Entrée [1]:

```
# Badr TADJER | Naoufal ARRADI | Leo TRAN
# M1-LS1-APP
```

Predict survival on the Titanic

In this Lab, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy

Dataset

The dataset contains 891 observations of 12 variables:

- **PassengerId**: Unique ID for each passenger
- **Survived**: Survival (0 = No; 1 = Yes)
- **Pclass**: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- **Name**: Name
- **Sex**: Sex
- **Age**: Age
- **Sibsp**: Number of Siblings/Spouses Aboard
- **Parch**: Number of Parents/Children Aboard
- **Ticket**: Ticket Number
- **Fare**: Passenger Fare
- **Cabin**: Cabin
- **Embarked** Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Entrée [2]:

```
# imports
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
# your code here
```

Entrée [3]:

```
titanic = pd.read_csv('titanic.csv', index_col = 0)
titanic.head()
```

Out[3]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabir
PassengerId										
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

Entrée [4]:

```
# print some info about the dataframe
# your code here
titanic.info
```

Out[4]:

```
<bound method DataFrame.info of              Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3
...               ...     ...
887                 0       2
888                 1       1
889                 0       3
890                 1       1
891                 0       3

                                                          Name     Sex
Age  \
PassengerId
1                                      Braund, Mr. Owen Harris    male
22.0
2            Cumings, Mrs. John Bradley (Florence Briggs Th...  female
38.0
3                                       Heikkinen, Miss. Laina  female
26.0
4                 Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
35.0
5                                     Allen, Mr. William Henry    male
35.0
...                                                        ...     ...
...
887                                      Montvila, Rev. Juozas    male
27.0
888                               Graham, Miss. Margaret Edith  female
19.0
889                   Johnston, Miss. Catherine Helen "Carrie"  female
NaN
890                                      Behr, Mr. Karl Howell    male
26.0
891                                        Dooley, Mr. Patrick    male
32.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
...            ...    ...               ...      ...   ...      ...
887              0      0            211536  13.0000   NaN        S
888              0      0            112053  30.0000   B42        S
889              1      2        W./C. 6607  23.4500   NaN        S
890              0      0            111369  30.0000  C148        C
891              0      0            370376   7.7500   NaN        Q
```

```
[891 rows x 11 columns]>
```

Looks like there are some Nan values, let's see how many for each column

Entrée [5]:

```
titanic.isnull().sum()
```

Out[5]:

```
Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Cabin contains a lot of Nan values, we'll drop this column
We'll replace the Nan values in **Age** with the age's median, and the ones in **Embarked** with **'S'**, which is the most frequent one in this column

Entrée [6]:

```
# your code here to drop Cabin
titanic = titanic.drop(columns = "Cabin")
# check the fillna documentation: http://pandas.pydata.org/pandas-docs/stable/genera
titanic["Age"] = titanic["Age"].fillna(titanic["Age"].median())
titanic["Embarked"].fillna('S', inplace = True)
titanic.isnull().sum()
```

Out[6]:

```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    0
dtype: int64
```

Visualization

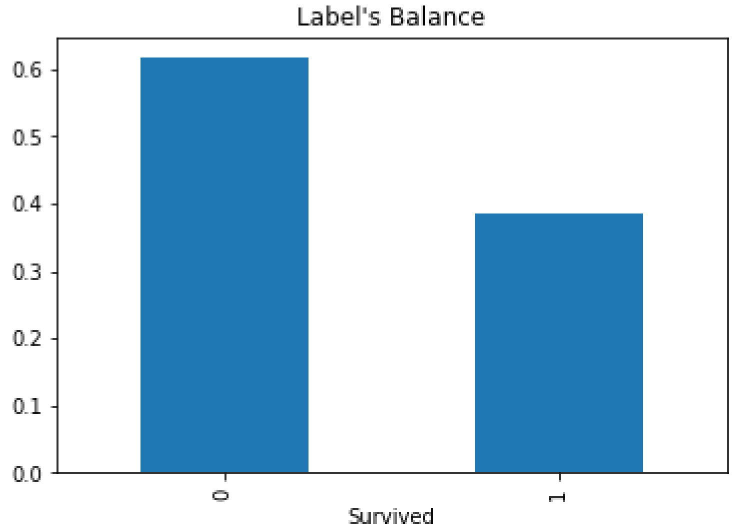
Entrée [7]:

```
%matplotlib inline
import matplotlib.pyplot as plt
print ('survival rate =', titanic.Survived.mean())
(titanic.groupby('Survived').size() / titanic.shape[0]).plot(kind = "bar", title = '
```

```
survival rate = 0.3838383838383838
```

Out[7]:

```
<AxesSubplot:title={'center':"Label's Balance"}, xlabel='Survived'>
```



Entrée [8]:

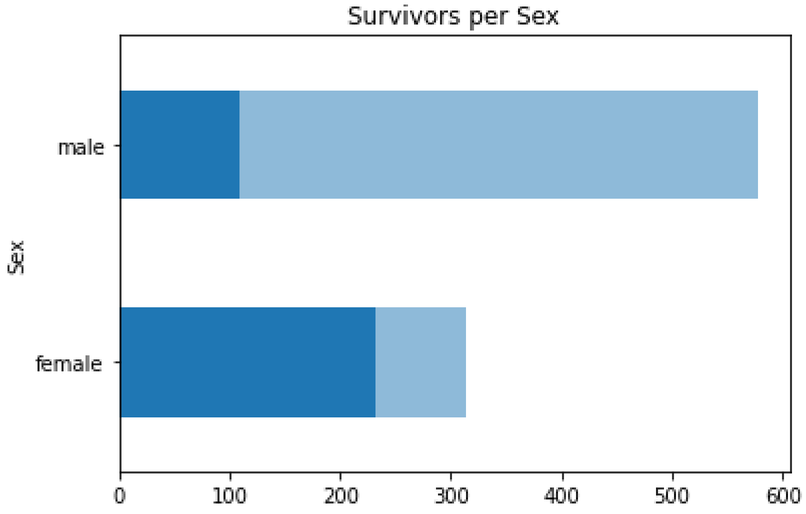
```
# make a function to plot survival against passenger attribute
def survival_rate(column, t):
    df = pd.DataFrame()
    df['total'] = titanic.groupby(column).size()
    df['survived'] = titanic.groupby(column).sum()['Survived']
    df['percentage'] = round(df['survived'] / df['total'] * 100,2)
    print(df)

    df['survived'].plot(kind = t)
    df['total'].plot(kind = t, alpha = 0.5, title = "Survivors per " + str(column))
    plt.show()
```

Entrée [9]:

```
# Draw survival per Sex
survival_rate("Sex", "barh")
```

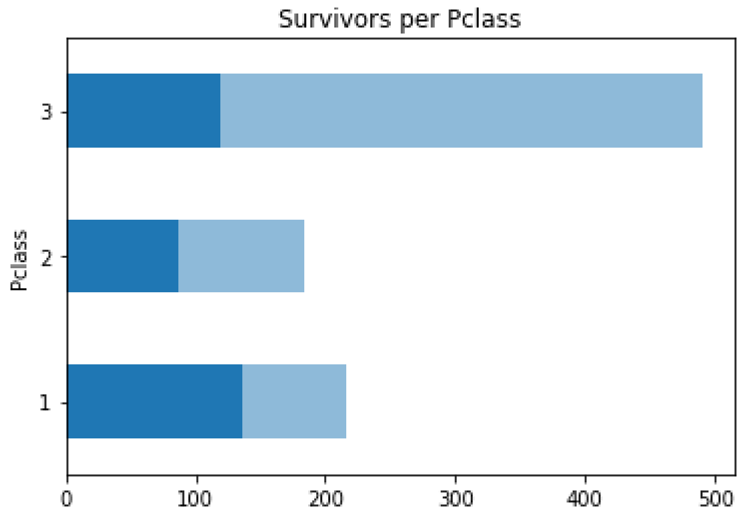
```
        total  survived  percentage
Sex
female    314       233       74.20
male      577       109       18.89
```



Entrée [10]:

```
# Draw survival per Class
survival_rate("Pclass", "barh")
```

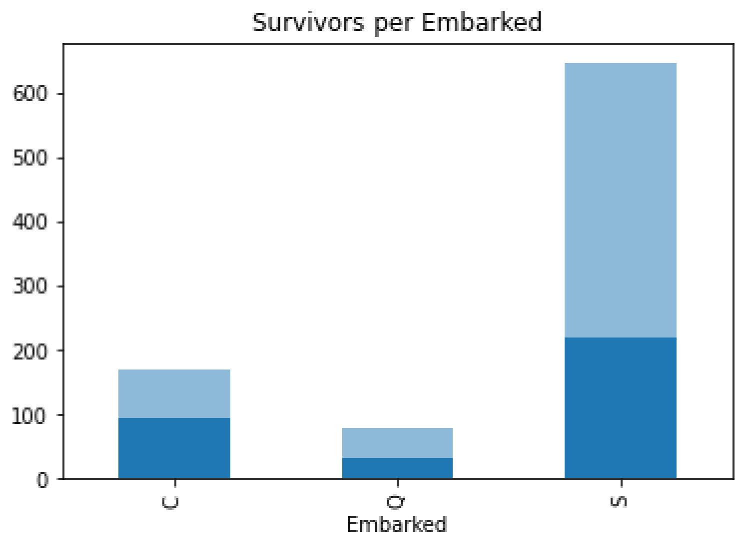
```
        total  survived  percentage
Pclass
1         216       136       62.96
2         184        87       47.28
3         491       119       24.24
```



Entrée [11]:

```
# Graph survived per port of embarkation
survival_rate("Embarked", "bar")
```

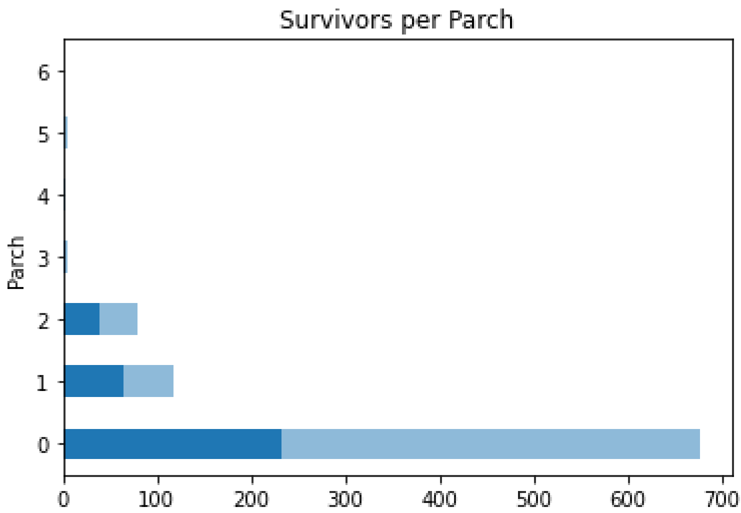
```
          total  survived  percentage
Embarked
C           168        93       55.36
Q            77        30       38.96
S           646       219       33.90
```



Entrée [12]:

```
# Draw survived per Number of Parents/Children Aboard (Parch)
survival_rate("Parch", "barh")
```

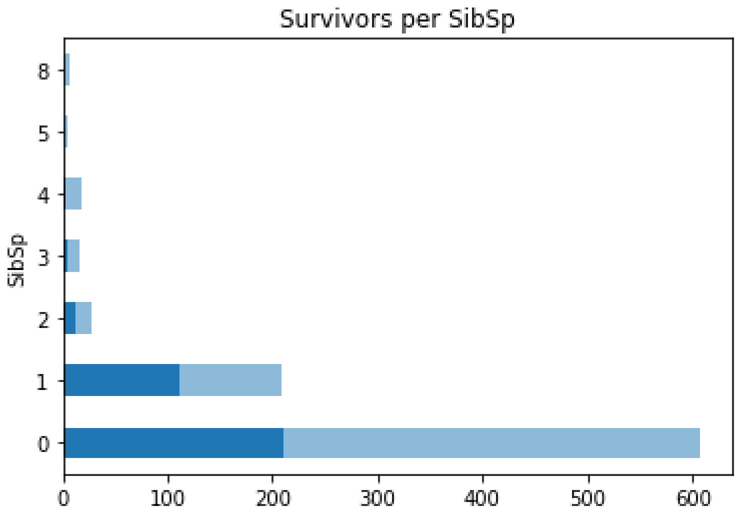
```
       total  survived  percentage
Parch
0        678       233       34.37
1        118        65       55.08
2         80        40       50.00
3          5         3       60.00
4          4         0        0.00
5          5         1       20.00
6          1         0        0.00
```



Entrée [13]:

```
# Draw survived per Number of Siblings/Spouses Aboard (SibSp)
survival_rate("SibSp", "barh")
```

```
       total  survived  percentage
SibSp
0        608       210       34.54
1        209       112       53.59
2         28        13       46.43
3         16         4       25.00
4         18         3       16.67
5          5         0        0.00
8          7         0        0.00
```



Model training

Some of the columns don't have predictive power, so let's specify which ones are included for prediction

Entrée [14]:

```
predictors = ["Pclass", "Sex", "Age", 'SibSp', 'Parch', "Fare", "Embarked"]
```

We need now to convert text columns in **predictors** to numerical ones

Entrée [15]:

```
for col in predictors: # Loop through all columns in predictors
    if titanic[col].dtype == 'object':  # check if column's type is object (text)
        titanic[col] = pd.Categorical(titanic[col]).codes  # convert text to numeric

titanic.head()
```

Out[15]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embark
1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	
5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	

Entrée [16]:

```
# Split the data into a training set and a testing set. Set: test_size=0.3, random_s
from sklearn.model_selection import train_test_split
# your code here

y = titanic["Survived"]
X = titanic[predictors]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_st

print ("train shape", X_train.shape, y_train.shape)
print ("test shape", X_test.shape, y_test.shape)
```

```
train shape (623, 7) (623,)
test shape (268, 7) (268,)
```

Entrée [17]:

```
# import LogisticRegression from: http://scikit-learn.org/stable/modules/generated/s

# your code here
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state = 1)
clf = clf.fit(X_train, y_train)

# your code here
train_score = clf.score(X_train, y_train)
test_score = clf.score(X_test, y_test)

print ('train accuracy =', train_score)
print ('test accuracy =', test_score)
```

```
train accuracy = 0.8073836276083467
test accuracy = 0.7723880597014925
```

Let's print the model's parameters

Entrée [18]:

```
coeff = pd.DataFrame()
coeff['Feature'] = X_train.columns
coeff['Coefficient Estimate'] = pd.Series(clf.coef_[0])
coeff.loc[len(coeff)]=['Intercept', clf.intercept_[0]]
print(coeff)
```

```
     Feature  Coefficient Estimate
0     Pclass             -1.158693
1        Sex             -2.708761
2        Age             -0.040634
3      SibSp             -0.334012
4      Parch              0.071939
5       Fare             -0.000570
6   Embarked             -0.223307
7  Intercept              5.391545
```

We now need to predict class labels for the test set. We will also generate the class probabilities

Entrée [19]:

```
# predict class labels for the test set
y_pred = clf.predict(X_test)
print (y_pred)
```

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

Entrée [20]:

```
# generate class probabilities : http://scikit-learn.org/stable/modules/generated/sk
y_probs = clf.predict_proba(X_test)
print (y_probs)
```

```
[[0.13931138 0.86068862]
 [0.91545699 0.08454301]
 [0.12666421 0.87333579]
 [0.36591842 0.63408158]
 [0.07784301 0.92215699]
 [0.89067951 0.10932049]
 [0.80772606 0.19227394]
 [0.11922783 0.88077217]
 [0.49862954 0.50137046]
 [0.44730682 0.55269318]
 [0.90553194 0.09446806]
 [0.39942792 0.60057208]
 [0.75883959 0.24116041]
 [0.7731844  0.2268156 ]
 [0.36586717 0.63413283]
 [0.51884413 0.48115587]
 [0.92555342 0.07444658]
 [0.92813017 0.07186983]
 [0.93745225 0.06254775]
```

As you can see, the classifier outputs two probabilities for each row. It's predicting a 1 (Survived) any time the probability in the second column is greater than 0.5. Let's visualize it all together.

Entrée [21]:

```
pred = pd.DataFrame({
        "Survived_original": y_test,
        "Survived_predicted": y_pred,
        "Survived_proba": np.transpose(y_probs)[1]
        })
pred["Comparison"] = pred.Survived_original == pred.Survived_predicted
pred.head()
```

Out[21]:

PassengerId	Survived_original	Survived_predicted	Survived_proba	Comparison
863	1	1	0.860689	True
224	0	0	0.084543	True
85	1	1	0.873336	True
681	0	1	0.634082	False
536	1	1	0.922157	True

Confusion matrix

Entrée [22]:

```
from sklearn import metrics
print (metrics.confusion_matrix(y_test, y_pred))
print (metrics.classification_report(y_test, y_pred))
```

```
[[129  24]
 [ 37  78]]
              precision    recall  f1-score   support

           0       0.78      0.84      0.81       153
           1       0.76      0.68      0.72       115

    accuracy                           0.77       268
   macro avg       0.77      0.76      0.76       268
weighted avg       0.77      0.77      0.77       268
```

As you can see, we can have the classification report for each class

K-Fold Cross Validation

Entrée [23]:

```
# import cross_validation from: http://scikit-learn.org/stable/modules/generated/skl
# your code here
from sklearn.model_selection import cross_val_score
clf = LogisticRegression(random_state = 1)
scores = cross_val_score(clf, titanic[predictors], titanic["Survived"], scoring = 'a
## see model
print(scores)
# Take the mean of the scores (because we have one for each fold)
print(scores.mean())
```

```
[0.7877095  0.78651685 0.78089888 0.76966292 0.82022472]
0.7890025735986442
```

When you are improving a model, you want to make sur that you are really doing it and not just being lucky. This is why it's good to work with cross validation instead of one train/test split.

Entrée [ ]:

```

```