Lez12_Titanic

November 23, 2022

[1]:
```
# Imports
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import sklearn.metrics as metrics

# Import data of the titanic passengers
data_titanic = pd.read_csv("titanic_data.csv")

print(data_titanic)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         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
..                                                   …     …   …    …
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
..     …               …       …   …      …
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]
```

[2]:
```
# Not all data is important for the training. The PassengerId, the name, the number of siblings, the parch, the ticket number
# and the cabin number are not important and therefore they will be dropped
data_titanic = data_titanic.drop(data_titanic.columns[[0, 3, 6, 7, 8, 10]], axis = 1)
print(data_titanic)
```

```
     Survived  Pclass     Sex   Age     Fare Embarked
0           0       3    male  22.0   7.2500        S
1           1       1  female  38.0  71.2833        C
2           1       3  female  26.0   7.9250        S
3           1       1  female  35.0  53.1000        S
4           0       3    male  35.0   8.0500        S
..        …     …      …   …      …      …
886         0       2    male  27.0  13.0000        S
887         1       1  female  19.0  30.0000        S
888         0       3  female   NaN  23.4500        S
889         1       1    male  26.0  30.0000        C
890         0       3    male  32.0   7.7500        Q

[891 rows x 6 columns]
```

[3]:
```
# Check for missing values in the data set
data_titanic.isnull().sum() # Sums all the empty entries of a columns
# As can be seen, the age and the Embarked columns have missing values
```

[3]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
```

```
dtype: int64
```

[4]:
```
# Fill the gaps:
# For the age: Compute the average of ages of all passengers on board and take␣
 ↪it as value
ages = data_titanic["Age"]
age_average = round(ages.mean(axis = 0,  skipna = True))
print('Age average: ', age_average)

def set_age(Age):
    age = Age
    if pd.isnull(age):
        return age_average
    else:
        return age

data_titanic['Age'] = data_titanic['Age'].apply(set_age)

# For the embarked: Take the most occuring value, available options: S, C and Q
Embarked = data_titanic["Embarked"]
count_S = 0
count_C = 0
count_Q = 0

# Count embarks
for i in range(data_titanic.shape[0]):
    if (Embarked[i] == 'S'):
        count_S += 1
    if (Embarked[i] == 'C'):
        count_C += 1
    if (Embarked[i] == 'Q'):
        count_Q += 1

if ((count_S >= count_C) and (count_S >= count_Q)):
    common_embarked = 'S'
if ((count_C > count_S) and (count_C >= count_Q)):
    common_embarked = 'C'
if ((count_Q >= count_S) and (count_Q >= count_C)):
    common_embarked = 'Q'

def set_Embarked(Embarked):
    embarked = Embarked
    if pd.isnull(embarked):
        return common_embarked
    else:
        return embarked
```

```
data_titanic['Embarked'] = data_titanic['Embarked'].apply(set_Embarked)

# Check replacement
data_titanic.isnull().sum()
```

```
Age average:  30
```

[4]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    0
dtype: int64
```

[5]:
```
# Gender and Embarked have to be replaced by values: Set male = 0, female = 1 and S = 0, C = 1, Q = 2
replace_gender = {data_titanic.columns[2]:     {"male": 0, "female": 1}}
replace_embarked = {data_titanic.columns[5]:     {"S": 0, "C": 1, "Q": 2}}
data_titanic = data_titanic.replace(replace_gender)
data_titanic = data_titanic.replace(replace_embarked)

# Check replacement
print(data_titanic)
```

```
     Survived  Pclass  Sex   Age      Fare  Embarked
0           0       3    0  22.0    7.2500         0
1           1       1    1  38.0   71.2833         1
2           1       3    1  26.0    7.9250         0
3           1       1    1  35.0   53.1000         0
4           0       3    0  35.0    8.0500         0
..        ...     ...  ...   ...       ...       ...
886         0       2    0  27.0   13.0000         0
887         1       1    1  19.0   30.0000         0
888         0       3    1  30.0   23.4500         0
889         1       1    0  26.0   30.0000         1
890         0       3    0  32.0    7.7500         2

[891 rows x 6 columns]
```

[8]:
```
# Run a KNN prediction

# Split data into train and evaluation subsets, proportion 1:4
input_data = data_titanic.drop(data_titanic.columns[[0]], axis = 1) # Input has to be without "Survived" statement
target_data = data_titanic['Survived'] # KNN algorithm has to predict survied statement
```

```
input_train, input_evaluate, target_train, target_evaluate =␣
 ↪train_test_split(input_data , target_data, test_size = 0.25)

# Test for k = 3, 4 and 5
# Initalise
KNN_3 = KNeighborsClassifier(n_neighbors=3)
KNN_4 = KNeighborsClassifier(n_neighbors=4)
KNN_5 = KNeighborsClassifier(n_neighbors=5)

#Train
KNN_3.fit(input_train, target_train)
KNN_4.fit(input_train, target_train)
KNN_5.fit(input_train, target_train)

#Perform evaluation
target_test_3 = KNN_3.predict(input_evaluate)
target_test_4 = KNN_4.predict(input_evaluate)
target_test_5 = KNN_5.predict(input_evaluate)

# Investigate accuracy, recall and precision
accuracy_3 = metrics.accuracy_score(target_test_3, target_evaluate)
accuracy_4 = metrics.accuracy_score(target_test_4, target_evaluate)
accuracy_5 = metrics.accuracy_score(target_test_5, target_evaluate)

recall_3 = metrics.recall_score(target_test_3, target_evaluate)
recall_4 = metrics.recall_score(target_test_4, target_evaluate)
recall_5 = metrics.recall_score(target_test_5, target_evaluate)

precision_3 = metrics.precision_score(target_test_3, target_evaluate)
precision_4 = metrics.precision_score(target_test_4, target_evaluate)
precision_5 = metrics.precision_score(target_test_5, target_evaluate)

print("KNN_3: Accuracy: ", accuracy_3, ", Recall: ", recall_3, ", Precision: ",␣
 ↪precision_3)
print("KNN_4: Accuracy: ", accuracy_4, ", Recall: ", recall_4, ", Precision: ",␣
 ↪precision_4)
print("KNN_5: Accuracy: ", accuracy_5, ", Recall: ", recall_5, ", Precision: ",␣
 ↪precision_5)
```

```
KNN_3: Accuracy:  0.672645739910314 , Recall:  0.6486486486486487 , Precision:
0.5052631578947369
KNN_4: Accuracy:  0.6591928251121076 , Recall:  0.7021276595744681 , Precision:
0.5473684210526316
KNN_5: Accuracy:  0.6816143497757847 , Recall:  0.6666666666666666 , Precision:
0.5052631578947369
```

[ ]: