# Lab 26 - k-Nearest Neighbors classifier 2

We will continue using the Titanic training and test data from Kaggle (https://www.kaggle.com/c/titanic) from Lab 24 and 25.

First import the necessary libraries.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
%matplotlib inline
```

## Loading and cleaning the data

The code for loading in the data and cleaning it from Lab 25 is below.

In [2]:
```
train = pd.read_csv("train.csv")
train.head()
```

Out[2]:

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

In [3]:

```
# fill in missing age data
train["Age"] = train["Age"].fillna(train["Age"].median())

# fill in the missing embarked data
train["Embarked"] = train["Embarked"].fillna("S")
```

In [4]:

```
# create dummy variables for passenger class, sex, and embarked
train2 = pd.get_dummies(train, columns = ["Pclass","Sex","Embarked"], drop_first = True)
train2.head()
```

Out[4]:

| | PassengerId | Survived | Name | Age | SibSp | Parch | Ticket | Fare | Cabin | Pclass_2 | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | Braund, Mr. Owen Harris | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | 0 | |
| **1** | 2 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | 0 | |
| **2** | 3 | 1 | Heikkinen, Miss. Laina | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | 0 | |
| **3** | 4 | 1 | Futrelle, Mrs. Jacques Heath | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | 0 | |

In [5]:

```
# remove the remaining qualitative columns
train2.drop("Cabin",axis = 1,inplace = True)
train2.drop("Name",axis = 1,inplace = True)
train2.drop("Ticket",axis = 1,inplace = True)

# we should also drop PassengerId, although we did not last lab
train2.drop("PassengerId",axis = 1, inplace = True)
```

In [6]:

```
# split the original training data into training and test sets
X_train,X_test,y_train, y_test =train_test_split(train2.drop("Survived",axis=1),train2["Survived"],
                                         test_size = 0.2)
```

In [7]:

```
# create a 3-nearest neighbor classifirer
knn = KNeighborsClassifier(n_neighbors=3)
# fit the classifier to the training data
knn.fit(X_train, y_train)
# test and score the classifier on our test data (part of the original training data)
# notice this line corrects a mistake in Lab 25
knn.score(X_test, y_test)
```

Out[7]:

```
0.7262569832402235
```

Now we are going to try running our classifier on the Kaggle test data and use all of our training data to fit the classifier.

First, load the test data from Kaggle into the dataframe test .

In [8]:
```
test = pd.read_csv("test.csv")
test.head()
```

Out[8]:

| | PassengerId | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 892 | 3 | Kelly, Mr. James | male | 34.5 | 0 | 0 | 330911 | 7.8292 | NaN | Q |
| **1** | 893 | 3 | Wilkes, Mrs. James (Ellen Needs) | female | 47.0 | 1 | 0 | 363272 | 7.0000 | NaN | S |
| **2** | 894 | 2 | Myles, Mr. Thomas Francis | male | 62.0 | 0 | 0 | 240276 | 9.6875 | NaN | Q |
| **3** | 895 | 3 | Wirz, Mr. Albert | male | 27.0 | 0 | 0 | 315154 | 8.6625 | NaN | S |
| **4** | 896 | 3 | Hirvonen, Mrs. Alexander (Helga E Lindqvist) | female | 22.0 | 1 | 1 | 3101298 | 12.2875 | NaN | S |

We have to process the test data in the same way as the training data, namely filling in the missing age and embarked data, creating the dummy variables for Pclass, Sex, and Embarked, and dropping the Cabin, Name, and Ticket columns. Do this below, adding as many extra cells as you need.

In [9]:
```
test["Age"] = test["Age"].fillna(test["Age"].median())
test["Embarked"] = test["Embarked"].fillna("S")

test2 = pd.get_dummies(test, columns = ["Pclass","Sex","Embarked"], drop_first = True)

test2.drop("Cabin",axis = 1,inplace = True)
test2.drop("Name",axis = 1,inplace = True)
test2.drop("Ticket",axis = 1,inplace = True)

test2.head()
```

Out[9]:

| | PassengerId | Age | SibSp | Parch | Fare | Pclass_2 | Pclass_3 | Sex_male | Embarked_Q | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 892 | 34.5 | 0 | 0 | 7.8292 | 0 | 1 | 1 | 1 | |
| **1** | 893 | 47.0 | 1 | 0 | 7.0000 | 0 | 1 | 0 | 0 | |
| **2** | 894 | 62.0 | 0 | 0 | 9.6875 | 1 | 0 | 1 | 1 | |
| **3** | 895 | 27.0 | 0 | 0 | 8.6625 | 0 | 1 | 1 | 0 | |
| **4** | 896 | 22.0 | 1 | 1 | 12.2875 | 0 | 1 | 0 | 0 | |

Store the column PassengerId in a variable. We need this information for submitting our predictions

In [10]:
```
passengerId = test2["PassengerId"]
passengerId
```

Out[10]:
```
0       892
1       893
2       894
3       895
4       896
       ...
413    1305
414    1306
415    1307
416    1308
417    1309
Name: PassengerId, Length: 418, dtype: int64
```

Drop the `PassengerId` column from the test data.

In [11]:
```
test2.drop("PassengerId",axis = 1, inplace = True)
```

In [12]:
```
test2.head()
```

Out[12]:

| | Age | SibSp | Parch | Fare | Pclass_2 | Pclass_3 | Sex_male | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 34.5 | 0 | 0 | 7.8292 | 0 | 1 | 1 | 1 | 0 |
| **1** | 47.0 | 1 | 0 | 7.0000 | 0 | 1 | 0 | 0 | 1 |
| **2** | 62.0 | 0 | 0 | 9.6875 | 1 | 0 | 1 | 1 | 0 |
| **3** | 27.0 | 0 | 0 | 8.6625 | 0 | 1 | 1 | 0 | 1 |
| **4** | 22.0 | 1 | 1 | 12.2875 | 0 | 1 | 0 | 0 | 1 |

Next we split up our training data into the answer (the `Survived` column) and the input data (all other columns).

First, store the `Survived` column in the variable `y_train_kaggle`.

In [13]:
```
y_train_kaggle = train2["Survived"]
y_train_kaggle.head()
```

Out[13]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

Next, drop the `Survived` column from the training data, and store the new data frame in the variable `X_train_kaggle`.

In [14]:
```
X_train_kaggle = train2.drop("Survived",axis = 1)
X_train_kaggle.head()
```

Out[14]:

| | Age | SibSp | Parch | Fare | Pclass_2 | Pclass_3 | Sex_male | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 22.0 | 1 | 0 | 7.2500 | 0 | 1 | 1 | 0 | 1 |
| **1** | 38.0 | 1 | 0 | 71.2833 | 0 | 0 | 0 | 0 | 0 |
| **2** | 26.0 | 0 | 0 | 7.9250 | 0 | 1 | 0 | 0 | 1 |
| **3** | 35.0 | 1 | 0 | 53.1000 | 0 | 0 | 0 | 0 | 1 |
| **4** | 35.0 | 0 | 0 | 8.0500 | 0 | 1 | 1 | 0 | 1 |

Create a new k-nearest neighbors object and fit it on the entire training data ( X_train_kaggle ).

In [15]:
```
knn_kaggle = KNeighborsClassifier(n_neighbors=3)
knn_kaggle.fit(X_train_kaggle,y_train_kaggle)
```

Out[15]:
```
KNeighborsClassifier(n_neighbors=3)
```

Now use our fitted classifier to make predictions on the test data ( test2 ), and store it in the variable y_pred .

In [17]:
```
y_pred = knn_kaggle.predict(test2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\S1406~1.ALI\AppData\Local\Temp/ipykernel_18680/290278459.py in <mod
ule>
----> 1 y_pred = knn_kaggle.predict(test2)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py in
predict(self, X)
    193             Class labels for each data sample.
    194         """
--> 195         X = check_array(X, accept_sparse='csr')
    196 
    197         neigh_dist, neigh_ind = self.kneighbors(X)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*a
rgs, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_arra
y(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensur
e_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    718 
    719         if force_all_finite:
--> 720             _assert_all_finite(array,
    721                                allow_nan=force_all_finite == 'allow-nan')
    722 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all
_finite(X, allow_nan, msg_dtype)
    101                 not allow_nan and not np.isfinite(X).all()):
    102             type_err = 'infinity' if allow_nan else 'NaN, infinity'
--> 103             raise ValueError(
    104                     msg_err.format
    105                     (type_err,

ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

The error message says something about NaN. Could there be missing data (NaN) is a different column in the test data? Use describe to see if this is the case.

In [18]:
```
test2.describe()
```

Out[18]:

| | Age | SibSp | Parch | Fare | Pclass_2 | Pclass_3 | Sex_male | Embarke |
|---|---|---|---|---|---|---|---|---|
| **count** | 418.000000 | 418.000000 | 418.000000 | 417.000000 | 418.000000 | 418.000000 | 418.000000 | 418.00 |
| **mean** | 29.599282 | 0.447368 | 0.392344 | 35.627188 | 0.222488 | 0.521531 | 0.636364 | 0.11 |
| **std** | 12.703770 | 0.896760 | 0.981429 | 55.907576 | 0.416416 | 0.500135 | 0.481622 | 0.31 |
| **min** | 0.170000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **25%** | 23.000000 | 0.000000 | 0.000000 | 7.895800 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **50%** | 27.000000 | 0.000000 | 0.000000 | 14.454200 | 0.000000 | 1.000000 | 1.000000 | 0.00 |
| **75%** | 35.750000 | 1.000000 | 0.000000 | 31.500000 | 0.000000 | 1.000000 | 1.000000 | 0.00 |
| **max** | 76.000000 | 8.000000 | 9.000000 | 512.329200 | 1.000000 | 1.000000 | 1.000000 | 1.00 |

The fare column is missing one value. Fill it in with the median fare.

In [19]:
```
test2["Fare"] = test2["Fare"].fillna(test2["Fare"].median())
```

Now try making the prediction again.

In [20]:
```
y_pred = knn_kaggle.predict(test2)
```

In [21]:
```
y_pred
```

Out[21]:
```
array([0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1],
      dtype=int64)
```

Finally, we want to write our prediction to a file, along with the passenger ID.

In [22]:
```
df = pd.DataFrame(data = {"PassengerId":passengerId, "Survived":y_pred})
```

In [23]:
```
df.to_csv("test1.csv",index = 0)
```

In [ ]:
```

```