Data Analysis - Lab 5

ISEP – November 19th, 2019

Instructions: Prepare a report and send it to the deposit box on moodle

Libraries

This lab may require the following libraries : pandas, iPython, pydot, matplotlib, numpy, scipy et sklearn.

A Titanic Dataset

A.1 Preliminary analysis

1. Open the files "titanic_train.csv" and "titanic_test.csv"
   - Why do we have 2 different files ? Are they identical ?
   - Describe the different attributes of the titanic_train file using the function *head()* and the property *dtype*.
   - Using the functions *isnull()* and *sum()* what can you say about the missing data for each attribute ?
2. Using the titanic_train file, display the following pieces of information:
   - Display a histogram of the age for the people on board. Be careful on how you handle the missing values.
   - What is the percentage of survivors and people that died ?
   - What is the percentage of people in each class (1st, 2nd and 3rd) ? Display a pie plot.
   - Display the percentage of survivors for men and women.
   - Knowing that children were considered adults at 18yo, what is the percentage of survivors for children and adults ? Explain how you handled the missing data.
   - Display the survival rates for all possible combinations adult/children, men/women.
   - Display the survival rates in 1st, 2nd and 3rd class. Once again, explain how you handled missing data.
3. Based on your previous answers, what can you say about the policy "women and children first" on board the Titanic ?

4. Display a correlation matrix of all the attributes and comment on any interesting thing that may show up.

5. In both training and test sets, create a "Fare2" attribute whose value is 1 if the ticket price was below 10, 2 below 20, 3 below 30 and 4 otherwise. Using the right tools, determine the strength of the connection between your new "Fare2" attribute and the class the passengers traveled into. Then display the survival rates for each value of "Fare2". Comment and conclude on the influence of the ticket price on survival chances.

A.2 Naive Bayes Prediction

1. If you did not do so already in the preliminary analysis section, add a binary attribute "Child" whose value is "1" for passenger below 18yo and 0 for adults. Do so in both training and test sets. Don't forget to remind the policy of your choice for missing data.

2. Use the Nave Bayes classifier to predict survival rates using the sex of the passengers and whether or not they were adults. Confirm your results in both training and test sets, and explain the different metrics proposed by the function *classification_report*. An example of code you can adapt and use is proposed below.

```
# needed imports
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

#classifier choice
gnbModel=GaussianNB()
#choice of the training set, considered attributes and variable to predict
gnbModel.fit(df_train[['Child','Sex']], df_train['Survived'])

#expected results are stored in a separate vector
expected =df_train['Survived']

#predictions on the training set
predicted = gnbModel.predict(df_train[['Child','Sex']])
#displaying relevant metrics
print(metrics.classification_report(expected, predicted))

#same when applying the model to the test set
expected =df_test['Survived']
predicted = gnbModel.predict(df_test[['Child','Sex']])
print(metrics.classification_report(expected, predicted))
```

3. Try again a Naive Bayes classification with the sex, child or not and the Fare2 attribute. Comment on the results.

A.3 Decision trees Prediction

1. Train a decision tree to predict the survival chances of all passengers. You may use any attributes of your choice, including the ones you created (Child and Fare2). You can try several combinations. Comments on your results.

Remark : All classifiers in sklearn work the same way and have the same main prototype functions. The function *fit()* is used to train the algorithm on the training set using chosen attributes, and to set the algorithm parameters if any.

The function *predict()* is used to applied the trained model on new data to predict their classes. The code below gives an example of decision trees and how to display them.

```
# needed imports
import pydotplus
import collections
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
from sklearn import tree

clf =tree.DecisionTreeClassifier()
clf=clf.fit(df_train[['Pclass','Fare2','Child','Sex']],df_train_survived)
data_feature_names = [ 'Pclass','Fare2','Child','Sex']

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,feature_names=data_feature_names,
    filled=True,rounded=True,class_names=True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```