Custom Test Train Splits for Refined Random Forest Models

Stefan Bund, bund@umich.edu

Project Outline

This synthesis project is based on a dataset of physiological sensor measurements collected from Smartphone based sensors. The original research sought to determine the particular activity of the subject based on the physiological measurements obtained from wearables and a SmartPhone.

The physiological measurements were used to depict the test subject in one of four activities as follows:

- neutral
- emotional
- mental
- physical

The study will produce a model that, based on a limited number of features, returns the best possible estimate of the activity being performed by the test subject. While the original analysis utilized more advanced Machine Learning methods, we will concentrate on the supervised learning methods covered in this course.

Data Schema

The sensor dataset consists of 4480 rows, each with the subject ID, the activity label and 533 measurement features. Each of the 40 test volunteers were subjected to a series of 28 data collection events for each of the four activity types,

- neutral
- emotional
- mental
- physical

	Subject_ID	Activity_Label	ECG_original_mean_1	ECG_original_std_2	ECG_original_trimmean25_3	E
0	QNNG	1	-0.004125	0.254095	0.001426	
1	QNNG	1	0.031029	0.193761	0.012918	
2	QNNG	1	0.015678	0.182336	-0.003028	
3	QNNG	1	0.014525	0.176636	-0.006161	
4	QNNG	1	0.010349	0.179248	-0.008526	
...	...	...	...	...	...	
245	GVJD	1	0.000069	0.187193	-0.043176	
246	GVJD	1	-0.001865	0.186908	-0.046055	
247	GVJD	1	-0.002578	0.193167	-0.043957	
248	GVJD	1	-0.003646	0.193385	-0.045367	
249	GVJD	1	-0.000650	0.195513	-0.042715	

250 rows × 535 columns

Column Depth, and Managing Qualitative Labeling

The width of the feature set dimensional complexity.

This must be resolved, in addition to non-quantitative labels, which must be consumed with a customized test/train split

```
Index(['Subject_ID', 'Activity_Label', 'ECG_original_mean_1',
       'ECG_original_std_2', 'ECG_original_trimmean25_3',
       'ECG_original_median_4', 'ECG_original_skewness_5',
       'ECG_original_kurtosis_6', 'ECG_original_max_7', 'ECG_original_min_8',
       ...
       'EDA_Functionals_power_Filt2skewness_524',
       'EDA_Functionals_power_Filt2kurtosis_525',
       'EDA_Functionals_power_Filt2max_526',
       'EDA_Functionals_power_Filt2min_527',
       'EDA_Functionals_power_Filt2prctile25_528',
       'EDA_Functionals_power_Filt2prctile75_529',
       'EDA_Functionals_power_Filt2geomean(abs)_530',
       'EDA_Functionals_power_Filt2harmmean_531',
       'EDA_Functionals_power_Filt2mad_532',
       'EDA_Functionals_power_Filt2baseline_533'],
      dtype='object', length=535)
```

The Baseline Model

Our initial decision tree was based on a stratified 80/20 test train split, where columns were denoted by the string ‘**base_feature_selector’** in our global data set.

This initial set of features enabled an **R square value of 81.6%**, using a generic sklearn Decision Tree.

Our goals were to improve this score by selecting an appropriate subset of the data and a more apt method of supervised learning.

```
(['ECG_original_mad_13',
  'ECG_RR_window_mad_27',
  'ECG_amplitude_RR_mad_41',
  'ECG_HR_min_div_mad_55',
  'ECG_hrv_mad_79',
  'ECG_PSD_mad_93',
  'ECG_p_VFL_mad_107',
  'ECG_p_LF_mad_121',
  'ECG_p_MF_mad_135',
  'ECG_p_HF_mad_149',
  'ECG_p_total_LF_mad_163',
  'IT_Original_mad_187',
  'IT_LF_mad_202',
  'IT_RF_mad_217',
  'IT_BRV_mad_233',
  'IT_PSD_mad_247',
  'IT_VLF_mad_261',
  'IT_LF_mad_275',
  'IT_MF_mad_289',
  'IT_HF_mad_303',
  'IT_p_Total_mad_317',
  'EDA_Original_mad_338',
  'EDA_processed_mad_352',
  'EDA_Filt1_mad_366',
  'EDA_Filt2_mad_380',
  'EDA_Original_mad_442',
  'EDA_processed_mad_456',
  'EDA_Filt1_mad_470',
  'EDA_Filt2_mad_484'],
 0.8158482142857143)
```

Data Leakage in Generic Test Train Splits

The 81% accuracy score reflects the tree model's ability to correctly predict a classifier, given the multiple labels in the dataset. The breadth of labels suggests that the model possesses some rigor. The depth of the tree which was used is the default. This implies that the tree will use a minimum of two levels in order to predict a class of object.

I am concerned about data leakage in this model. There are only four basic Activity Labels, which are dispersed across roughly four thousand instances, where only 40 users recorded their data.

This implies that using a standard test-train split may expose the training data to leakage.

Given the test split of 20%, this allows for a healthy segment of the data set to be used for training. However, if the individual users' data leaks between train and test, the trainer will have absorbed biased data.

Devising a Custom Test Train Split

The number of labeled columns trained and fit by the tree creates a rich set of classifying features, but the strategy of the test/split will expose the model to situations where conditions are shared between test and train, thus creating an illusion that the prediction is better than it is.

The reality is that rules learned by the decision tree in the training phase actually consumes use-cases which will exist within the test cases, thereby creating more accuracy in the score, without providing actual predictive capability in the model.

The current strategy does not respect the boundary between use cases, grouped by Subject_ID. This means that as production data is put into the model, its performance will likely reduce.

Pitfalls: Initial Decision Tree Baselines Fail

1. the custom test/train split, scored significantly lower than the original baseline (~68% versus ~81%).

2. in constructing the new custom split, a rigorous boundary was created between test and training data, through these techniques:

- in the custom split, a randomization function removes grouped or sequenced entries, thus removing sequential sets of rows from whatever context they had. This removes bias in the data set, and improves generality.
- the new custom split also ensured data could not leak between train and test, thus causing the score to be artificially accurate. The custom split explicitly prevented testing data from being fit, using our predictive measure.
- I suspect that the baseline score did result from randomization. Thus, some bias could have been removed from the standard split. However, the subject_id field was a key organizational tool, in the custom split.

Removing Bias in Initial Baseline Trees

Thus, subject_id was used to organize the subjects, not row. This means that no subject_id entries could have been leaked between train and test sets, thus improving the prediction score and undermining its accuracy against false positives.

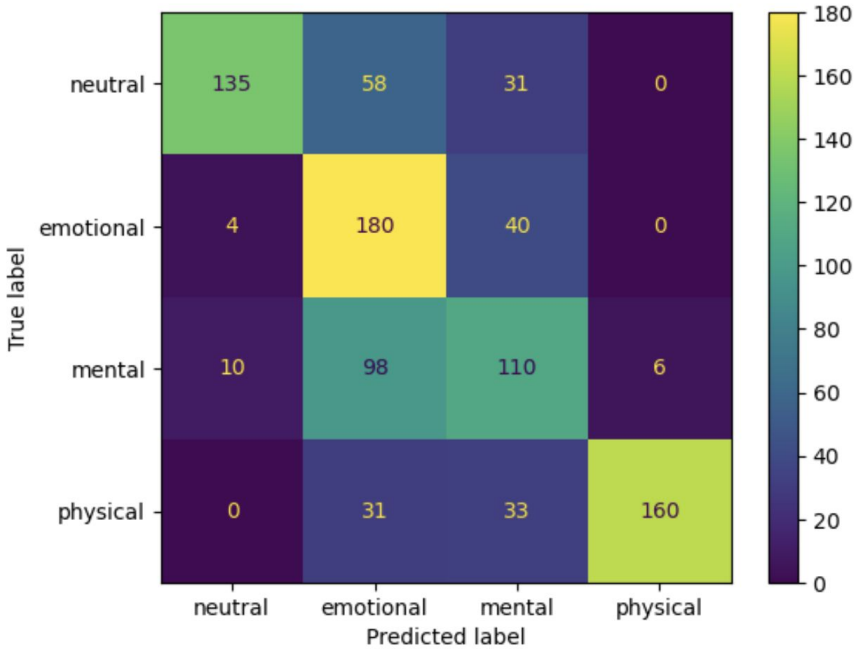
The initial Decision Tree delivers a healthy accuracy score, until the biased elements are removed, via randomized test/train splitting.

The actual, unbiased accuracy, using the same feature set, plummets to ~63%.

```
(['ECG_original_mad_13', 'ECG_RR_window_mad_27', 'ECG_amplitude_RR_mad_41', 'ECG_HR_min_div_mad_55', 'ECG_hrv_mad_
79', 'ECG_PSD_mad_93', 'ECG_p_VFL_mad_107', 'ECG_p_LF_mad_121', 'ECG_p_MF_mad_135', 'ECG_p_HF_mad_149', 'ECG_p_tot
al_LF_mad_163', 'IT_Original_mad_187', 'IT_LF_mad_202', 'IT_RF_mad_217', 'IT_BRV_mad_233', 'IT_PSD_mad_247', 'IT_V
LF_mad_261', 'IT_LF_mad_275', 'IT_MF_mad_289', 'IT_HF_mad_303', 'IT_p_Total_mad_317', 'EDA_Original_mad_338', 'EDA
_processed_mad_352', 'EDA_Filt1_mad_366', 'EDA_Filt2_mad_380', 'EDA_Original_mad_442', 'EDA_processed_mad_456', 'E
DA_Filt1_mad_470', 'EDA_Filt2_mad_484'], 0.6372767857142857)
```

Visualizing the new Baseline Accuracy

```
total tests 896
total predictions 896 <class 'numpy.ndarray'>
```



Once sequences are removed and a unbiased test/train set of samples is prepared, the baseline accuracy begins at roughly 63%, where ‘emotional’ types are best predicted.

Isolating the Most Promising Features, by Gini Importance

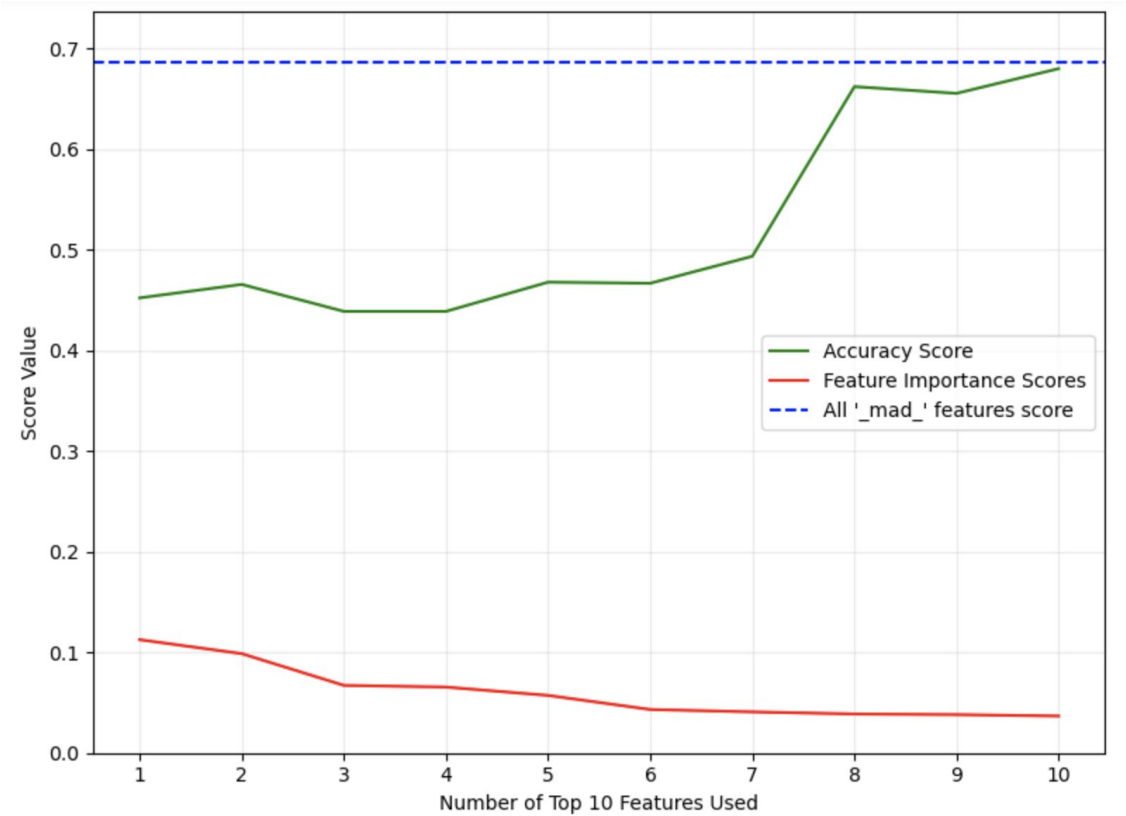
A search was done to locate the most highly important features among the original feature set, and gauge the improvement in accuracy, while implementing a Random Forest model.

The accuracy improves to ~68%, while highlighting features with the highest Gini Importance.

```
([('IT_Original_mad_187', 0.11256580865773229),
  ('IT_LF_mad_202', 0.09870328537109924),
  ('IT_RF_mad_217', 0.0671973359402965),
  ('ECG_original_mad_13', 0.0654974139451176),
  ('ECG_amplitude_RR_mad_41', 0.05722319121238872),
  ('EDA_processed_mad_456', 0.043237638008223805),
  ('ECG_HR_min_div_mad_55', 0.04086105034492251),
  ('IT_p_Total_mad_317', 0.038767754759935644),
  ('IT_PSD_mad_247', 0.03813835740667452),
  ('ECG_RR_window_mad_27', 0.0367376843873016 2)],
 0.6863839285714286)
```

Per https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.feature_importances_

Experimentally Adding Features, to Improve Accuracy



The top-most important features were incrementally added to the model, then tested for AUC-ROC accuracy.

Generally, adding more high-importance features delivered the most accurate results.

Locating the topmost correlated features

A correlation matrix was used to identify the most highly correlated pairs of columns, with relation to the classifying label. Approximately 240 top qualifying pairs were located.

```
corr_matrix = df10.corr()
corr_matrix = corr_matrix.where(~(corr_matrix >= .9)).stack().nlargest(50000)
print(corr_matrix)
```

```
IT_HF_mean_291                  IT_p_Total_min_312                0.899960
IT_p_Total_min_312              IT_HF_mean_291                    0.899960
EDA_processed_trimmean25_342    EDA_processed_median_343          0.899883
EDA_processed_median_343        EDA_processed_trimmean25_342      0.899883
ECG_RR_window_min_22            ECG_RR_window_harmmean_26         0.899827
                                                                    ...
EDA_Original_min_437            IT_Original_std_176               0.141394
IT_Original_harmmean_186        EDA_processed_geomean(abs)_454    0.141393
EDA_processed_geomean(abs)_454  IT_Original_harmmean_186          0.141393
ECG_hrv_prctile75_76            IT_BRV_min_228                    0.141389
IT_BRV_min_228                  ECG_hrv_prctile75_76              0.141389
Length: 50000, dtype: float64
```

Delivering an improved model

The final step entailed a grid search among the 240 most correlated features, and locating the most highly accurate set. A random forest classifier was used to try permutations of the most correlated sets of features, to improve the model to ~87% accuracy.

```
test size => 8
(RandomForestClassifier(n_jobs=-1, random_state=42), ['ECG_original_std_2', 'IT_MF_skewness_281', 'ECG_original_mad_13', 'IT_BR_
mean_220', 'ECG_original_geomean(abs)_11', 'ECG_original_prctile75_10', 'ECG_RSAindex_61', 'ECG_HR_min_div_std_44', 'ECG_amplitu
de_RR_prctile75_38', 'IT_BRV_baseline_234'], 0.8761575919430272)
```