

Chapter 3 - Regression Models

Part 3 - Logistic Linear Regression

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import sklearn

from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
```

```
In [4]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict

from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
```

```
In [7]: %matplotlib inline
rcParams['figure.figsize'] = 5,4
sb.set_style('whitegrid')
```

Logistic regression on the titantic dataset

```
In [8]: address = address = 'C:/Users/danal/Desktop/Ex_Files_Python_Data_Science_EssT_Pt2/
titanic_training = pd.read_csv(address)
titanic_training.columns = ['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
print(titanic_training.head())
```

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	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	

	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

```
In [9]: print(titanic_training.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  891 non-null    int64
1   Survived     891 non-null    int64
2   Pclass       891 non-null    int64
3   Name         891 non-null    object
4   Sex          891 non-null    object
5   Age         714 non-null    float64
6   SibSp        891 non-null    int64
7   Parch        891 non-null    int64
8   Ticket       891 non-null    object
9   Fare         891 non-null    float64
10  Cabin        204 non-null    object
11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

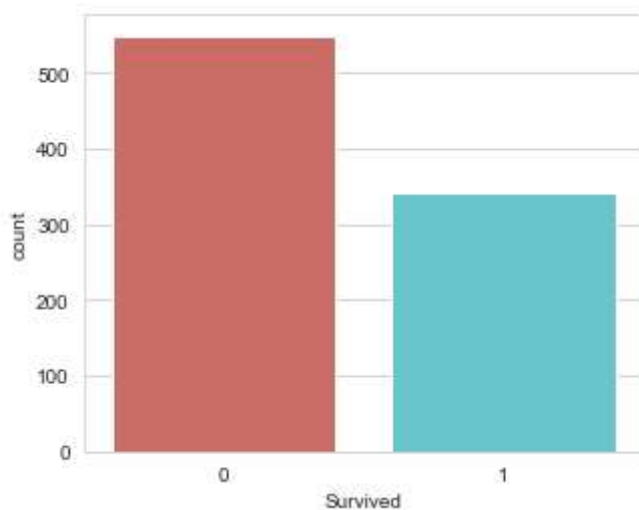
VARIABLE DESCRIPTIONS
Survived - Survival (0 = No; 1 = Yes)
Pclass - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Name - Name
Sex - Sex
Age - Age

SibSp - Number of Siblings/Spouses Aboard
Parch - Number of Parents/Children Aboard
Ticket - Ticket Number
Fare - Passenger Fare (British pound)
Cabin - Cabin
Embarked - Port of Embarkation (C = Cherbourg, France; Q = Queenstown, UK; S = Southampton - Cobh, Ireland)

Checking that your target variable is binary

```
In [11]: sb.countplot(x='Survived', data=titanic_training, palette='hls')
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1f05bc99f48>
```



Checking for missing values

```
In [13]: titanic_training.isnull().sum()
```

```
Out[13]: 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 [15]: titanic_training.describe()
```

```
Out[15]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Taking care of missing values

Dropping missing values

So let's just go ahead and drop all the variables that aren't relevant for predicting survival.

We should at least keep the following:

Survived - This variable is obviously relevant.

Pclass - Does a passenger's class on the boat affect their survivability?

Sex - Could a passenger's gender impact their survival rate?

Age - Does a person's age impact their survival rate?

SibSp - Does the number of relatives on the boat (that are siblings or a spouse) affect a person survivability? Probability

Parch - Does the number of relatives on the boat (that are children or parents) affect a person survivability? Probability

Fare - Does the fare a person paid effect his survivability? Maybe - let's keep it.

Embarked - Does a person's point of embarkation matter? It depends on how the boat was filled... Let's keep it.

What about a person's name, ticket number, and passenger ID number? They're irrelevant for predicting survivability.

And as you recall, the cabin variable is almost all missing values, so we can just drop all of these.

```
In [16]: titanic_data = titanic_training.drop(['Name', 'Ticket', 'Cabin'], axis=1)
titanic_data.head()
```

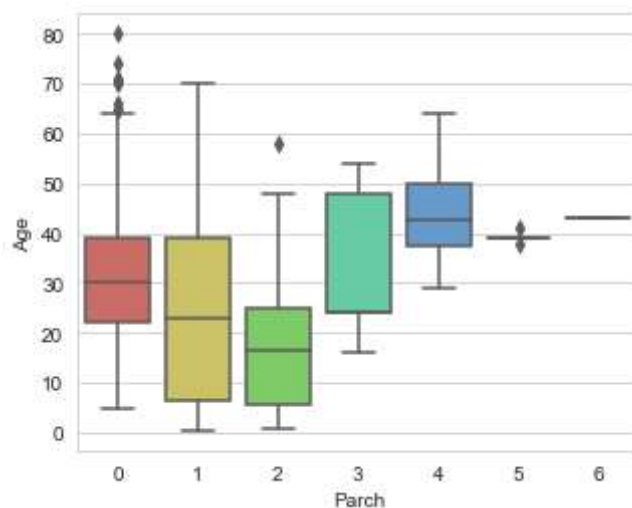
Out[16]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

Imputing missing values

```
In [18]: sb.boxplot(x='Parch', y='Age', data=titanic_data, palette='hls')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1f05d80b308>



```
In [19]: Parch_groups = titanic_data.groupby(titanic_data['Parch'])
Parch_groups.mean()
```

Out[19]:

	PassengerId	Survived	Pclass	Age	SibSp	Fare
Parch						
0	445.255162	0.343658	2.321534	32.178503	0.237463	25.586774
1	465.110169	0.550847	2.203390	24.422000	1.084746	46.778180
2	416.662500	0.500000	2.275000	17.216912	2.062500	64.337604
3	579.200000	0.600000	2.600000	33.200000	1.000000	25.951660
4	384.000000	0.000000	2.500000	44.500000	0.750000	84.968750
5	435.200000	0.200000	3.000000	39.200000	0.600000	32.550000
6	679.000000	0.000000	3.000000	43.000000	1.000000	46.900000

```
In [21]: def age_approx(cols):
Age = cols[0]
Parch = cols[1]

    if pd.isnull(Age):
        if Parch == 0:
            return 32
        elif Parch == 1:
            return 24
        elif Parch == 2:
            return 17
        elif Parch == 3:
            return 33
        elif Parch == 4:
            return 45
        else:
            return 30
    else:
        return Age
```

```
In [22]: titanic_data['Age'] = titanic_data[['Age', 'Parch']].apply(age_approx, axis=1)
titanic_data.isnull().sum()
```

Out[22]:

PassengerId	0
Survived	0
Pclass	0
Sex	0
Age	0
SibSp	0
Parch	0
Fare	0
Embarked	2
dtype:	int64

```
In [23]: titanic_data.dropna(inplace=True)
titanic_data.reset_index(inplace=True, drop=True)

print(titanic_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 889 entries, 0 to 888
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId     889 non-null    int64
1   Survived        889 non-null    int64
2   Pclass          889 non-null    int64
3   Sex             889 non-null    object
4   Age             889 non-null    float64
5   SibSp           889 non-null    int64
6   Parch           889 non-null    int64
7   Fare            889 non-null    float64
8   Embarked        889 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 62.6+ KB
None
```