Empathy Prediction

# Objective

To predict a person’s nature as either “very empathetic” or “not very empathetic” using the Young People Survey dataset.

# Dataset

The dataset contains survey information of 1010 individual. The survey questionnaire included 150 questions with multiple choice type answers.

# Preprocessing

Preprocessing required extensive use of functions from the *Pandas* and *Numpy* libraries. It involved the following steps:

1. Converting .csv file to a data frame.
2. Setting values in Empathy column to ‘0’ and ‘1’ for values {1, 2, 3} and {4, 5} respectively.
3. Removing rows in which Empathy is NaN. These columns won’t help in training or testing.
4. Removing NaN values from all the columns using *mode* of respective columns.
5. Splitting categorical features such as Gender, smoking habits, drinking habits with One Hot Encoding approach. Obtained 184 features in total.
6. Splitting dataset into training, validation and test set. 10% of the data was made the test set; 10% of the remaining data was made validation set; and rest of the data was made training set.

# Feature Importance approach

Used two types of training set for training the models:

1. **Entire data:** This set uses all the 183 features to train the model
2. **Correlated features:** This set uses 36 features which were highly correlated to the target feature ‘Empathy’.

# Training

Used following models to predict empathy of an individual with help of *scikit-learn* library:

1. Most frequent classification (mf, Baseline)
2. Random classification (rc, Baseline)
3. Logistic regression (lr)
4. Perceptron (perc)
5. K Nearest Neighbors (knn)
6. Random Forest (rf)
7. Support Vector Machine (svm)

# Result

|  |  |  |
| --- | --- | --- |
| Model | All Features | 36 Correlated features |
| Random classification | 45%-60% | 45%-60% |
| Most frequent Class | 67.33% | 67.33% |
| Logistic regression | 68.31% | 64.36% |
| Perceptron | 68.32% | 71.29% |
| K Nearest Neighbors | 67.32% | 69.31% |
| **Random Forest** | **74.25%** | 69.30% |
| Support Vector Machine | 72.27% | 71.28% |

# Conclusion

We see that **Random Forest** performs the best. The main reasons for this is the simplicity of the algorithm, use of the bagging technique, ability to handle high dimensional data, robustness for binary classification. This model performs up to **7%** better than the baseline model.

The advantages of eliminating features based on correlation with the target feature was only limited to KNN. KNN treats all features equally, hence eliminating uncorrelated features helps the model improve the prediction. Accuracy is improved up to **2%**.

Also tried normalizing the data to values [0, 1], but accuracy of various models decreased. This might have happened as the data was categorical. Normailizing would generally help is the data is numerical.

# Running the project

Following functions can be called to perform various steps:

**import**

Train, Test, PreProc

**Preprocess (optional, clean data present in the folder)**

PreProc.LoadCsv('youngpeoplesurvey/responses.csv')

**Select either all features of correlated features:**

PreProc.AllSet() – call this function before training

PreProc.corrSet() - call this function before training

**Train:**

Train.rc(), Train.mf()

Train.knn(10) - parameter - K from 1 to n

Train.rf(15) parameter - max depth from 1 to n

Train.lr()

Train.perc(100) - parameter - epochs from 1 to n

Train.sv(10) - paramemter - C from 1 to n

**Test:**

Test.test("Random Classifier")

Test.test("Most Frequent")

Test.test("Logistic Regression")

Test.test("Perceptron")

Test.test("K Nearest Neighbors")

Test.test("Random Forest")

Test.test("Support Vector Machine")

# References

1. <https://scikit-learn.org/stable/user_guide.html>
2. <http://pandas.pydata.org/pandas-docs/version/0.15/index.html>
3. <https://docs.scipy.org/doc/numpy/user/quickstart.html>