# exam-2021-questions

February 23, 2021

**30E03000 - Data Science for Business I (2021)**

## 1 First exam - 23.02.2021

#### 1.0.1 Case: Healthcare analytics (60pt)

Assume that you are consulting a hospital that aims at improving patient outcomes, efficiency and reducing costs. The data set that the hospital has provided to you contains profiles of several patients including their demographic details along with a set of medical measurements taken during their recent visit (M1 - M9). The objective is to predict which patients should be called for further clinical testing. The testing procedure is costly and the hospital would like to avoid calling healthy patients for an additional test.

The data set is provided in the file **healthcare.csv**. Relevant background information on the variables is provided in the file **healthcare-variables.txt**.

When completing the following steps, you should comment on your findings by adding markdown-boxes along with code-boxes. **Add markdown and code boxes as many as you need**. You can also add subtitles to improve readability of your answer.

**Make sure to always answer the questions verbally!!** Providing only code will not show your interpretation.

#### 1.0.2 Submission

Download your filled-out notebook in **.ipynb format** and upload it to the MyCourses exam submission box. Name your file studentNum-lastname-firstname-exam.ipynb.

### 1.1 Task 1 (5pt):

Load data set from “healthcare.csv” file. Perform exploratory data analysis to understand your data better. Check for missing (or otherwise weird) values and variable types. Transform variables when needed (e.g., dummies). You can also drop variables that you don’t want to include, but justify your decision! Use at least one data visualization technique. Remember to discuss your choices in a markdown box.

[17]:

```
import pandas as pd

#add all necessary libraries here
import pandas as pd

#add all necessary libraries here
import numpy as np #scientific computing
import pandas as pd #data management
import itertools

#matplotlib for plotting
import matplotlib.pyplot as plt
from matplotlib import gridspec
import matplotlib.ticker as mtick #for percentage ticks

#sklearn for modeling
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier #Decision Tree algorithm
from sklearn.model_selection import train_test_split #Data split function
from sklearn.preprocessing import LabelEncoder #OneHotEncoding
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc

#Decision tree plot
import pydotplus
from IPython.display import Image

from matplotlib import gridspec
import matplotlib.ticker as mtick #for percentage ticks
import scikitplot as skplt


from sklearn.model_selection import train_test_split #Data split function
from imblearn.over_sampling import SMOTE
from sklearn.decomposition import PCA

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc

from collections import Counter
```

Read the file

[32]:

```
# read file into a dataframe
healthcare = pd.read_csv("healthcare.csv")
print(healthcare.head())
```

```
   Unnamed: 0     sex  age             education                employment  \
0           1    MALE   52    VOCATIONAL/GRAMMAR      LONG-TERM SICK/DISABLED
1           2  FEMALE   52             SECONDARY    EMPLOYED IN PUBLIC SECTOR
2           3  FEMALE   78  PRIMARY/NO EDUCATION  OTHER ECONOMICALLY INACTIVE
3           4    MALE   38    VOCATIONAL/GRAMMAR   EMPLOYED IN PRIVATE SECTOR
4           5    MALE   61  PRIMARY/NO EDUCATION                      RETIRED

   marital  unempdur  income abroad  depress  …     m1    m2    m3    m4  \
0  MARRIED         0   560.0     NO     10.0  … -0.40  2.16 -0.93  1.24
1  WIDOWED        -8  1600.0     NO      9.0  …  0.92  1.07  0.06  1.19
2  MARRIED        -8     NaN     NO     10.0  …  1.15  0.71  2.02 -0.18
3  MARRIED         0  2200.0     NO      1.0  …  1.08  0.34  1.41 -0.99
4  MARRIED        24     NaN     NO     14.0  …  0.14  0.21 -0.14  1.62

     m5    m6    m7    m8    m9    y
0  0.74  0.95  0.16 -0.20  3.06  1.0
1  0.03  0.79  0.28 -0.82  0.77  0.0
2  0.92  0.39  0.22  0.85  1.08  0.0
3  0.05  0.07  0.35  1.57 -1.11  0.0
4 -0.33  0.62  0.12 -0.19  1.91  0.0

[5 rows x 25 columns]
```

[58]:

```
print("Dimensions of original data:", data.shape)
```

```
Dimensions of original data: (1500, 33)
```

[59]:

```
print(list(healthcare.columns))
```

```
['Unnamed: 0', 'sex', 'age', 'education', 'employment', 'marital', 'unempdur',
'income', 'abroad', 'depress', 'trust', 'sport', 'smoke', 'weight', 'bmi', 'm1',
'm2', 'm3', 'm4', 'm5', 'm6', 'm7', 'm8', 'm9', 'y']
```

[60]:

```
## I only keep these variables to do with the model: age, sex, employment,␣
 ↪income, bmi, 1-M9 to predict the need for clinical testing
```

[61]:

```
modified = healthcare.drop(["Unnamed: 0", "education", "marital", "unempdur",␣
 ↪"abroad", "depress", "trust", "sport", "smoke", "weight"], axis=1)
print(modified.columns)
```

```
Index(['sex', 'age', 'employment', 'income', 'bmi', 'm1', 'm2', 'm3', 'm4',
       'm5', 'm6', 'm7', 'm8', 'm9', 'y'],
      dtype='object')
```

[62]:
```
modified.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 15 columns):
sex           1500 non-null object
age           1500 non-null int64
employment    1491 non-null object
income        1082 non-null float64
bmi           1483 non-null float64
m1            1500 non-null float64
m2            1500 non-null float64
m3            1500 non-null float64
m4            1500 non-null float64
m5            1500 non-null float64
m6            1500 non-null float64
m7            1500 non-null float64
m8            1500 non-null float64
m9            1500 non-null float64
y             1498 non-null float64
dtypes: float64(12), int64(1), object(2)
memory usage: 175.9+ KB
```

[63]:
```
## I also drop catagory 'OTHER ECONOMICALLY INACTIVE' and 'UNEMPLOYED' because␣
↪these two category will not impact the result of clinical testing
```

[64]:
```
con1 = modified.employment != "UNEMPLOYED"
con2 = modified.employment != "OTHER ECONOMICALLY INACTIVE"
modified[con1 & con2]
```

[64]:
```
         sex  age                   employment  income        bmi    m1    m2  \
0       MALE   52     LONG-TERM SICK/DISABLED    560.0  22.160665 -0.40  2.16
1     FEMALE   52   EMPLOYED IN PUBLIC SECTOR   1600.0  26.233556  0.92  1.07
3       MALE   38  EMPLOYED IN PRIVATE SECTOR   2200.0  26.643599  1.08  0.34
4       MALE   61                      RETIRED     NaN  20.761246  0.14  0.21
5       MALE   77                      RETIRED  1330.0  27.434842 -1.11  0.30
…        …   …                           …     …        …   …   …
1493  FEMALE   55                SELF-EMPLOYED     NaN  27.639801  0.63  0.43
1494  FEMALE   51     LONG-TERM SICK/DISABLED      NaN  33.593750 -0.74 -1.10
1495    MALE   24            PUPIL OR STUDENT   1600.0  25.432686  0.40  0.73
1496  FEMALE   52  EMPLOYED IN PRIVATE SECTOR   2000.0  30.811246  1.27  1.16
1498  FEMALE   53   EMPLOYED IN PUBLIC SECTOR   3000.0  20.820940 -0.66  1.30

        m3    m4    m5    m6    m7    m8    m9    y
0    -0.93  1.24  0.74  0.95  0.16 -0.20  3.06  1.0
1     0.06  1.19  0.03  0.79  0.28 -0.82  0.77  0.0
3     1.41 -0.99  0.05  0.07  0.35  1.57 -1.11  0.0
```

```
4     -0.14  1.62 -0.33  0.62  0.12 -0.19  1.91  0.0
5     -0.03 -0.45 -0.08  0.52  0.98  0.31 -0.55  0.0
…     …    …    …    …    …    …    …    …
1493  0.77 -0.15  0.79  0.46  0.95 -0.78 -0.54  1.0
1494  0.52 -0.23 -0.54  0.32  0.35  0.89  0.40  0.0
1495 -0.09 -1.24  0.10  0.31  0.73 -0.51  2.03  0.0
1496 -0.47 -1.24  0.46  0.43  0.71 -0.75  0.20  0.0
1498  0.61  0.41  0.17  0.25  0.67 -1.26  1.06  0.0

[1237 rows x 15 columns]
```

[65]:

```
modified = modified.dropna(axis=0, how="any")
modified.info()
print(modified.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1061 entries, 0 to 1498
Data columns (total 15 columns):
sex           1061 non-null object
age           1061 non-null int64
employment    1061 non-null object
income        1061 non-null float64
bmi           1061 non-null float64
m1            1061 non-null float64
m2            1061 non-null float64
m3            1061 non-null float64
m4            1061 non-null float64
m5            1061 non-null float64
m6            1061 non-null float64
m7            1061 non-null float64
m8            1061 non-null float64
m9            1061 non-null float64
y             1061 non-null float64
dtypes: float64(12), int64(1), object(2)
memory usage: 132.6+ KB
      sex  age                  employment  income        bmi    m1    m2  \
0    MALE   52     LONG-TERM SICK/DISABLED   560.0  22.160665 -0.40  2.16
1  FEMALE   52    EMPLOYED IN PUBLIC SECTOR  1600.0  26.233556  0.92  1.07
3    MALE   38   EMPLOYED IN PRIVATE SECTOR  2200.0  26.643599  1.08  0.34
5    MALE   77                      RETIRED  1330.0  27.434842 -1.11  0.30
6    MALE   55                      RETIRED  1600.0  27.636054  1.49  2.31

     m3    m4    m5    m6    m7    m8    m9    y
0 -0.93  1.24  0.74  0.95  0.16 -0.20  3.06  1.0
1  0.06  1.19  0.03  0.79  0.28 -0.82  0.77  0.0
3  1.41 -0.99  0.05  0.07  0.35  1.57 -1.11  0.0
5 -0.03 -0.45 -0.08  0.52  0.98  0.31 -0.55  0.0
```

```
6  0.63 -0.15 -0.29  0.23  0.65  0.52 -1.03  1.0
```

[69]:

```python
new = pd.get_dummies(modified[['sex', 'employment']]).head()
modified = pd.concat([new, modified], axis=1)
print(modified.head())
modified.info()
```

```
   sex_FEMALE  sex_MALE  employment_EMPLOYED IN PRIVATE SECTOR  \
0         0.0       1.0                                    0.0   
1         1.0       0.0                                    0.0   
3         0.0       1.0                                    1.0   
5         0.0       1.0                                    0.0   
6         0.0       1.0                                    0.0   

   employment_EMPLOYED IN PUBLIC SECTOR  employment_FARMER  \
0                                   0.0                0.0   
1                                   1.0                0.0   
3                                   0.0                0.0   
5                                   0.0                0.0   
6                                   0.0                0.0   

   employment_LONG-TERM SICK/DISABLED  employment_OTHER ECONOMICALLY INACTIVE  \
0                                 1.0                                     0.0   
1                                 0.0                                     0.0   
3                                 0.0                                     0.0   
5                                 0.0                                     0.0   
6                                 0.0                                     0.0   

   employment_PUPIL OR STUDENT  employment_RETIRED  employment_SELF-EMPLOYED  \
0                          0.0                 0.0                       0.0   
1                          0.0                 0.0                       0.0   
3                          0.0                 0.0                       0.0   
5                          0.0                 1.0                       0.0   
6                          0.0                 1.0                       0.0   

   employment_UNEMPLOYED     sex  age                  employment  income  \
0                    0.0    MALE   52     LONG-TERM SICK/DISABLED   560.0   
1                    0.0  FEMALE   52   EMPLOYED IN PUBLIC SECTOR  1600.0   
3                    0.0    MALE   38  EMPLOYED IN PRIVATE SECTOR  2200.0   
5                    0.0    MALE   77                     RETIRED  1330.0   
6                    0.0    MALE   55                     RETIRED  1600.0   

         bmi    m1    m2    m3    m4    m5    m6    m7    m8    m9    y  
0  22.160665 -0.40  2.16 -0.93  1.24  0.74  0.95  0.16 -0.20  3.06  1.0  
1  26.233556  0.92  1.07  0.06  1.19  0.03  0.79  0.28 -0.82  0.77  0.0  
3  26.643599  1.08  0.34  1.41 -0.99  0.05  0.07  0.35  1.57 -1.11  0.0  
5  27.434842 -1.11  0.30 -0.03 -0.45 -0.08  0.52  0.98  0.31 -0.55  0.0  
6  27.636054  1.49  2.31  0.63 -0.15 -0.29  0.23  0.65  0.52 -1.03  1.0  
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1061 entries, 0 to 1498
Data columns (total 26 columns):
sex_FEMALE                                5 non-null float64
sex_MALE                                  5 non-null float64
employment_EMPLOYED IN PRIVATE SECTOR     5 non-null float64
employment_EMPLOYED IN PUBLIC SECTOR      5 non-null float64
employment_FARMER                         5 non-null float64
employment_LONG-TERM SICK/DISABLED        5 non-null float64
employment_OTHER ECONOMICALLY INACTIVE    5 non-null float64
employment_PUPIL OR STUDENT               5 non-null float64
employment_RETIRED                        5 non-null float64
employment_SELF-EMPLOYED                  5 non-null float64
employment_UNEMPLOYED                     5 non-null float64
sex                                       1061 non-null object
age                                       1061 non-null int64
employment                                1061 non-null object
income                                    1061 non-null float64
bmi                                       1061 non-null float64
m1                                        1061 non-null float64
m2                                        1061 non-null float64
m3                                        1061 non-null float64
m4                                        1061 non-null float64
m5                                        1061 non-null float64
m6                                        1061 non-null float64
m7                                        1061 non-null float64
m8                                        1061 non-null float64
m9                                        1061 non-null float64
y                                         1061 non-null float64
dtypes: float64(23), int64(1), object(2)
memory usage: 223.8+ KB
```

```
[70]: data = modified.drop(["sex", "employment"], axis =1)
      data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1061 entries, 0 to 1498
Data columns (total 24 columns):
sex_FEMALE                                5 non-null float64
sex_MALE                                  5 non-null float64
employment_EMPLOYED IN PRIVATE SECTOR     5 non-null float64
employment_EMPLOYED IN PUBLIC SECTOR      5 non-null float64
employment_FARMER                         5 non-null float64
employment_LONG-TERM SICK/DISABLED        5 non-null float64
employment_OTHER ECONOMICALLY INACTIVE    5 non-null float64
employment_PUPIL OR STUDENT               5 non-null float64
employment_RETIRED                        5 non-null float64
employment_SELF-EMPLOYED                  5 non-null float64
```

```
employment_UNEMPLOYED                       5 non-null float64
age                                         1061 non-null int64
income                                      1061 non-null float64
bmi                                         1061 non-null float64
m1                                          1061 non-null float64
m2                                          1061 non-null float64
m3                                          1061 non-null float64
m4                                          1061 non-null float64
m5                                          1061 non-null float64
m6                                          1061 non-null float64
m7                                          1061 non-null float64
m8                                          1061 non-null float64
m9                                          1061 non-null float64
y                                           1061 non-null float64
dtypes: float64(23), int64(1)
memory usage: 207.2 KB
```

[71]:

```
pd.set_option('display.max_columns', None)
```

[72]:

```
data.corr().style
```

[72]:

```
<pandas.io.formats.style.Styler at 0x7f7a245b0630>
```

[81]:

```
y = data['y'].values #define target variable
X = data.loc[:, data.columns != 'y'] #define feature matrix
```

---

## 1.2 Task 2 (5pt):

Split the data into training (70%) and testing (30%) data sets. Check outcome distributions on training and test datasets.

[82]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
 ↪random_state = 1234)
```

[83]:

```
train_dist = y_train.value_counts() / len(y_train) #normalize absolute count␣
 ↪values for plotting
test_dist = y_test.value_counts() / len(y_test)
data_dist = y.value_counts() / len(y)

fig, ax = plt.subplots()

ax.barh(['Test','Train','Data'], [test_dist[0], train_dist[0], data_dist[0]],␣
 ↪color='#1f77b4', label='0 (no)')
```

```
ax.barh(['Test','Train','Data'], [test_dist[1], train_dist[1], data_dist[1]],␣
 ↪left=[test_dist[0], train_dist[0], data_dist[0]], color='#ff7f0e', label='1␣
 ↪(yes)')
ax.set_title('Split visualization')
ax.legend(loc='upper left')
plt.xlabel('Proportion')
plt.ylabel('Partition')

#plot bar values
for part, a, b in zip(['Test', 'Train','Data'], [test_dist[0], train_dist[0],␣
 ↪data_dist[0]], [test_dist[1], train_dist[1], data_dist[1]]):
    plt.text(a/2, part, str(np.round(a, 2)))
    plt.text(b/2+a, part, str(np.round(b, 2)));
```

```
      ␣
↪---------------------------------------------------------------------------

       AttributeError                            Traceback (most recent call␣
↪last)

       <ipython-input-83-e0c838cad8eb> in <module>
   ----> 1 train_dist = y_train.value_counts() / len(y_train) #normalize␣
↪absolute count values for plotting
         2 test_dist = y_test.value_counts() / len(y_test)
         3 data_dist = y.value_counts() / len(y)
         4
         5 fig, ax = plt.subplots()


       AttributeError: 'numpy.ndarray' object has no attribute 'value_counts'
```

[84]:

```
#Define Decision tree classifier with some default parameters
clf = tree.DecisionTreeClassifier(criterion = "gini", random_state = 100,
                               max_depth=3, min_samples_leaf=3)

#Fit the training data
clf.fit(X_train,y_train) #what do we need here?
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)
```

```
    <ipython-input-84-f2ee423b2da9> in <module>
      4
      5 #Fit the training data
----> 6 clf.fit(X_train,y_train) #what do we need here?


    /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪fit(self, X, y, sample_weight, check_input, X_idx_sorted)
    875             sample_weight=sample_weight,
    876             check_input=check_input,
--> 877             X_idx_sorted=X_idx_sorted)
    878         return self
    879


    /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪fit(self, X, y, sample_weight, check_input, X_idx_sorted)
    147
    148         if check_input:
--> 149             X = check_array(X, dtype=DTYPE, accept_sparse="csc")
    150             y = check_array(y, ensure_2d=False, dtype=None)
    151             if issparse(X):


    /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
↪ensure_min_features, warn_on_dtype, estimator)
    576         if force_all_finite:
    577             _assert_all_finite(array,
--> 578                                allow_nan=force_all_finite ==␣
↪'allow-nan')
    579
    580     if ensure_min_samples > 0:


    /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪_assert_all_finite(X, allow_nan, msg_dtype)
     58                     msg_err.format
     59                     (type_err,
---> 60                      msg_dtype if msg_dtype is not None else X.dtype)
     61             )
     62     # for object dtype data, we only check for NaNs (GH-13254)


    ValueError: Input contains NaN, infinity or a value too large for␣
↪dtype('float32').
```

[85]:

```
#Use classifier to predict labels
y_pred = clf.predict(X_test) #what do we need here?
```

```
            ␣
↪---------------------------------------------------------------------------

      NotFittedError                            Traceback (most recent call␣
↪last)

      <ipython-input-85-56cee0ea7a3f> in <module>
        1 #Use classifier to predict labels
  ----> 2 y_pred = clf.predict(X_test) #what do we need here?


      /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪predict(self, X, check_input)
      416             The predicted classes, or the predict values.
      417         """
  --> 418         check_is_fitted(self)
      419         X = self._validate_X_predict(X, check_input)
      420         proba = self.tree_.predict(X)


      /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
      965
      966     if not attrs:
  --> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
      968
      969


      NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[86]:

```
#probabilities
y_pred_probs = clf.predict_proba(X_test)
```

```
            ␣
↪---------------------------------------------------------------------------
```

```
    NotFittedError                            Traceback (most recent call␣
↪last)

    <ipython-input-86-46af445fb991> in <module>
      1 #probabilities
----> 2 y_pred_probs = clf.predict_proba(X_test)


    /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪predict_proba(self, X, check_input)
    902             classes corresponds to that in the attribute :term:
↪`classes_`.
    903         """
--> 904         check_is_fitted(self)
    905         X = self._validate_X_predict(X, check_input)
    906         proba = self.tree_.predict(X)


    /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
    965
    966     if not attrs:
--> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
    968
    969


    NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[87]:

```
'''
The graphviz library is used to visualize the tree.
'''

#Decision tree plot
import pydotplus
from IPython.display import Image

# Create DOT data
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X_train.columns,
                                class_names=['need for testing', 'no need for␣
 ↪testing'], filled=True) #or use y_train.unique()

# Draw graph
```

```
graph = pydotplus.graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())

# Create PNG
#graph.write_png("clf.png") #uncomment this line to save the plot as a .png file
```

```
     ␣
↪---------------------------------------------------------------------------

       NotFittedError                            Traceback (most recent call␣
↪last)

       <ipython-input-87-1d41bfbe3054> in <module>
        10 dot_data = tree.export_graphviz(clf, out_file=None,
        11                                 feature_names=X_train.columns,
   ---> 12                                 class_names=['need for testing', 'no␣
↪need for testing'], filled=True) #or use y_train.unique()
        13
        14 # Draw graph


       /opt/conda/lib/python3.7/site-packages/sklearn/tree/_export.py in␣
↪export_graphviz(decision_tree, out_file, max_depth, feature_names,␣
↪class_names, label, filled, leaves_parallel, impurity, node_ids, proportion,␣
↪rotate, rounded, special_characters, precision)
       743     """
       744
   --> 745     check_is_fitted(decision_tree)
       746     own_file = False
       747     return_string = False


       /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
       965
       966     if not attrs:
   --> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
       968
       969


       NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[ ]:

```
```

[ ]:

```
```

---

### 1.3 Task 3 (15pt):

Build a simple decision tree model. You can adjust the complexity and parameters of the model to ensure interpretability.

[88]:

```
#Define Decision tree classifier with some default parameters
clf = tree.DecisionTreeClassifier(criterion = "gini", random_state = 100,
                                  max_depth=3, min_samples_leaf=3)

#Fit the training data
clf.fit(X_train,y_train) #what do we need here?
```

```
       ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-88-f2ee423b2da9> in <module>
         4
         5 #Fit the training data
   ----> 6 clf.fit(X_train,y_train) #what do we need here?


       /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪fit(self, X, y, sample_weight, check_input, X_idx_sorted)
       875             sample_weight=sample_weight,
       876             check_input=check_input,
   --> 877             X_idx_sorted=X_idx_sorted)
       878         return self
       879


       /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪fit(self, X, y, sample_weight, check_input, X_idx_sorted)
       147
       148         if check_input:
   --> 149             X = check_array(X, dtype=DTYPE, accept_sparse="csc")
```

```
        150             y = check_array(y, ensure_2d=False, dtype=None)
        151             if issparse(X):


        /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
↪ensure_min_features, warn_on_dtype, estimator)
        576         if force_all_finite:
        577             _assert_all_finite(array,
    --> 578                                allow_nan=force_all_finite ==␣
↪'allow-nan')
        579
        580     if ensure_min_samples > 0:


        /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪_assert_all_finite(X, allow_nan, msg_dtype)
         58                     msg_err.format
         59                     (type_err,
    ---> 60                      msg_dtype if msg_dtype is not None else X.dtype)
         61             )
         62     # for object dtype data, we only check for NaNs (GH-13254)


        ValueError: Input contains NaN, infinity or a value too large for␣
↪dtype('float32').
```

[89]:
```
#Use classifier to predict labels
y_pred = clf.predict(X_test) #what do we need here?
```

```
      ␣
↪---------------------------------------------------------------------------

        NotFittedError                            Traceback (most recent call␣
↪last)

        <ipython-input-89-56cee0ea7a3f> in <module>
          1 #Use classifier to predict labels
    ----> 2 y_pred = clf.predict(X_test) #what do we need here?


        /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪predict(self, X, check_input)
        416             The predicted classes, or the predict values.
        417         """
    --> 418         check_is_fitted(self)
        419         X = self._validate_X_predict(X, check_input)
        420         proba = self.tree_.predict(X)


        /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
        965
        966     if not attrs:
    --> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
        968
        969


        NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[90]:

```
'''
The graphviz library is used to visualize the tree.
'''

#Decision tree plot
import pydotplus
from IPython.display import Image

# Create DOT data
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X_train.columns,
                                class_names=['NO RESPONSE', 'RESPONSE'],␣
 ↪filled=True) #or use y_train.unique()

# Draw graph
graph = pydotplus.graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())

# Create PNG
#graph.write_png("clf.png") #uncomment this line to save the plot as a .png file
```

```
      ␣
↪---------------------------------------------------------------------------
```

```
       NotFittedError                            Traceback (most recent call␣
↪last)

       <ipython-input-90-df6fcba86956> in <module>
        10 dot_data = tree.export_graphviz(clf, out_file=None,
        11                                 feature_names=X_train.columns,
   ---> 12                                 class_names=['NO RESPONSE',␣
↪'RESPONSE'], filled=True) #or use y_train.unique()
        13
        14 # Draw graph


       /opt/conda/lib/python3.7/site-packages/sklearn/tree/_export.py in␣
↪export_graphviz(decision_tree, out_file, max_depth, feature_names,␣
↪class_names, label, filled, leaves_parallel, impurity, node_ids, proportion,␣
↪rotate, rounded, special_characters, precision)
       743     """
       744
   --> 745     check_is_fitted(decision_tree)
       746     own_file = False
       747     return_string = False


       /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
       965
       966     if not attrs:
   --> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
       968
       969


       NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[91]:

```
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]
feature_order = np.array([X.columns.values])
i = np.argsort(importances)[::-1]
feature_order = feature_order[:,i]
```

```
      ␣
↪---------------------------------------------------------------------------
```

```
        NotFittedError                            Traceback (most recent call␣
↪last)

        <ipython-input-91-aff7d3daae72> in <module>
    ----> 1 importances = clf.feature_importances_
          2 indices = np.argsort(importances)[::-1]
          3 feature_order = np.array([X.columns.values])
          4 i = np.argsort(importances)[::-1]
          5 feature_order = feature_order[:,i]


        /opt/conda/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
↪feature_importances_(self)
        574             (Gini importance).
        575         """
    --> 576         check_is_fitted(self)
        577
        578         return self.tree_.compute_feature_importances()


        /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪check_is_fitted(estimator, attributes, msg, all_or_any)
        965
        966     if not attrs:
    --> 967         raise NotFittedError(msg % {'name': type(estimator).
↪__name__})
        968
        969


        NotFittedError: This DecisionTreeClassifier instance is not fitted yet.␣
↪Call 'fit' with appropriate arguments before using this estimator.
```

[ ]:

### 1.3.1 3.1. Analyze the performance of the model. How does it perform on test set? (5pt)

[ ]:

### 1.3.2 3.2. Which variables appear to be most important for predicting the success of a campaign? (5pt)

[ ]:

### 1.3.3 3.3. Interpret the decision tree produced by the model. What insights can you get from it? (5pt)

[ ]:

---

## 1.4 Task 4 (10pt):

Could the model be improved by using rebalancing techniques? To answer this, check how the model from Task 3 would perform on a balanced dataset. What performance measures are you considering?

[ ]:

---

## 1.5 Task 5 (25pt):

Use the data to train two more competing classification models (in addition to the decision tree models from Task 3 and 4). Justify your choice of settings for the models.

[ ]:

### 1.5.1 5.1. Which model has the highest accuracy in testing data? (5pt)

[ ]:

### 1.5.2 5.2. Compare the most important predictors of the different models and comment on their similarity and/or difference. Do they make sense? (5pt)

[ ]:

### 1.5.3 5.3. Plot ROC graphs to compare model performance. Which of the models would you choose based on training and testing data? Plot additional graphs to evaluate the models and interpret them. (5pt)

[ ]:

### 1.5.4 5.4. Suppose that the cost of a single test is 500, and the reward (or costs saved) due to a successfully detected case is 13000. What kind of advice you would give them? You can use various model performance charts to support you recommendation. Note that this is a 10 point question! Verbally explain and justfy your recomendation to the hospital stakeholders! (10pt)

[ ]: