Data Mining Final Project

Suicide Attempt Prediction using Classification

Arif Ahmet Balik
180303019

Computer Engineering
Istanbul Arel University
Buyukcekmece, Turkey
21.05.2019

Introduction

This report covers prediction and analysis of *ForeverAlone* dataset which will be described in Data section. The aim is to create a model that can accurately predict suicidal status of a person based on some of their personal data. Throughout the modeling, a few different method will be tested (Decision tree, rainforest, etc.). There will be experiments on different settings of data modeling experiments, and results will be discussed in more detail. Also data normalization, and reasoning on the dataset will be shown in steps.

At the end a model will be chosen and source code will be added in the document.

1 Dataset and Normalization

The dataset is created with contributions of reddit users on the subreddit */r/ForeverAlone*. A first look at data columns gives a hint for which features can be used to create a stable model.

```
time [date]
gender [string]
sexuallity [string]
age [numeric]
income [string]
race [string]
bodyweight [string]
virgin [string]
prostitution_legal [string]
pay_for_sex [string]
friends [numeric]
social_fear [string]
depressed [string]
what_help_from_others [string]
attempt_suicide [string]
employment [string]
job_title [string]
edu_level [string]
improve_yourself_how [string]

Total columns : 19
```

1.1 Preprocessing

There are missing values of course. In the program all rows that contains corrupted and NaN values are removed from dataset. For some there is no way to fill the missing values because it contains too much information that can't be accurately generated. More details are given in the next section.

1.2 Normalization

As seen in data types, many columns needs normalization in order a proper model to be constructed. But many of those columns contains too many different values inside, for instance income and race of the person is as follows;

	income	**race**	**...**
1	$20,000 to $29,999	White non-Hispanic	
2	$1 to $10,000	Asian	
3	...	...	

There are 13 different income and 25 different race value in total. This makes it very hard to apply normalization.

Following tables show normalization of some columns.

	attempt_suicide, depressed, social_fear, prostitution_legal, virgin	
1	**Original**	**Normalized**
2	Yes	1
3	No	0

	pay_for_sex	
1	**Original**	**Normalized**
2	No	0
3	Yes	1
4	Yes but I haven't	2

	gender	
1	**Original**	**Normalized**
2	Male	0
3	Female	1
4	Transgender male	2
5	Transgender female	3

2 Modeling and Experiments

2.1 Modeling

The type of modeling is classification, since prediction only outputs true or false, based on the data. Three methods are used in the project.

- Decision Tree
- Random Forest
- K-Nearest Neighbors
- MLP Neural Network

Also following list shows featured columns used in model;

```
gender [numeric]
age [numeric]
virgin [numeric]
prostitution_legal [numeric]
pay_for_sex [numeric]
friends [numeric]
social_fear [numeric]
depressed [numeric]
income [numeric]
bodyweight [numeric]
Total columns : 10
```

In the next section all of those methods will be tested along with some parameters described also in the next section. After experiments, results will be discussed and best one (accuracy-wise) will be chosen.[1]

2.2 Experiments

[1]Assuming at least one experiment's accuracy is grater than 0.8

2.2.1 Decision Tree

Output

```
Model: Decision Tree
Experiment # 1.1 , Accuracy:  0.7954545454545454
Test Size:  10.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        37
           1       0.25      0.14      0.18         7
Confusion Matrix:
 [[34  3]
 [ 6  1]]

Experiment # 1.2 , Accuracy:  0.8181818181818182
Test Size:  10.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
'bodyweight']
              precision    recall  f1-score   support

           0       0.87      0.92      0.89        37
           1       0.40      0.29      0.33         7
Confusion Matrix:
 [[34  3]
 [ 5  2]]

Experiment # 1.3 , Accuracy:  0.6628571428571428
Test Size:  40.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.82      0.75      0.79       144
           1       0.18      0.26      0.21        31
Confusion Matrix:
 [[108  36]
 [ 23   8]]

Experiment # 1.4 , Accuracy:  0.72
Test Size:  40.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
 'bodyweight']
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       144
           1       0.26      0.32      0.29        31
Confusion Matrix:
 [[116  28]
 [ 21  10]]

Best Experiment: # 1.2 , Accuracy:  0.8181818181818182
```

2.2.2 Random Forest

Output

```
Method: Random Forest
Experiment # 2.1 , Accuracy:  0.7954545454545454
Test Size:  10.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.83      0.95      0.89        37
           1       0.00      0.00      0.00         7
Confusion Matrix:
 [[35  2]
 [ 7  0]]

Experiment # 2.2 , Accuracy:  0.8181818181818182
Test Size:  10.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
'bodyweight']
              precision    recall  f1-score   support

           0       0.85      0.95      0.90        37
           1       0.33      0.14      0.20         7
Confusion Matrix:
 [[35  2]
 [ 6  1]]

Experiment # 2.3 , Accuracy:  0.8057142857142857
Test Size:  40.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.84      0.94      0.89       144
           1       0.38      0.16      0.23        31
Confusion Matrix:
 [[136   8]
 [ 26   5]]

Experiment # 2.4 , Accuracy:  0.7885714285714286
Test Size:  40.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
'bodyweight']
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       144
           1       0.35      0.23      0.27        31
Confusion Matrix:
 [[131  13]
 [ 24   7]]

Best Experiment: # 2.2 , Accuracy:  0.8181818181818182
```

2.2.3 K-Nearest Neighbors

Output

```
Method: K-Nearest Neighbors
Experiment # 3.1 , Accuracy:  0.7954545454545454
Test Size:  10.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.83      0.95      0.89        37
           1       0.00      0.00      0.00         7
Confusion Matrix:
 [[35  2]
 [ 7  0]]

Experiment # 3.2 , Accuracy:  0.7954545454545454
Test Size:  10.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
'bodyweight']
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        37
           1       0.25      0.14      0.18         7
Confusion Matrix:
 [[34  3]
 [ 6  1]]

Experiment # 3.3 , Accuracy:  0.7828571428571428
Test Size:  40.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.82      0.94      0.88       144
           1       0.11      0.03      0.05        31
Confusion Matrix:
 [[136   8]
 [ 30   1]]

Experiment # 3.4 , Accuracy:  0.7714285714285715
Test Size:  40.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
 'bodyweight']
              precision    recall  f1-score   support

           0       0.83      0.91      0.87       144
           1       0.24      0.13      0.17        31
Confusion Matrix:
 [[131  13]
 [ 27   4]]

Best Experiment: # 3.1 , Accuracy:  0.7954545454545454
```

2.2.4 MLP Neural Network

Output

```
Model: MLP Neural Network

Experiment # 4.1 , Accuracy:  0.8636363636363636
Test Size:  10.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.86      1.00      0.92        37
           1       1.00      0.14      0.25         7
Confusion Matrix:
 [[37  0]
 [ 6  1]]

Experiment # 4.2 , Accuracy:  0.8409090909090909
Test Size:  10.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
 'bodyweight']
              precision    recall  f1-score   support

           0       0.84      1.00      0.91        37
           1       0.00      0.00      0.00         7
Confusion Matrix:
 [[37  0]
 [ 7  0]]

Experiment # 4.3 , Accuracy:  0.8285714285714286
Test Size:  40.0 %
Feature Columns:  ['age', 'friends', 'depressed', 'social_fear', 'virgin',
 'gender', 'pay_for_sex', 'income', 'bodyweight']
              precision    recall  f1-score   support

           0       0.84      0.98      0.90       144
           1       0.57      0.13      0.21        31
Confusion Matrix:
 [[141   3]
 [ 27   4]]

Experiment # 4.4 , Accuracy:  0.8057142857142857
Test Size:  40.0 %
Feature Columns:  ['friends', 'depressed', 'social_fear', 'virgin', 'income',
 'bodyweight']
              precision    recall  f1-score   support

           0       0.83      0.97      0.89       144
           1       0.29      0.06      0.11        31
Confusion Matrix:
 [[139   5]
 [ 29   2]]

Best Experiment: # 4.1 , Accuracy:  0.8636363636363636
```

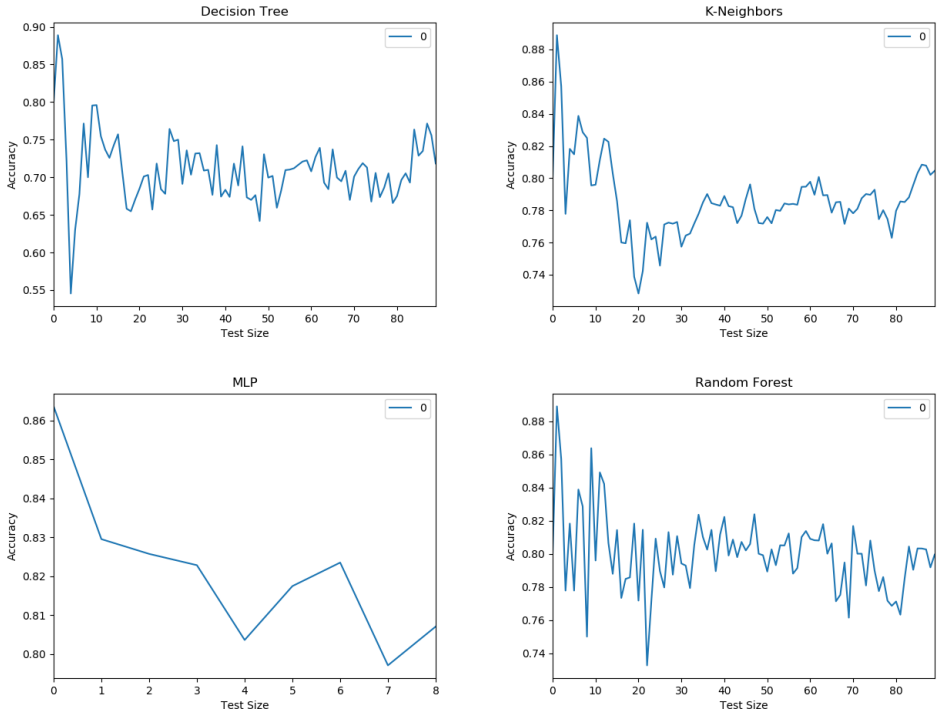

Figure 1: Accuracy with different test sizes.

3 Conclusion

It seems that increasing the test size, or decreasing the features does not help the models to generate better outcomes. Figure 2. shows accuracy response over changing test size.

3.1 Decision Tree

The best one among four experiments : **Experiment #1.2**

In the first method (Decision Tree) results seems relatively good. Even thou in the best experiment (#1.2) it has a good score, for *True Negative* values, experiment #1.1 was better. Best model has the third place among other models with **3** *False Positive* and **2** *False Negative* in total of **44** predictions.

3.2 Random Forest

The best one among those experiments : **Experiment #2.2**

It seems Decision Tree and this model both has the same accuracy value, but it can be seen from confusion matrix that *True Positive/Negative* predictions are higher in this model than those predicted in Decision Tree.

On the other hand average of four experiments also higher in this method, so it is clear that Random Forest has advantages over Decision Tree.

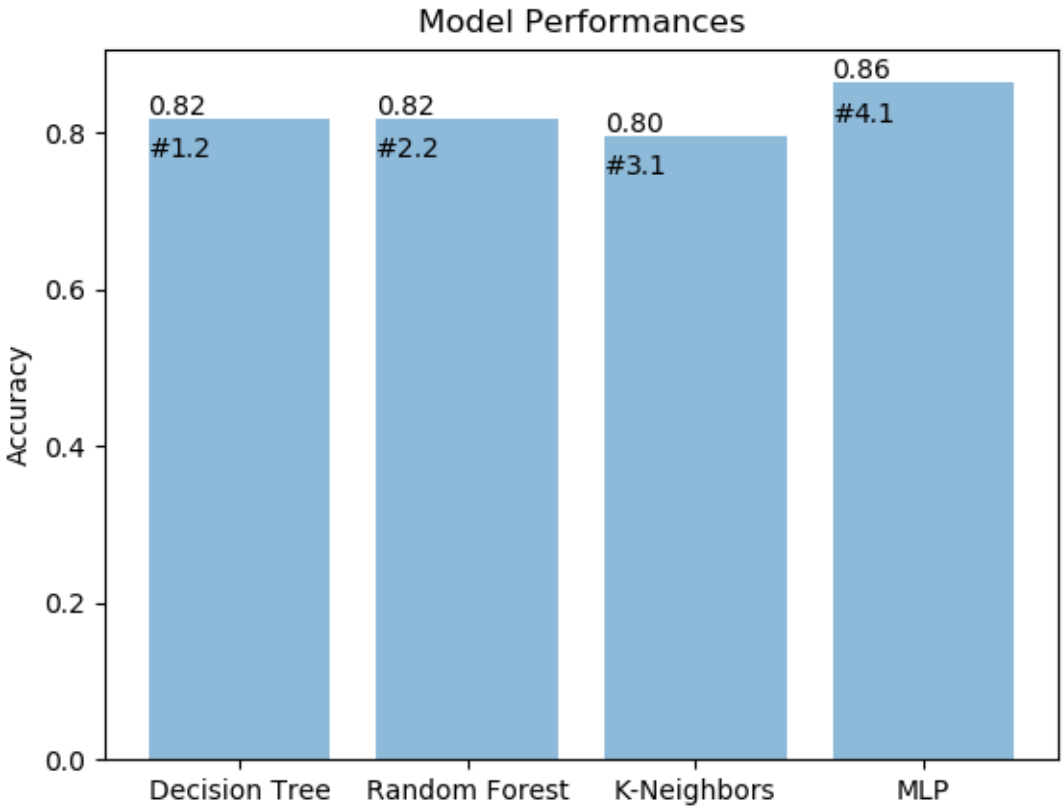

Figure 2: Model performances

3.3 K-Nearest Neighbors

The best one among those experiments : **Experiment #3.1** It turned out this method wasn't so successful for this dataset. After many runs, it gave the lowest accuracy value every time. But it seems from experiments that this model was rather successful about predicting *True Negative* values.

3.4 MLP Neural Network

The best one among those experiments : **Experiment #4.1** This model was the most successful one. In the experiment #4.2 predictions was %100 correct but even that wasn't the most successful one, the reason is unclear but experiment #4.1 turned out to be the best.

Appendix A

Supplementary materials are located on github.com/arifbalik/DMProject

Following program runs four different models mentioned above, every model is tested with 4 different settings and best result is stored.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt; plt.rcdefaults()
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')

pima = pd.read_csv("foreveralone.csv")

weight_m = {'Normal weight': 0, 'Underweight': 1, 'Overweight': 2, 'Obese': 3}
pima['bodyweight'] = pima['bodyweight'].map(weight_m)

income_m = {'$30,000 to $39,999': 0, '$1 to $10,000': 1, '$0': 2, '$50,000 to $74,999': 3,
'$20,000 to $29,999': 4, '$10,000 to $19,999': 5, '$75,000 to $99,999': 6,
 '$150,000 to $174,999': 7, '$125,000 to $149,999': 8, '$100,000 to $124,999': 9,
 '$174,999 to $199,999': 10, '$40,000 to $49,999': 11, '$200,000 or more':  12}
pima['income'] = pima['income'].map(income_m)

gender_m = {'Male': 0, 'Female': 1, 'Transgender male':2, 'Transgender female':3}
pima["gender"] = pima["gender"].map(gender_m)

yes_no_m = {'Yes': 1, 'No': 0}
pima["attempt_suicide"] = pima["attempt_suicide"].map(yes_no_m)
pima["depressed"] = pima["depressed"].map(yes_no_m)
pima["social_fear"] = pima["social_fear"].map(yes_no_m)
pima["virgin"] = pima["virgin"].map(yes_no_m)
pima["prostitution_legal"] = pima["prostitution_legal"].map(yes_no_m)

pay_m = {"No": 0, 'Yes': 1, "Yes but I haven't": 2}
pima["pay_for_sex"] = pima["pay_for_sex"].map(pay_m)

pima.dropna(inplace=True)



feature_cols = ['age','friends', 'depressed','social_fear', "virgin", "gender", "pay_for_sex", "incon
feature_cols2 = ['friends', 'depressed','social_fear', "virgin", "income", "bodyweight"]

y = pima.attempt_suicide

MLP = MLPClassifier(solver='adam', learning_rate='adaptive')
DT = DecisionTreeClassifier()
RF = RandomForestClassifier()
KN = KNeighborsClassifier()
```

```
performance = []
results = []
best_exp = []
for I in range(4):
    if (I == 0):
        model = DT
        print("Model: Decision Tree")
    elif (I == 1):
        model = RF
        print("Model: Random Forest")
    elif (I == 2):
        model = KN
        print("Model: K-Neighbors")
    elif (I == 3):
        model = MLP
        print("Model: MLP Neural Network")
    for K in range(4):
        if(K == 0):
            t_size = 0.1
            col = feature_cols
            X = pima[feature_cols]
        elif(K == 1):
            t_size = 0.1
            col = feature_cols2
            X = pima[feature_cols2]
        elif (K == 2):
            t_size = 0.4
            col = feature_cols
            X = pima[feature_cols]
        elif (K == 3):
            t_size = 0.4
            col = feature_cols2
            X = pima[feature_cols2]
        exp_no = (I + 1) + ((K+1) / 10)

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=t_size, random_state=1)
        model = model.fit(X_train, y_train)
        pred = model.predict(X_test)

        print("\nExperiment #",exp_no, ", Accuracy: ", metrics.accuracy_score(y_test, pred))
        print("Test Size: ", t_size * 100 ,"%")
        print("Feature Columns: ", col)
        print(classification_report(y_test, pred))
        print("Confusion Matrix:\n", confusion_matrix(y_test, pred))
        results.append(metrics.accuracy_score(y_test, pred))

    performance.append(max(results))
    best_exp.append((I + 1) + (results.index(max(results)) +1)/10)
    print("\nBest Experiment: #",best_exp[I],", Accuracy: ", max(results))
    del results[:]


objects = ( 'Decision Tree', 'Random Forest', 'K-Neighbors', 'MLP')
y_pos = np.arange(len(objects))

bars = plt.bar(y_pos, performance, align='center', alpha=0.5)
```

```
plt.xticks(y_pos, objects)
ix = 0
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), yval - .05, '#')
    plt.text(bar.get_x() + 0.09, yval - .05, best_exp[ix])
    plt.text(bar.get_x(), yval + .005, '%.2f' % yval)
    ix = ix + 1
plt.ylabel('Accuracy')
plt.title('Model Performances')
plt.show()
```

Appendix B

Following figures shows tree diagrams of the Decision Tree algorithm