Wavelet Machine Learning Challenge

Problem Description:

Using the data collected on user heartbeats for two months, train a model to predict a future heartbeat to user association. There are 10 types of users thus making this a multi-class classification problem.

Approach:

1. Explore the data to understand the distributions and nature of the data. Key questions to explore
    - Is the data balanced? or does the training data have huge number of observations from one class and very less number of observations from another class?
    - Is the data linearly separable?
    - Are there any key features which show a clear significance in determining a user?
    - Are there any categorical variables or missing values? If yes, how do we handle them?
    - Are there are any variables with minimal variance? Are there any outliers?
2. Preprocess the data
    - Replace missing values
    - Convert data types to 32 bit for x86 and x64 compatibility
    - Standardize the data
3. Analyze key features
    - Recursive Feature elimination
    - Extra trees importance measures
4. Select a classification approach by training few models with default configurations
    - Support Vector Machine
    - Random Forest
    - K - nearest neighbor
    - Decision tree with Adaboost
5. Parameter tuning for selected techniques using grid search and cross validation
    - Grid search for Random Forest
    - Grid search for Decision tree and Adaboost
6. Recommendations
    - Best model by accuracy
    - Best model in terms of model train time and comparable accuracy
    - Best model by user type, this comes into picture if there is an asymmetric cost associated with a particular user misclassification
7. What is not covered
    - Building an ensemble voting classifier with at least 3 of the techniques and cross-validate it for accuracy
    - Further feature engineering to improve execution efficiency. Evaluating interaction between features and Box-cox transformations are a point to start.
    - Batching execution of model trainings for computational efficiency.

For details of final model and conclusion skip to **section 5** in *page 10*

1. Exploratory Data Analysis and preprocessing

1.1 Data types

Dataset has 234263 observations. One categorical response variable and 22 feature variables. All of them are continuous variables. No categorical variables in the predictors. Data types are as follows

```
 _user_id          234263 non-null int64
 _calendar_date    234263 non-null object
 _cap_seq          234263 non-null int64
 _datetime         234263 non-null object
feat1             200110 non-null float64
feat2             232062 non-null float64
feat3             232075 non-null float64
feat4             212084 non-null float64
feat5             234263 non-null float64
feat6             191450 non-null float64
feat7             212084 non-null float64
feat8             123212 non-null float64
feat9             224572 non-null float64
feat10            213987 non-null float64
feat11            218590 non-null float64
feat12            218590 non-null float64
feat13            218590 non-null float64
feat14            218590 non-null float64
feat15            218590 non-null float64
feat16            110981 non-null float64
feat17            190249 non-null float64
feat18            190249 non-null float64
feat19            164110 non-null float64
feat20            204867 non-null float64
feat21            232008 non-null float64
feat22            164078 non-null float64
```

1.2 Analyze and replace missing values

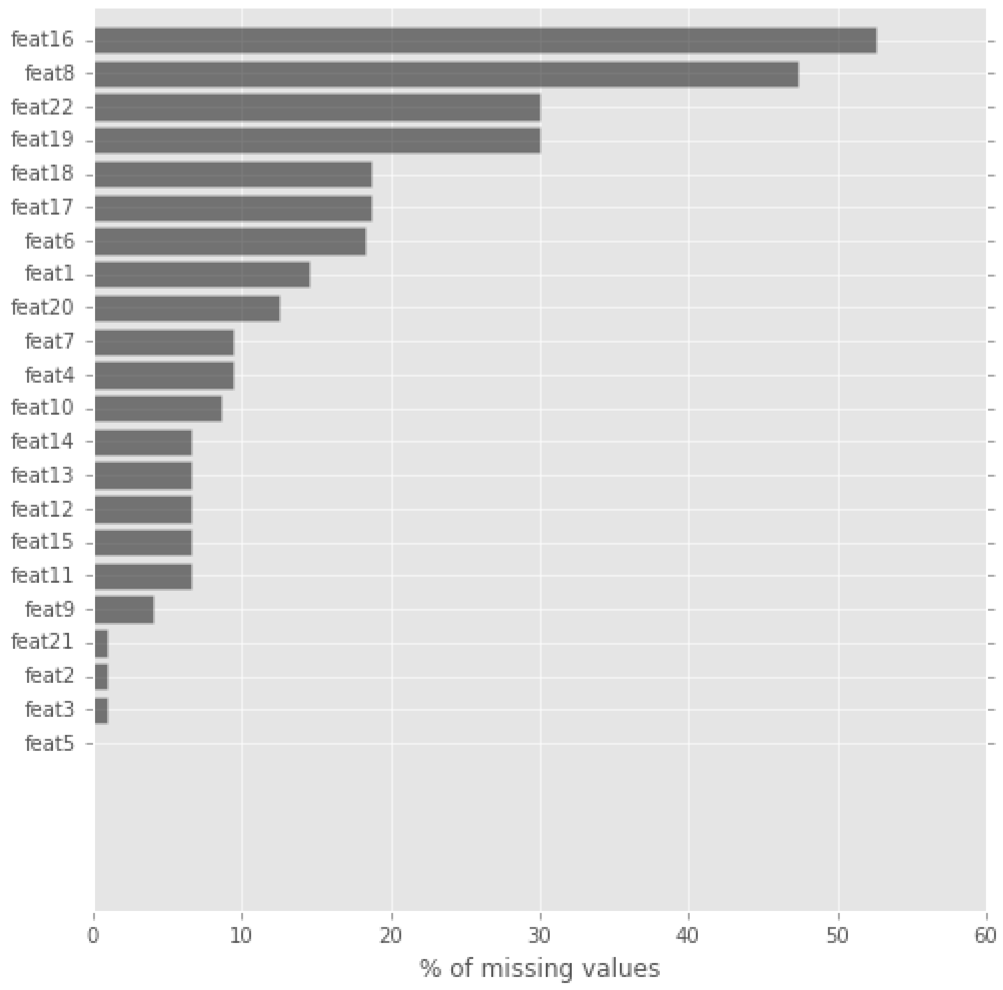

Fig: 1.2 % of Missing values by feature

Feat16 and feat8 have highest % of missing values. For techniques which work better with less number o f features like logistic regression. I would exclude these two variables from training feature set.

For the missing values, I would replace them with the mean of that particular userid. For example, a mis sing value in feat9 corresponding to user1 will be replaced by mean(feat9) for user1 observations, where as for user2 it will be replaced by mean(feat9) for user2 observations.

1.3 Standardize predictor variables Z-transform

To reduce the effect of high and low absolute values, we will standardize the data using a z-transform. That is observation = (observation- mean(observation))/std(observation). Resulting data will have mean=0 and std =1

1.4 Converting predictor variables to 32 bit

Type cast the data to 32 bit for numeric columns. Data at this stage will have all 32-bit data Types and standardized data with zero missing values.

1.5 Plotting the response variable

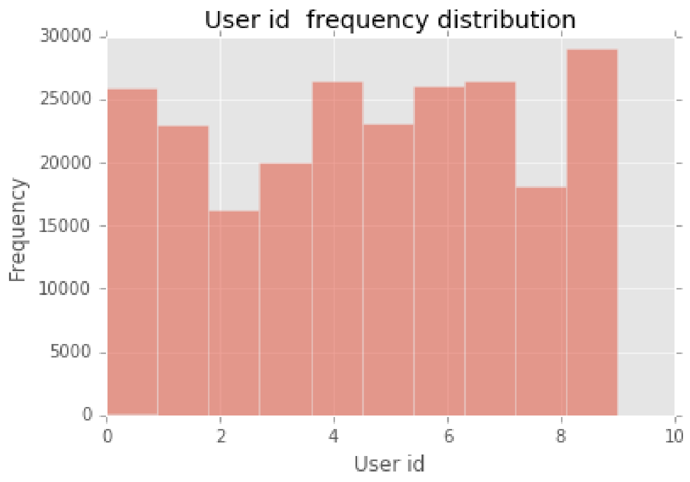

Fig 1.5 Response variable plot

As can be seen, data seems to be balanced with comparable number of observations in each category. User id 2 and 8 have slightly less number of observations relative to other categories, however the difference is not high enough to cause an imbalance in the categories.

1.6 Visualize data and analyze the data separation

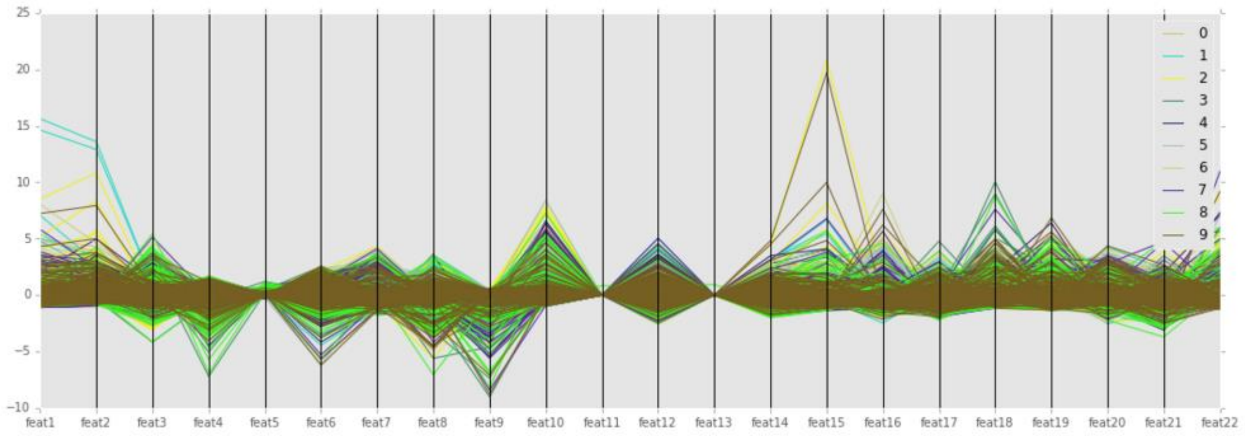

Fig 1.6 Parallel Plot for all the features

As can be observed from the plot(with sample data) there is no clear distinction between the user categories with the given features. Feat15 and feat9 seem to exhibit some variance, however the s pikes don't infer any clear separation in the categories of data.

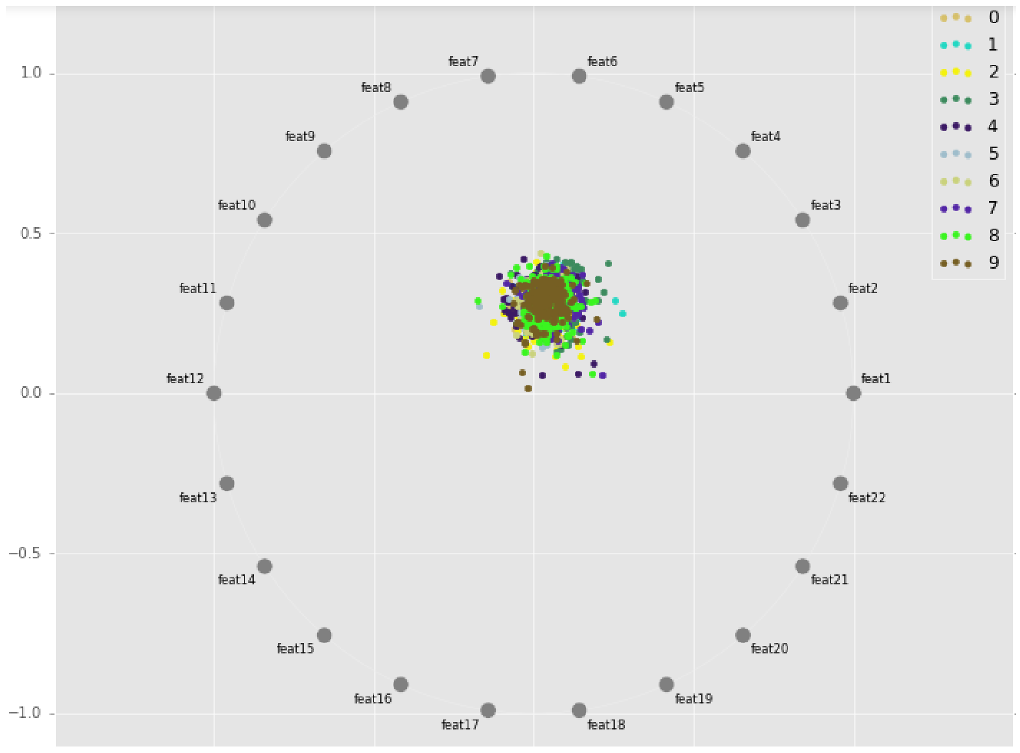

Fig 1.6.1 Visual test for linear separation with radviz spring constant plot

As observed data has no visually apparent boundaries and looks pretty closely spaced. This suggests that a kernel based approach may not be efficient. We will explore this further using different models in 'sele cting a classification method' section.

1.7 Histograms of predictor variables

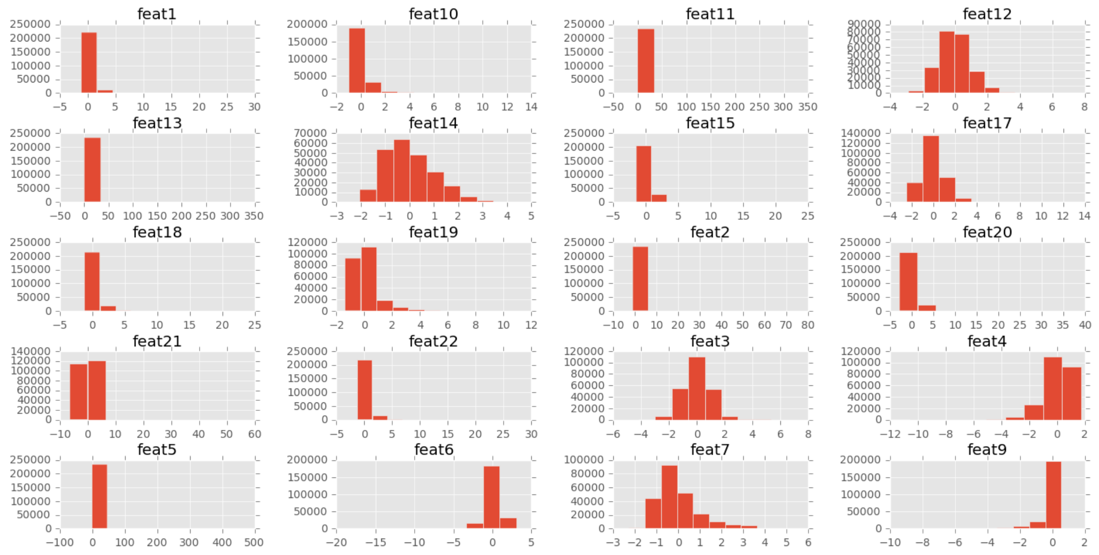

Fig 1.7 Histograms of predictor variables

As observed all the predictor variables exhibit reasonable variance, few of the features like feat5 have outliers leading to entire data being displayed in a single bin (due to scaling on x-axis).

Regarding outliers, I am not removing any outliers from the data. Considering this is heartbeat data, these outliers could prove to be distinguishing features for each user. For instance, a user with a specific condition might exhibit hear beat rates vastly different from a normal user.

2. Analyze Key Features

2.1 Recursive Feature Analysis and Extra trees importance

Using Logistic Regression model(RFE) and Extra trees classifier we evaluate the key features in determining the class of an observation. Results below

Feature	Tree ranking	RFE ranking
Feat 5	1	3
Feat 1	2	5
Feat 11	3	1
Feat 2	4	4
Feat 13	5	2
Feat 12	6	17
Feat 14	7	8
Feat 4	8	6
Feat 19	9	19
Feat 3	10	11
Feat 7	11	7
Feat 15	12	9
Feat 20	13	15
Feat 17	14	14
Feat 18	15	12
Feat 22	16	20
Feat 6	17	16
Feat 9	18	13
Feat 21	19	10
Feat 10	20	8

For the majority of the features both the approaches agree upon the importance. Both the approaches have 8 features common in the top 10 list this indicates that the feature selection is not random are specific to the feature selection time

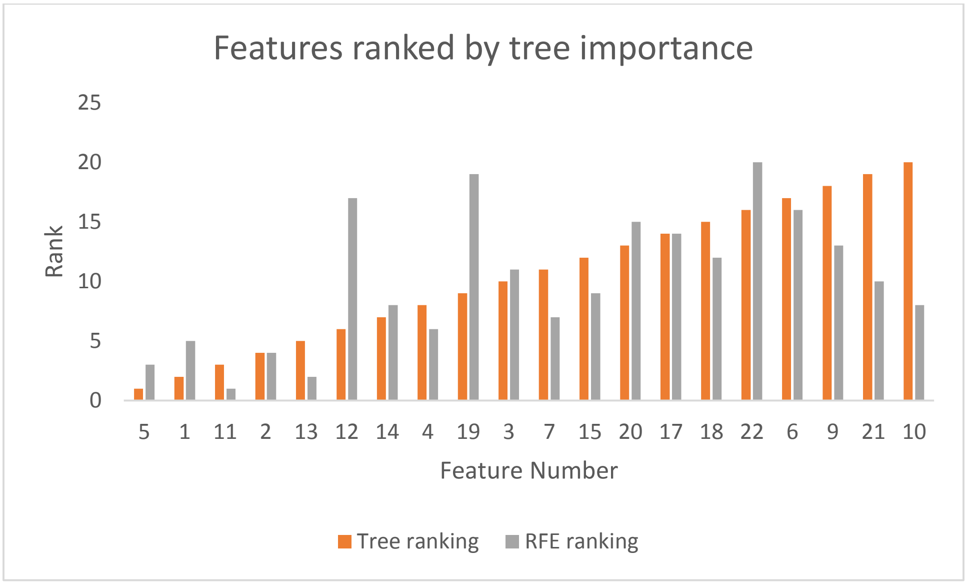

Fig 2.1: Feature importance sorted by tree ranking

3. Classifiers with default configuration

With default configuration, we have the following metrics for techniques with top 4 accuracies

Technique	Test data Accuracy	Time to execution (minutes)	User id with Best Precision	User id with least Precision
svm - RBF kernel	76.83	196	3	8
Random Forest	94.04	11	3	6
Decision tree - Adaboost	95.8	40	2	6
K nearest neighbors	72.02	132	5	8

As we can observe random forest and boosted decision tree have the best accuracy among the tested techniques. Below chart shows accuracy by user id which is consistent with overall accuracy where boosted decision tree performs better for all the users. However, in terms of time for execution, random forest performs better.

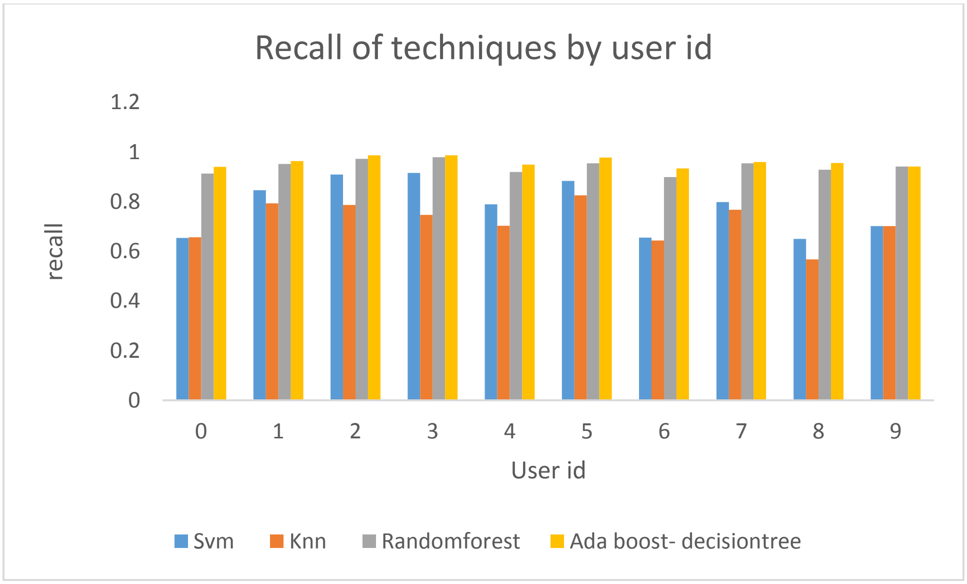

Fig 3.1 Recall of techniques by user id

4. Parameter tuning and cross validation

For the selected techniques Random forest and boosted decision tree, I performed cross validation and tuned the parameters. We perform a 3-fold cross validation using grid search in python.

For random forest: we arrive at the following best set of parameters. We stop cross validation, when the selected best parameter is not at the ends of the grid parameter value. For example, iterated values for min_samples_split are 1,2,4 we arrive at 2 as the best value which is neither at the start(which is 1) or at the end (which is 4) .

This is an expensive operation, execution time ~126 minutes

Mean validation score: 0.865 (std: 0.013)

Parameters	Values
Criterion	Gini
max_features	Auto
min_samples_leaf	4
min_samples_split	2
Bootstrap	FALSE

For boosted decision tree

Its highly expensive operation for boosted tree as the trees are built one after another. Considering there are 10 classes, with an increase in one parameter value to iterate, number of trees increases by a factor of 30 (10*3 as it is cross validation). I performed cross validation on a sample dataset form the original.

Higher this number better, however due to computational constraints I am limiting this to 30. Also, for the decision tree I am using the cross validated parameters obtained for random forest tree. Hence, parameters for boosted tree are as follows

Mean validation score: 0.933 (std: 0.002), execution time 23 minutes

Parameters	**values**
Algorithm	SAMME.R
min_samples_leaf	4
max_depth	4

5. Building Final tuned models

Based on the tuning parameter obtained we train our final models. We obtain the following metrics for the two alternatives

Technique	**Test data Accuracy**	**CV Accuracy**	**Recall**	**Precision**	**Time for execution**	**userid best recall**	**userid least recall**
RandomForest Tuned	95.2	86.5	95.3	95.4	1	2	9
AdaBoostTuned	94.6	93.3	95.02	94.9	13	2	9

Both the models perform similarly on test data in terms of accuracy and precision. However, RandomForest is much inexpensive to train and perform predictions.

Confusion matrix – Random forest

	Actual-0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Precision**
Predicted -0	7299	179	10	11	84	37	97	8	21	113	0.929
1	55	6687	39	3	27	62	31	3	3	20	0.965
2	10	59	4724	3	1	15	2	0	9	8	0.978
3	0	0	1	5878	6	0	17	10	18	40	0.985
4	32	26	1	13	7485	5	102	22	10	204	0.947
5	29	148	18	9	11	6651	35	12	4	13	0.960
6	58	20	0	102	130	10	7066	57	29	206	0.920
7	11	9	0	25	43	7	146	7657	20	80	0.957
8	25	5	1	27	51	2	51	27	5114	140	0.940
9	48	9	2	69	107	7	160	32	32	8244	0.946
Recall	0.965	0.93	0.98	0.95	0.94	0.97	0.91	0.97	0.97	0.90	

Confusion matrix – Adaboost

	Actual-0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Precision**
Predicted -0	7028	152	9	2	125	22	123	19	25	130	0.920
1	113	6617	17	0	24	81	34	7	0	21	0.957
2	16	30	4685	0	0	6	2	2	0	13	0.985
3	7	0	0	5891	8	3	32	13	13	30	0.982
4	99	11	1	11	7425	1	127	26	45	203	0.934
5	39	77	10	1	4	6806	36	7	0	12	0.973
6	112	36	0	33	112	12	7091	107	17	211	0.917
7	24	7	0	21	28	9	173	7642	16	108	0.952
8	22	0	1	18	47	2	23	29	5272	131	0.951
9	142	14	3	42	151	4	189	48	92	8049	0.922
Recall	0.92	0.95	0.99	0.98	0.94	0.98	0.91	0.97	0.96	0.90	

6. Conclusion

i. Random forest model with the given tuned parameters will be the best suitable choice in terms of accuracy and execution cost
ii. Feat5, feat1 and feat11 are the top 3 important features for classification. High variance and presence of outliers might be a contributing reason for these features to be good classifiers.
iii. Across techniques userid 9 is known to exhibit least recall, whereas userid 2 exhibits best recall