Detecting Conflict in Couples

Aditya Gujral

Problem Statement

With the increase in wearable technology, we have more data for individuals that can be used for various applications

This projects aims to detect conflict among couples based on audio and sensor data from individuals.

Data Pre-processing

The dataset is sampled at ~5 minute intervals and includes features such as GPS distance between couple, ECG, EDA, processed audio data, and ~80 other features.

The data had missing values(/NaN), and features with all missing values were dropped.

Samples for which output label was missing were dropped.

Remaining missing values were replaced by feature mean

Models

Support Vector Machines

Feedforward Neural Network

Long Term Short Memory (yay a fancy name!) - Basically a Recurrent Neural Network, that does not have the shortcoming of vanishing gradient

Hyper-parameter estimation

Split the data in training and testing data. Say 80-20

Use cross validation on remaining data for hyper-parameter estimation

Unequal class representation

The class ‘conflict’ had only 5% representation in the set, hence any predictor that always predicted ‘no conflict’ had 95% accuracy! - All unweighted models were indeed doing this.

Use a weighted approach, where misclassifying class ‘conflict’ has a higher penalty!

Accuracy is a bad measure here! Need a better measure

- Recall
- Precision
- Pearson correlation coefficient

Judging Accuracy

If the left half was the relevant samples, and we selected the circle, then the recall is highlighted green by total green samples.
Precision is the percentage with which we picked the correct class

Pearson correlation coefficient(x,y) =

Cov(x, y) / (std(x) * std(y))

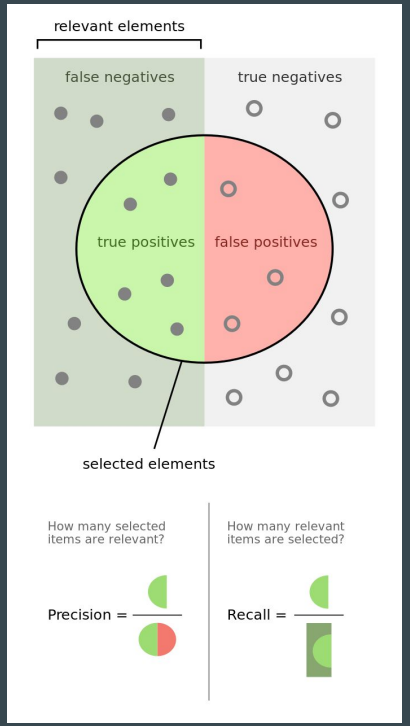

Image source : https://en.wikipedia.org/wiki/File:Precisionrecall.svg

Feature reduction

Tried auto-encoder, but did not necessarily improve the results.

Results

Model	0.5 * (Precision C1 + Precision C2)	0.5 * (Recall C1 + Recall C2)	Correlation Coefficient
Any unweighted model	0.47	0.5	-ve, Nan
SVM	0.57	0.765	0.273
NN	0.54	0.625	0.146
LSTM	0.57	0.82	0.301

Takeaways

SVM usually performs very well with minimal effort (based on whatever few projects I have tried). Few hyperparameter, and awesome baseline!

Start simple!

If Data is scarce, complex models usually will not work so good!

Do not ever touch the testing data, ever!