

# Human Activity Recognition from Accelerometer Data

J. M. Yao

FRIB/NSCL, Michigan State University, East Lansing, Michigan 48824, USA

April 12, 2019



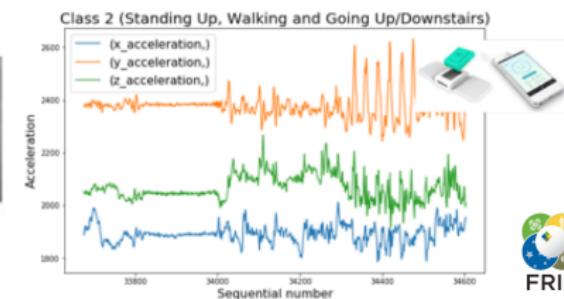
# Introduction



# Introduction to the dataset/problem

7 classes for 15 participants (**only 4 distinguishable activities**)

- 1 Working at Computer
- 2 Standing Up, Walking and Going Up/Downstairs
- 3 Standing
- 4 Walking
- 5 Going Up/Down Stairs
- 6 Walking and Talking with Someone
- 7 Talking while Standing



# Introduction to the problem and dataset

Task/Object: **multi-class classification**

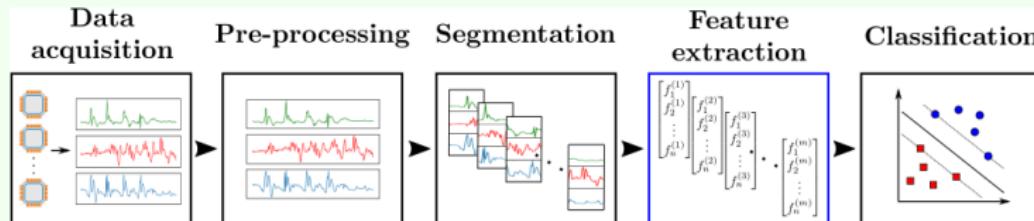
■ Input data:

Sequential number, x acceleration, y acceleration, z acceleration

■ Output:

Class label (1, 2, . . . , 7) for each activity.

## Procedure

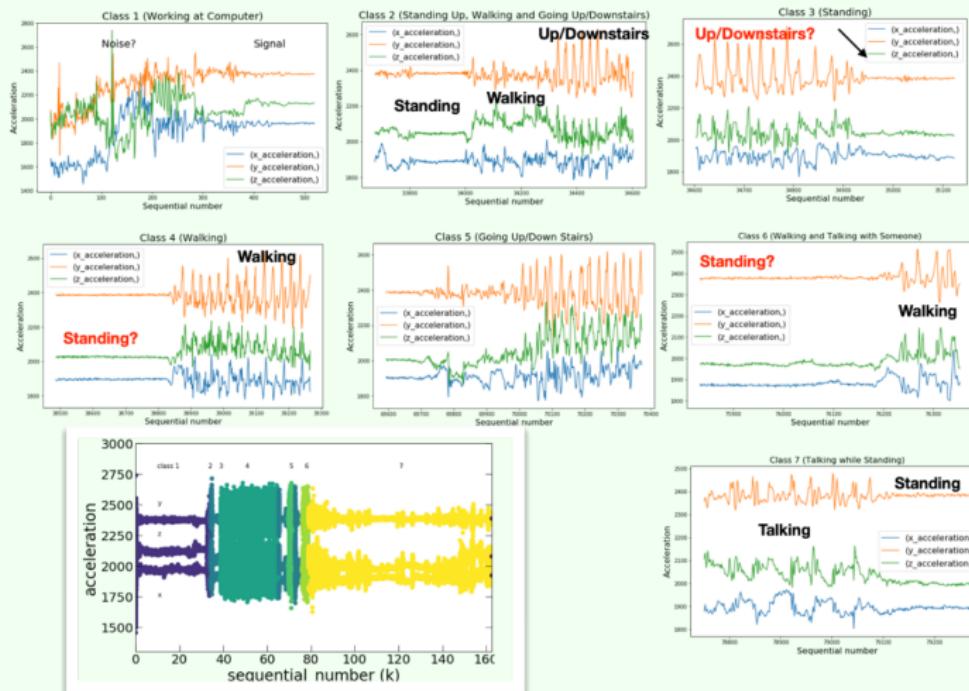


## Exploratory data analysis



# Exploratory data analysis: explore the data

## Raw data for participant 1



# Exploratory data analysis: feature extraction

## Feature extraction

- Generate samples/features using sliding temporal windows which split the raw data into segments.
- The data (x, y, z-accelerations) in each window form one sample.
- Hand-crafted features are extracted from each sample:
  - **Time domain features:** related to the distribution of the data

mean  $\mu_{i=x,y,z}$ , variance  $\sigma_i^2$ , min, max, mean absolute deviation, skewness, kurtosis, correlation  $\text{Corr}(i,j)$ , magnitude  $\sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2}$

- **Frequency domain features:** energy-weighted moments:

$$M^{(k)} = \int d\omega \cdot \omega^k \cdot F(\omega), \quad k = 1, 2, 3 \quad (1)$$

where  $F(\omega)$  is computed with Fast Fourier Transform (FFT)

$$F(\omega) = \int dt \cdot f(t) \cdot e^{-i(2\pi\omega)t}. \quad (2)$$

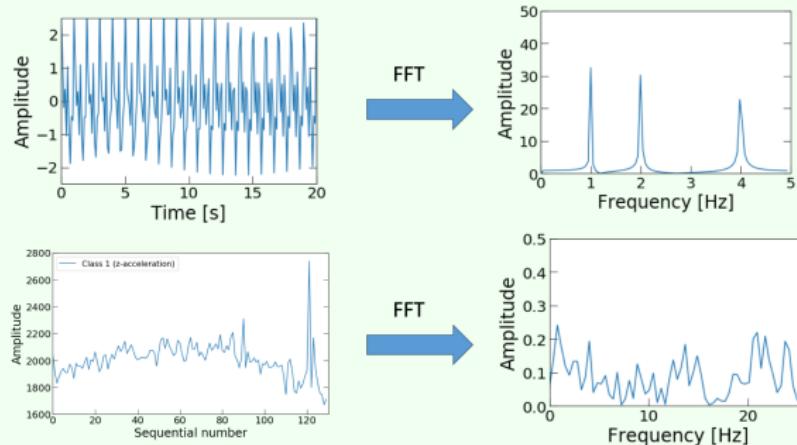


# Exploratory data analysis: feature extraction

## Fast Fourier Transformation (FFT)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad k = 0, \dots, N-1 \quad (3)$$

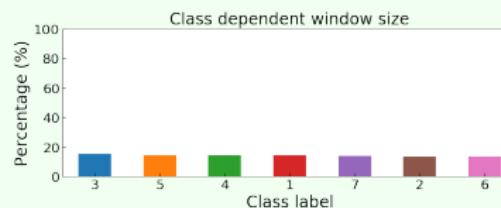
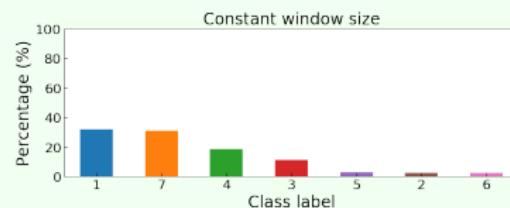
- Compute the discrete Fourier transform of a sequence
- Computation complexity  $O(n \log(n))$  (divide and conquer algorithm)



# Exploratory data analysis: check the samples

## Imbalanced data

- Multi-class classification with imbalanced data is challenge.
- Use a class dependent temporal window size.
- Risk of using window size information into training models



## Dimension of the data for training/test models

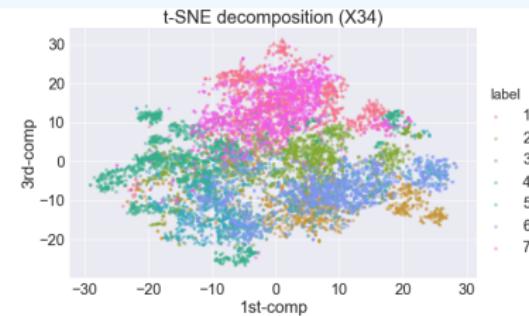
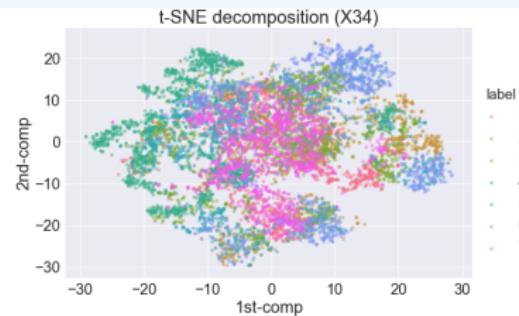
- Training data: (21684/10844, 34/25)
- Test data: (5422/2711, 34/25)

# Exploratory data analysis: explore the samples

## t-distributed Stochastic Neighbor Embedding (t-SNE)

- reduces dimensionality, trying to keep similar (dissimilar) instances close (apart)
- tool for visualizing clusters of instances

### Data with **34** features

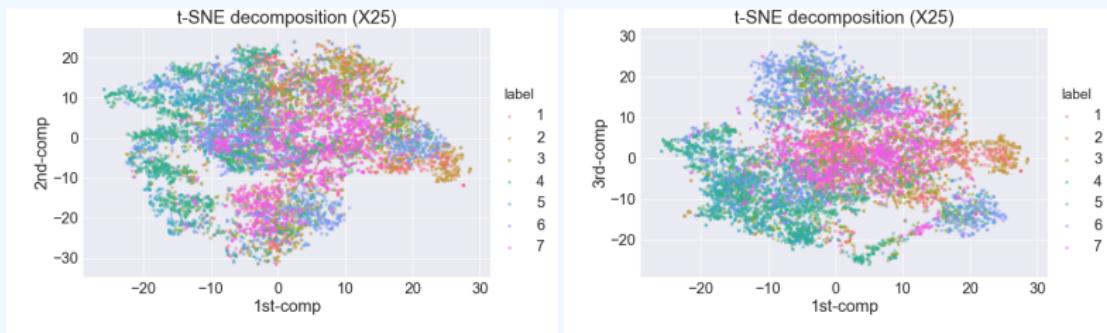


# Exploratory data analysis: explore the samples

## t-distributed Stochastic Neighbor Embedding (t-SNE)

- reduces dimensionality, trying to keep similar (dissimilar) instances close (apart)
- tool for visualizing clusters of instances

### Data with 25 features (without frequency domain features)



## Models

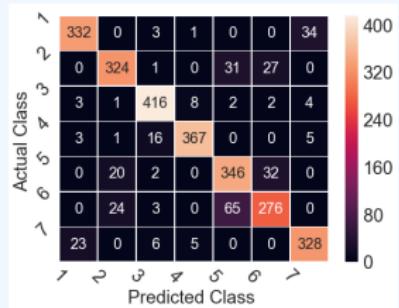


# Models

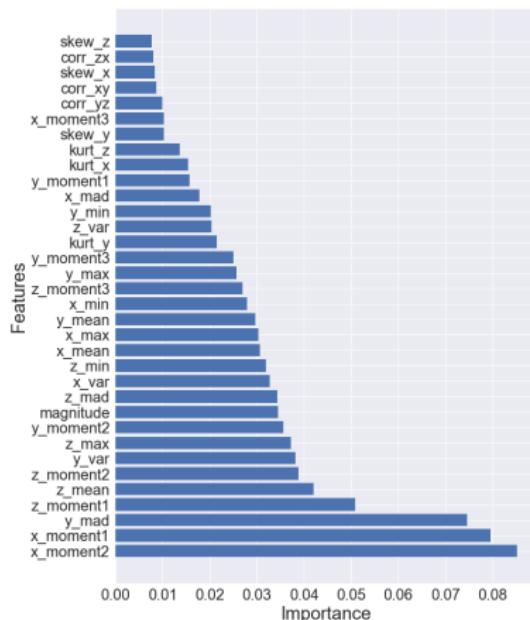
## Random Forest

- non-parametric model
- **provide importance of features**
- prone to overfitting
- regularization and monitoring train/test scores

## Confusion matrix (Test Data)



## Training the model with default parameters

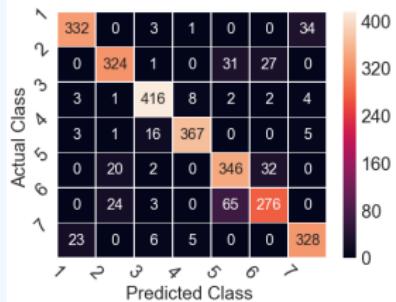


# Models

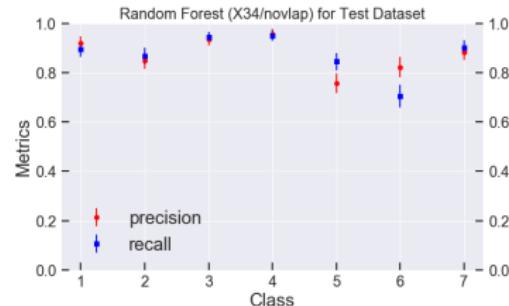
## Random Forest

- non-parametric model
- provide importance of features
- **prone to overfitting**
- regularization and monitoring train/test scores

## Confusion matrix (Test Data)



## Training the model with default parameters



# Models

## Random Forest

- non-parametric model
- provide importance of features
- prone to overfitting
- **regularization and monitoring train/test scores**

## GridSearch for the parameters

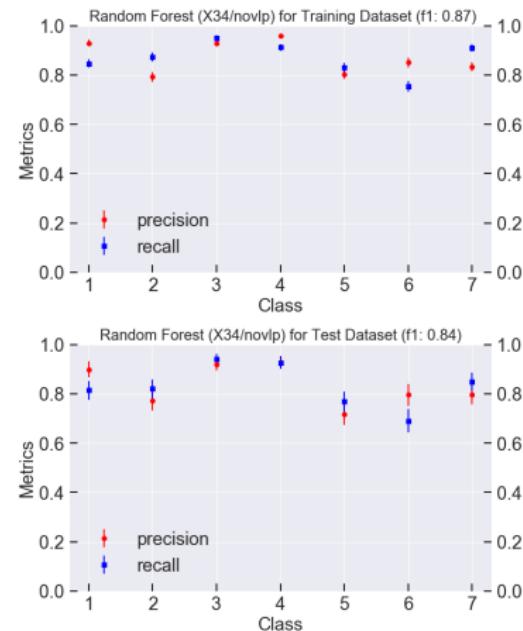
Introduce a new metric to take care of overfitting:

$$F_1 \equiv f_1(\text{test}) - 1.5|f_1(\text{train}) - f_1(\text{test})|$$

where the  $f_1$  measure is defined as

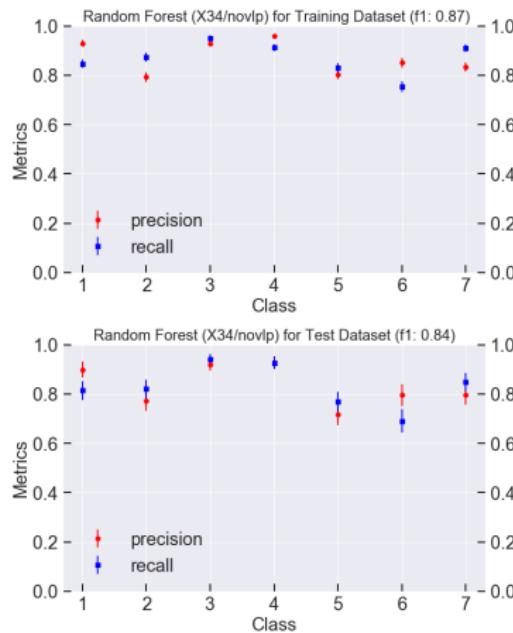
$$f_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \quad (4)$$

## Training the model with GridSearch

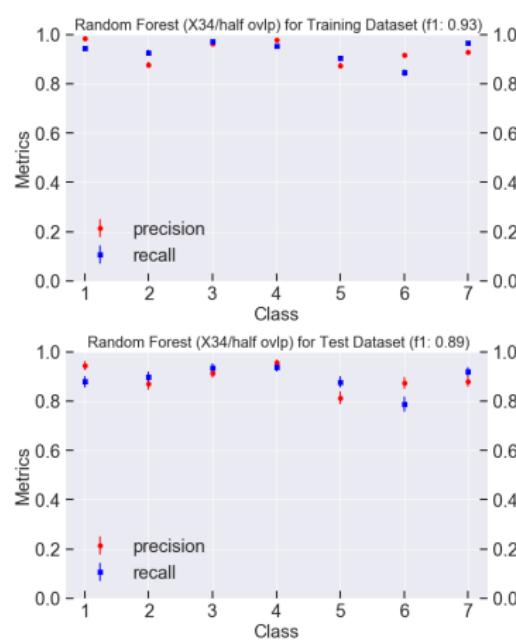


# Models: overlapping windows or not

## Non-overlapping windows

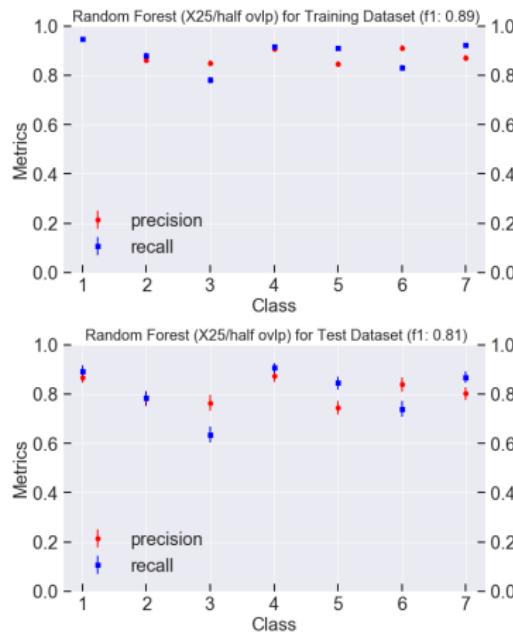


## Half-overlapping windows

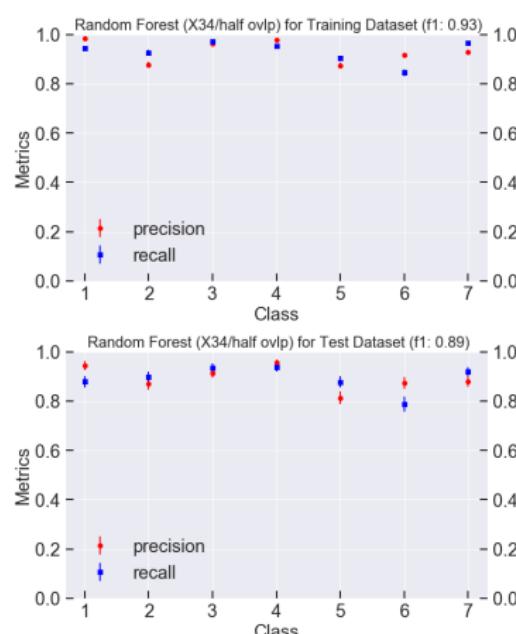


# Models: with frequency domain features or not

w/o frequency domain features

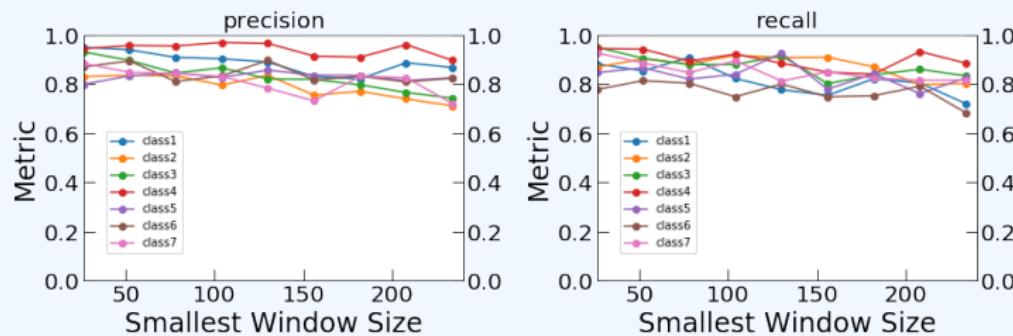


w/ frequency domain features



## Models: window size effect

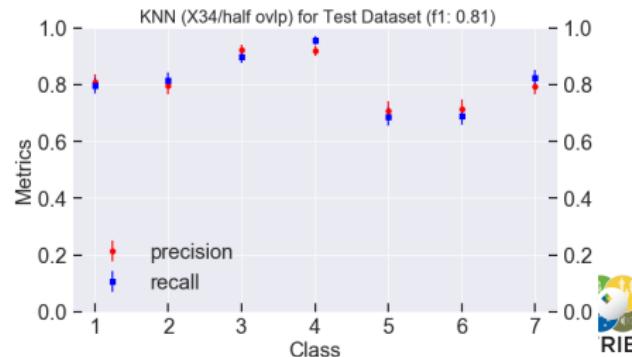
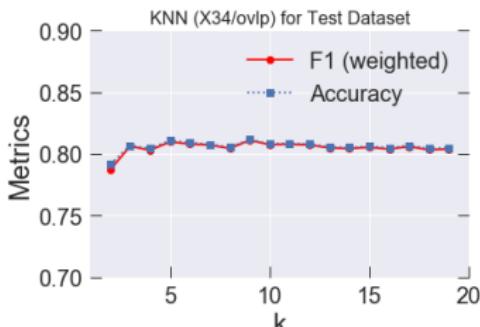
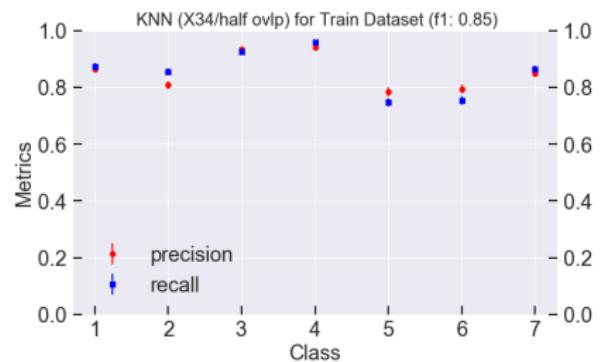
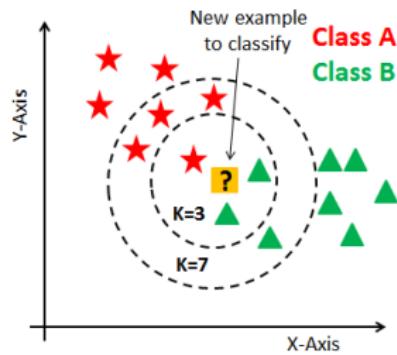
Precision/Recall for test data in the Random Forest



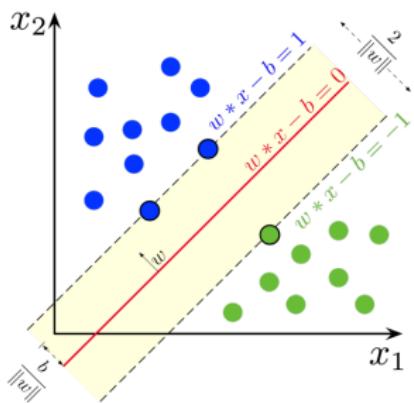
- Globally, the score is slightly decreasing with the window size.



# Models: KNeighborsClassifier (KNN)



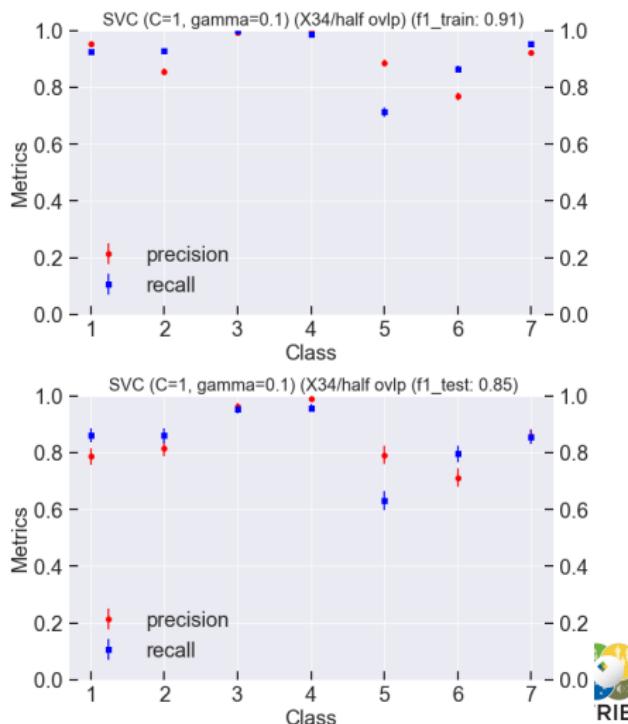
# Models: Support Vector Machine Classification (SVC)



- kernel: radial basis function

$$K(X_i, X_j) = \exp[-\gamma(X_i - X_j)^2] \quad (5)$$

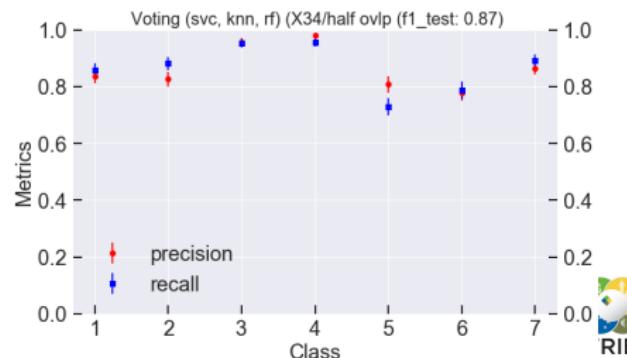
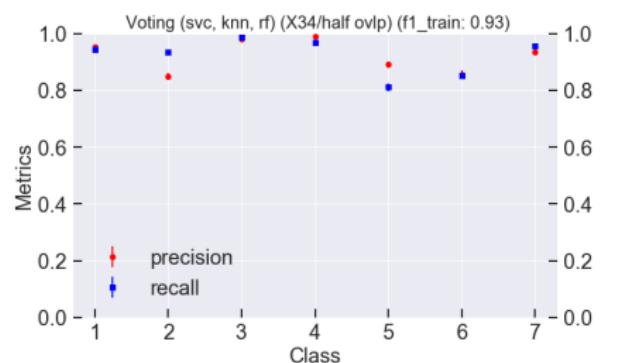
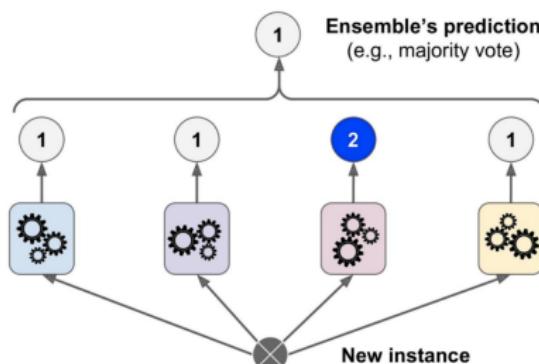
- decrease  $\gamma$  to prevent overfitting
- decrease  $C$  to prevent overfitting



# Models: Ensemble methods

## Voting

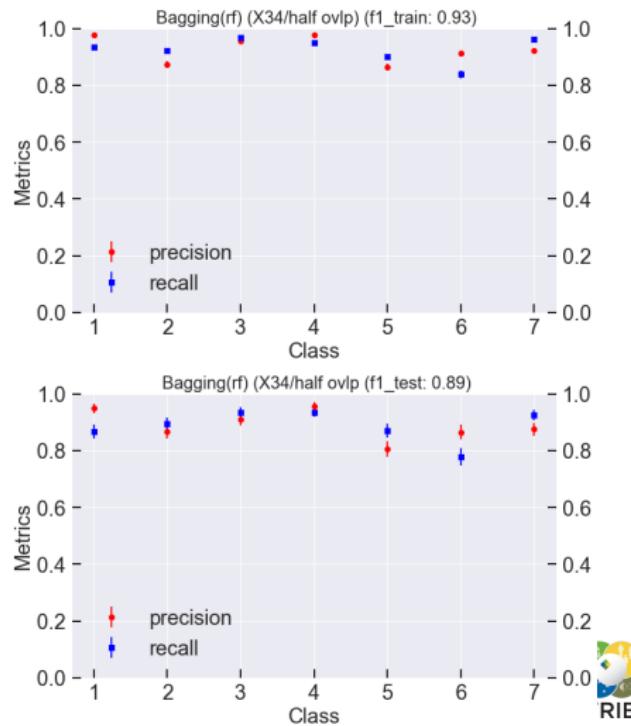
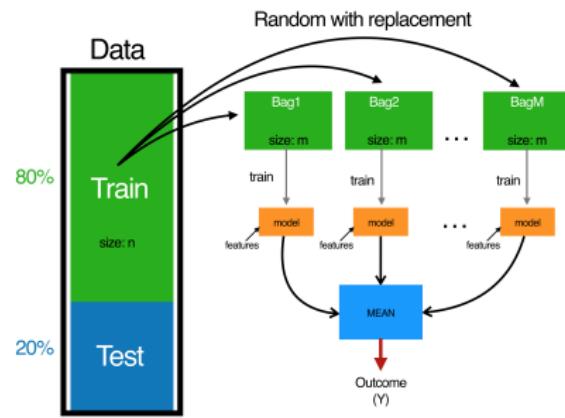
- Use the pre-trained (optimized) SVC, KNN, and Random Forest algorithms
- Aggregate the predictions of each classifier and predict the class that gets the most votes
- Generally, better than each of the three algorithms



# Models: Ensemble methods

## Bagging

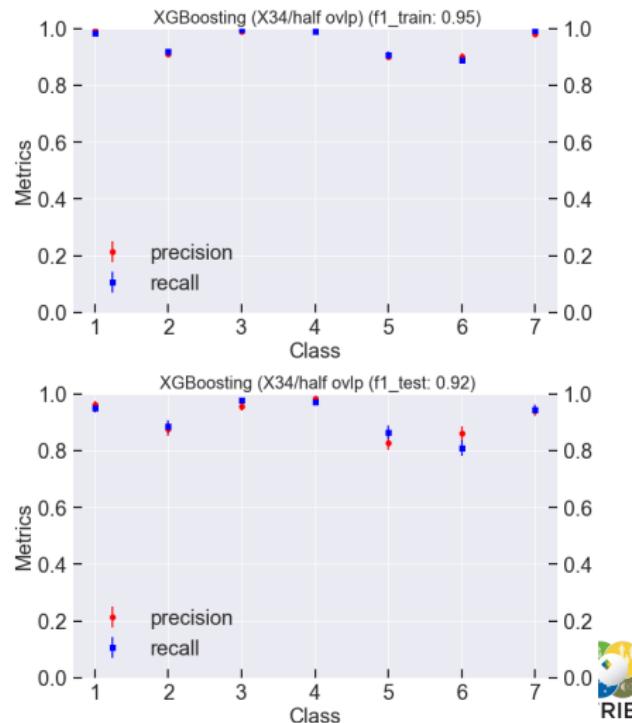
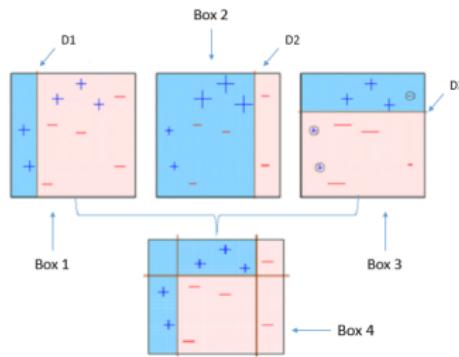
- Use the pre-trained Random Forest classifier as the base classifier
- Fit the base classifier to random 20 subsets of the original dataset (reduce variance)



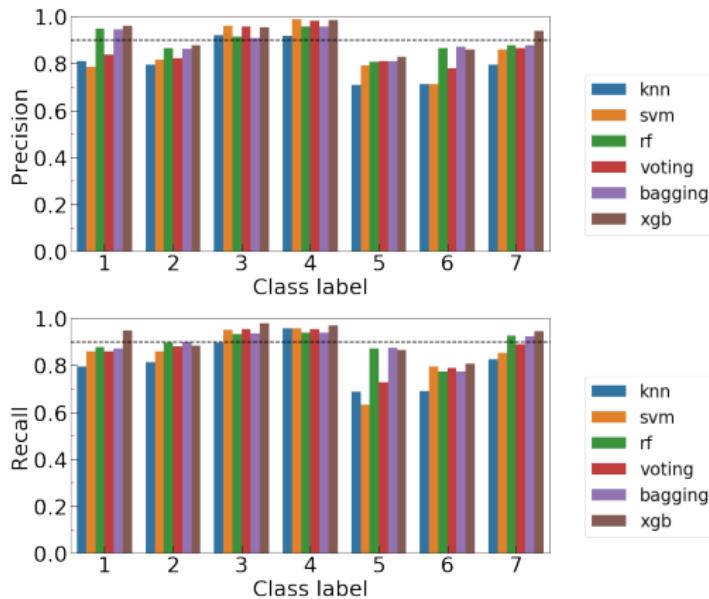
# Models: Ensemble methods

## XGBoost (eXtreme Gradient Boosting)

- Gradient boosted decision trees
- "state-of-the-art" machine learning algorithm for structured or tabular datasets on classification and regression (faster, wide variety of tuning parameters, better performance)

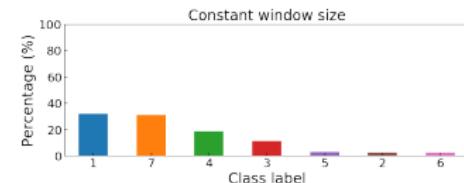


# Models: comparison



## Activities

- 1 Working at Computer
- 2 Standing Up, Walking and Going Up/Downstairs
- 3 Standing
- 4 Walking
- 5 Going Up/Down Stairs
- 6 Walking and Talking with Someone
- 7 Talking while Standing



## Summary and outlook



# Summary and outlook

## Summary

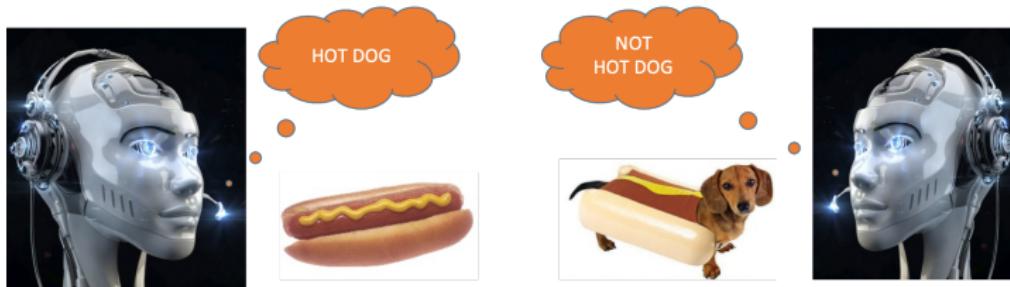
- Frequency domain features are very important
- Using non-overlapping and half-overlapping sliding windows have a significant influence (leaking test data into the training data, larger sample size?)
- Increase in window size globally decreases the performance (sample size effect?)
- Balanced data are important in multi-class classification to have a similar performance on each class.
- XGBoost turns out to have the best performance among machine learning models

## Outlook

- Collect more data for the class 2, 5, and 6
- Collect some information other than the acceleration along 3 directions (angles?)
- More detailed fine tunings on the model parameters
- Deep learning models (CNN/RNN) are worth to try when the dataset is large enough



# Acknowledgement



Special thanks to Alexander Tolpygo from SFL Scientific for providing all the data for the project.

Thank you all for your attention !

## Code Example #1

### XGBoost

```
1 def xgboost_fit_predict(X_train,y_train,X_test,y_test):
2     xgb_clf = xgboost.XGBClassifier(n_estimators=1000, random_state=42)
3     xgb_clf = xgb_reg.fit(X_train, y_train, eval_metric=["mlogloss"],
4                           eval_set=[[X_train, y_train],[X_test, y_test]],
5                           verbose=100,
6                           early_stopping_rounds=2)
7
8     y_pred4test      = xgb_clf.predict(X_test)
9     y_pred4train    = xgb_clf.predict(X_train)
10
11    xgb_clf_best_cm4test = metrics.confusion_matrix(y_test, y_pred4test)
12    xgb_clf_best_cm4train = metrics.confusion_matrix(y_train, y_pred4train)
13
14    print("F1 score for Test: {}".format(f1_score(y_test,y_pred4test, average
= 'weighted')))
15    print("F1 score for Train: {}".format(f1_score(y_train,y_pred4train,
average='weighted')))
16
17    return xgb_clf
18
```



## XGBoost: mirror in train and test

$$J(\Theta) = -\frac{1}{M} \sum_{m=1}^M \sum_{k=1}^K \left[ y_k^{(m)} \log \hat{p}_k^{(m)} \right] \quad (6)$$

