789d30>
```



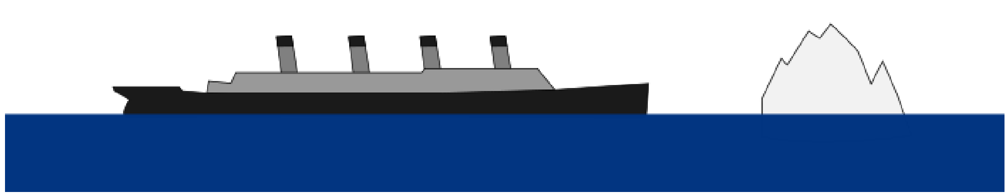
1.3.5 2.3 Hypotheses

Collect ideas which **features** of passengers increase their chances of survival and which decrease them. Only after that start building a model.

Observations:

- children are more likely to survive
- passengers from class 1+2 are more likely to survive
- women are more likely to survive

1.4 Part 3: Collision Course



1.5 3.1 Data wrangling

At this point we need to clean and reshape the data a bit.

- Remove all columns but `"Pclass"`, `"Age"`, `"Sex"` and `"Survived"`.
- Remove all lines containing missing data.
- Convert all **input features** to a matrix `X`.
- Convert the **target column** to an 1D-array `y`.

```
In [12]: cleaned = df[['Pclass', 'Age', 'Sex', 'Survived']]
         cleaned = cleaned.dropna()
```

```
In [13]: X = cleaned[['Pclass', 'Age']]
         X = X.values

         y = cleaned[['Survived']]
         y = y.values.ravel()
```

1.5.1 Challenge

View the dataset as a table before and after the data wrangling step.

```
In [14]: X, y
```

```
Out[14]: (array([[  3.,  22.],
                [  1.,  38.],
                [  3.,  26.],
                ...,
                [  1.,  19.],
                [  1.,  26.],
                [  3.,  32.]]),
          array([0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
                1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
                0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
                0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
                0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
                1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
                1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
                1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
                0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
                0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
                1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
                1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
                0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0,
```

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

1.5.2 3.2 Create a Training/Test set

Split the data into a training and a test set:

In [15]:
```python
from sklearn.model_selection import train_test_split

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state=42)
```

In [16]:
```python
Xtrain.shape
```

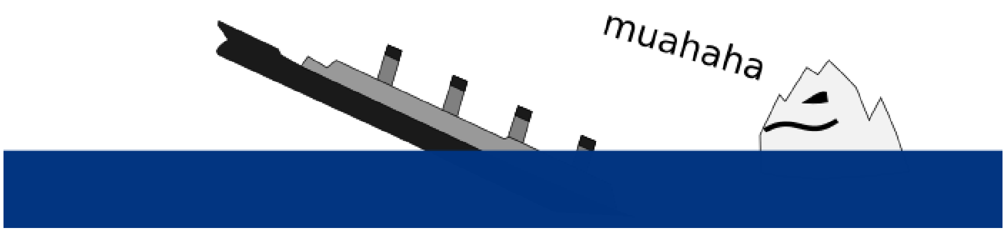
Out[16]:
```
(535, 2)
```

1.5.3 Question

- Why do we need to create a separate test set?

Answer: To check our model on *independent* data.

1.6 Part 4: Modeling and Prediction



1.6.1 4.1 Build a logistic regression model

Create a Machine Learning model using logistic regression and fit it with the training data:

```
In [17]: from sklearn.linear_model import LogisticRegression

         m = LogisticRegression()
         m.fit(Xtrain, ytrain)

Out[17]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

1.6.2 4.2 Evaluate the model

Calculate the accuracy of the model for the training data:

```
In [18]: m.score(Xtrain, ytrain)

Out[18]: 0.69719626168224302
```

With a *skewed dataset*, a confusion matrix is more robust:

```
In [19]: from sklearn.metrics import confusion_matrix

         ypred = m.predict(Xtrain)
         confusion_matrix(ytrain, ypred)

Out[19]: array([[266,  51],
                [111, 107]])
```

1.6.3 Challenge

Calculate the accuracy for the test data as well. Explain the differences.

```
In [20]: m.score(Xtest, ytest)

Out[20]: 0.70949720670391059
```

1.6.4 Question

Is this a good result? Why or why not?

Answer:

- 70% is better than a random coin toss 8505)
- 70% is only a bit better than always predicting "will not survive" (which gives 60% because the data is skewed)
- We still have a lot more data to use, so there is room for improvement!

1.6.5 4.3 More features

We will add more data to the prediction: gender. To use the data, we need to convert it to numbers using **one-hot encoding**.

```
In [21]: gender = pd.get_dummies(cleaned['Sex'])
         gender
```

```
Out[21]:      female  male
         0         0     1
         1         1     0
         2         1     0
         3         1     0
         4         0     1
         6         0     1
         7         0     1
         8         1     0
         9         1     0
         10        1     0
         11        1     0
         12        0     1
         13        0     1
         14        1     0
         15        1     0
         16        0     1
         18        1     0
         20        0     1
         21        0     1
         22        1     0
         23        0     1
         24        1     0
         25        1     0
         27        0     1
         30        0     1
         33        0     1
         34        0     1
         35        0     1
         37        0     1
         38        1     0
         ..      ...   ...
         856       1     0
         857       0     1
         858       1     0
         860       0     1
         861       0     1
         862       1     0
         864       0     1
         865       1     0
         866       1     0
```

```
867       0     1
869       0     1
870       0     1
871       1     0
872       0     1
873       0     1
874       1     0
875       1     0
876       0     1
877       0     1
879       1     0
880       1     0
881       0     1
882       1     0
883       0     1
884       0     1
885       1     0
886       0     1
887       1     0
889       0     1
890       0     1

[714 rows x 2 columns]
```

Of course, we need to add the column to the input table (one is enough).

```
In [22]: cleaned['female'] = gender['female']
```

1.6.6 Challenge

Re-run the prediction above using the additional feature. How does the accuracy change?

```
In [23]: X = cleaned[['Pclass', 'Age', 'female']]
         X = X.values

         y = cleaned[['Survived']]
         y = y.values.ravel()

         Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state=42)

         m = LogisticRegression()
         m.fit(Xtrain, ytrain)

Out[23]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)

In [24]: m.score(Xtrain, ytrain)
```

```
Out[24]: 0.8018691588785046
```

```
In [25]: m.score(Xtest, ytest)
```

```
Out[25]: 0.7821229050279329
```

1.6.7 4.4 Try a Random Forest Model

Let's try a different model: The Random Forest (an **ensemble of decision trees**)

```
In [27]: from sklearn.ensemble import RandomForestClassifier

         m = RandomForestClassifier()
```

1.6.8 Challenge

Fit the Random Forest model to the training data yourself and evaluate it on the test set.

```
In [28]: m.fit(Xtrain, ytrain)
         m.score(Xtrain, ytrain)
```

```
Out[28]: 0.9065420560747663
```

```
In [30]: m.score(Xtest, ytest)
         # dramatic overfitting!
```

```
Out[30]: 0.7821229050279329
```

Compare how the following parameters affect prediction quality:

```
In [ ]: m1 = RandomForestClassifier(max_depth=2)
        m2 = RandomForestClassifier(max_depth=3)
        m3 = RandomForestClassifier(max_depth=10)
```

Limiting the complexity of a model is called **regularization**

```
In [33]: m = RandomForestClassifier(max_depth=2)
         m.fit(Xtrain, ytrain)
         m.score(Xtrain, ytrain), m.score(Xtest, ytest)
         # neither 2 or 3 is ideal, further tweaking will be necessary.
```

```
Out[33]: (0.8074766355140187, 0.7653631284916201)
```

1.7 Part 5: Prediction

Create a data set for additional passengers and predict whether they will survive:

```
In [34]: leo = np.array([[22, 3, 0]])
         kate = np.array([[25, 1, 1]])

         print(m.predict(leo))
         print(m.predict(kate))
```

```
[0]
[1]
```

1.7.1 Challenge

There is (at least) one error in the definition of the data for prediction. Can you find and fix it?

```
In [36]: # swapped order of values, hard to spot.
         leo = np.array([[3, 22, 0]])
         kate = np.array([[1, 25, 1]])

         print(m.predict(leo))
         print(m.predict(kate))
```

```
[0]
[1]
```

```
In [ ]:
```