Time Series Classification Based on Smartphone Data

Brandon Archbald, Emily Nason, & Nicholas Tjandra

Overview

- The Data
- ARIMA and SARIMAX
- State-based Models
- Combining ARIMA & Classification

The Data: Human Activity Recognition Using Smartphones

From: UC Irvine Machine Learning Repository

Type: Multivariate, Time-Series

Each person performed activities while wearing a smartphone (Samsung Galaxy S II) on the waist. Features were captured via the device's embedded accelerometer and gyroscope. Activities were labelled manually from video recordings.

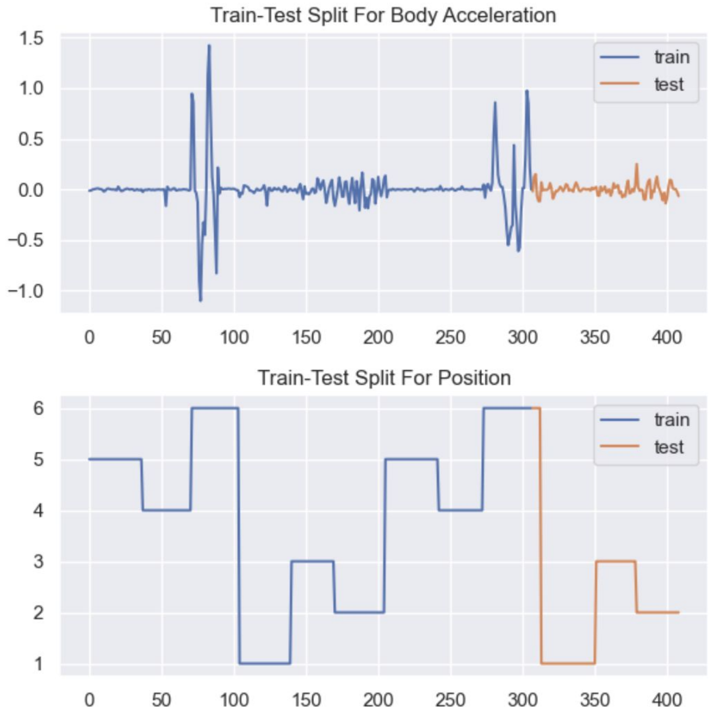
- 30 subjects
- 561 features
- 6 activities
  - WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING
-

ARIMA and SARIMAX

Our first pass at modeling the dataset

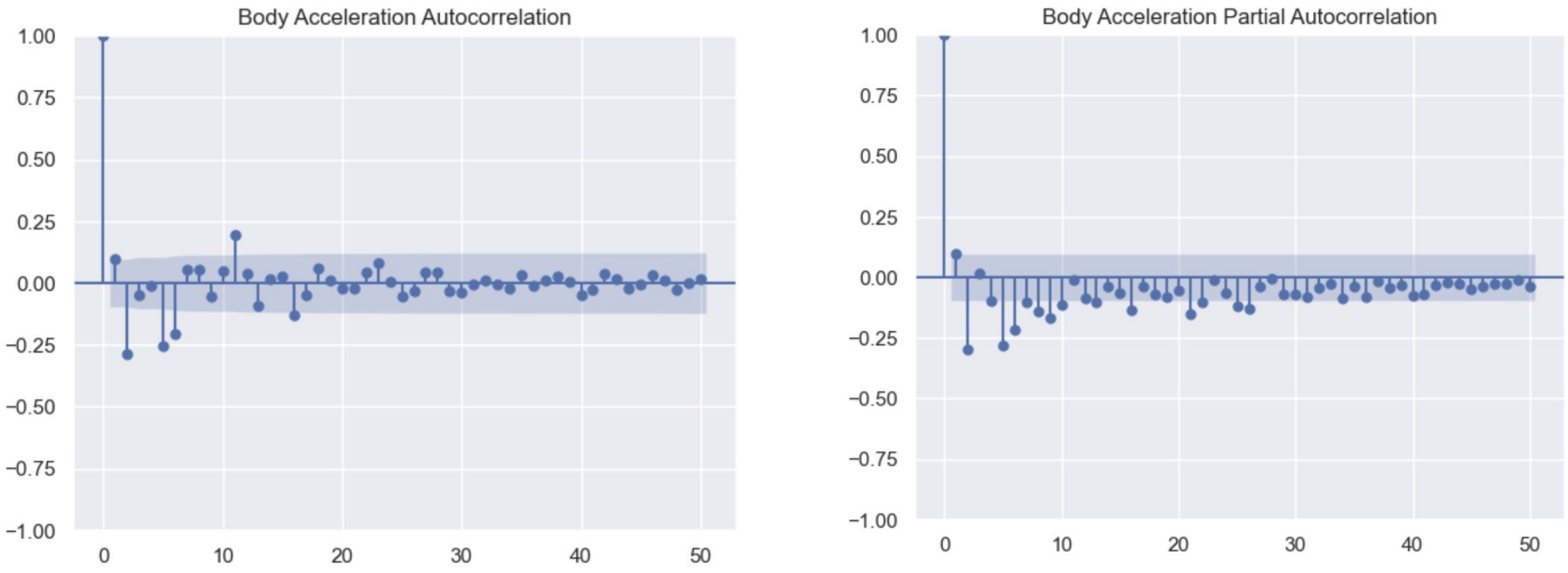
Data Preparation

- Chose subject with the largest number of time steps
- Selected body acceleration and position as predicted variables
- Performed PCA on three acceleration features (x, y, z) into one
- Train test split 75-25



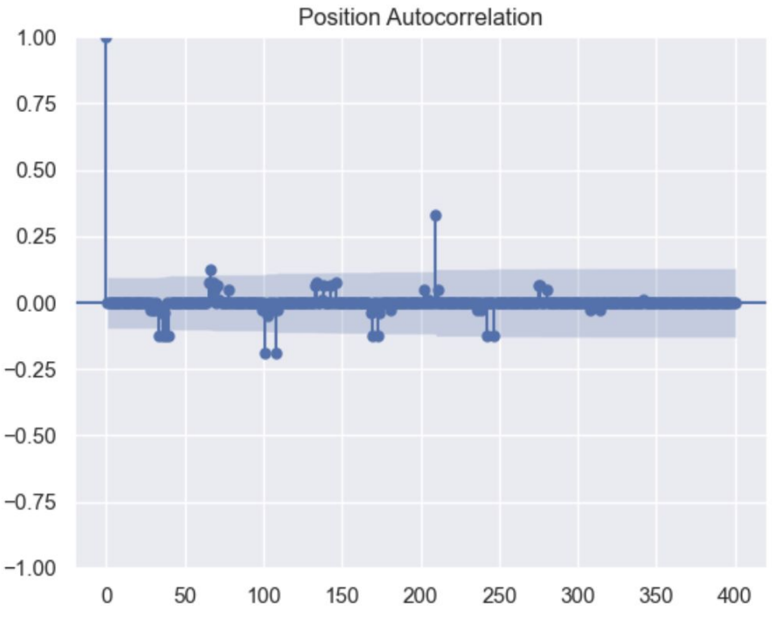
Choosing MA and AR parameters

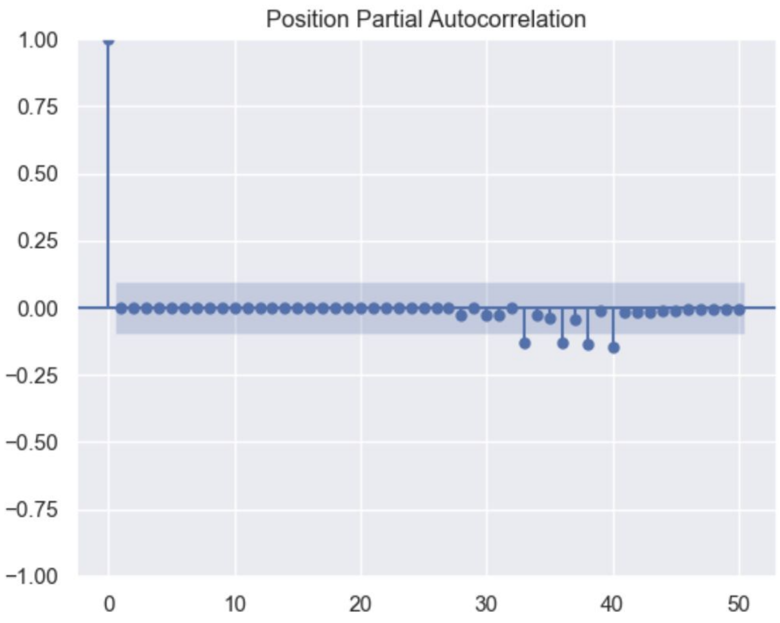
Appropriate AR and MA values were 1 for both



Choosing MA and AR parameters

Appropriate AR and MA values were 1 for both

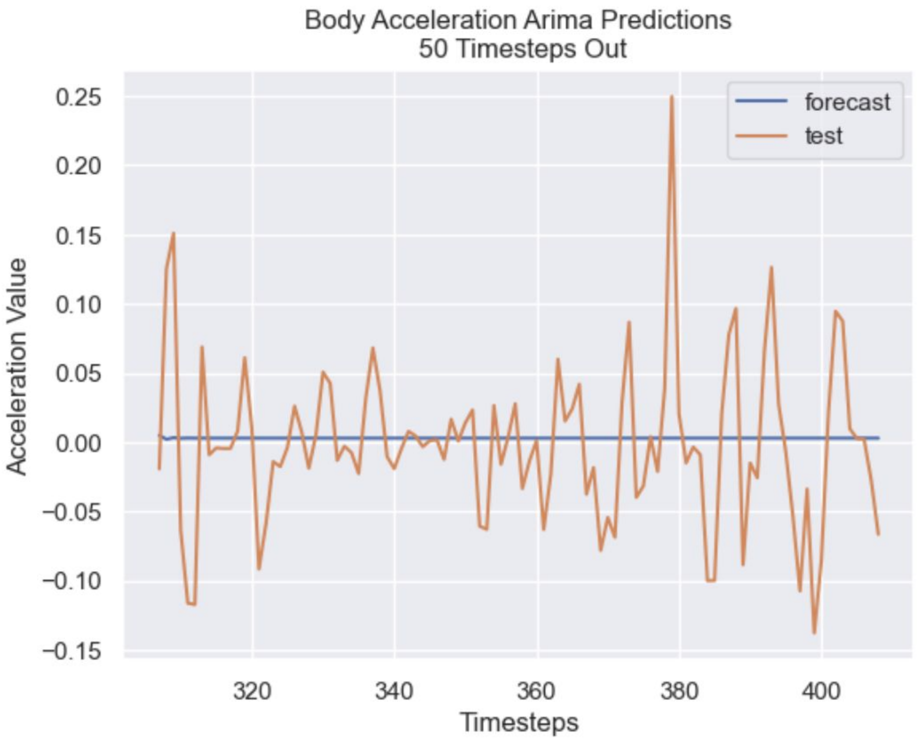




ARIMA - Body Acceleration

Test set RMSE: 0.059

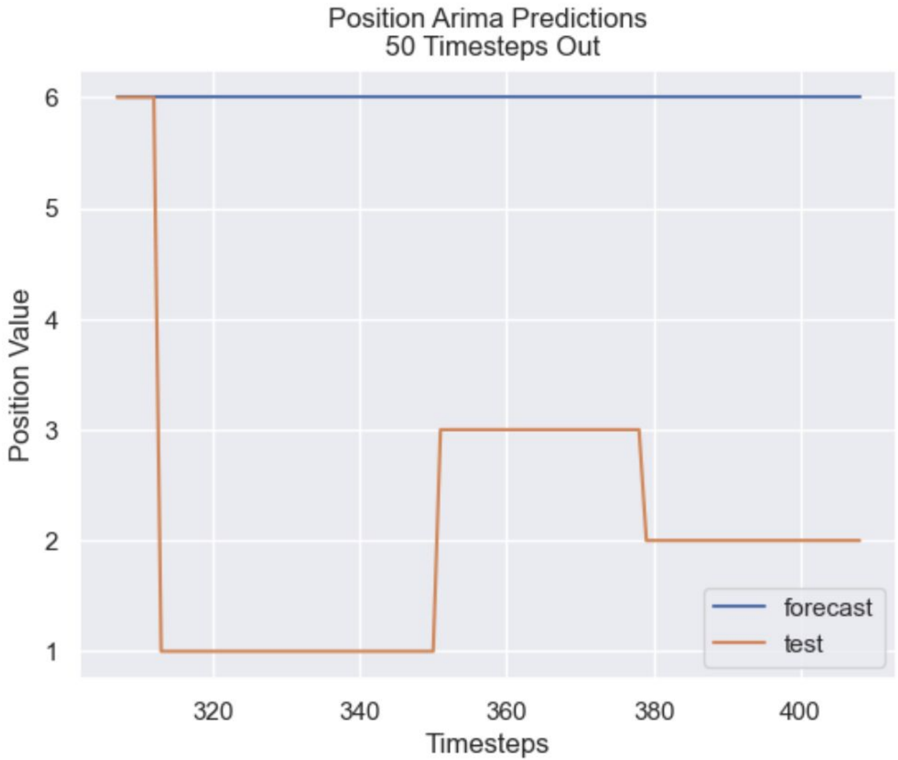
Model AIC: -177



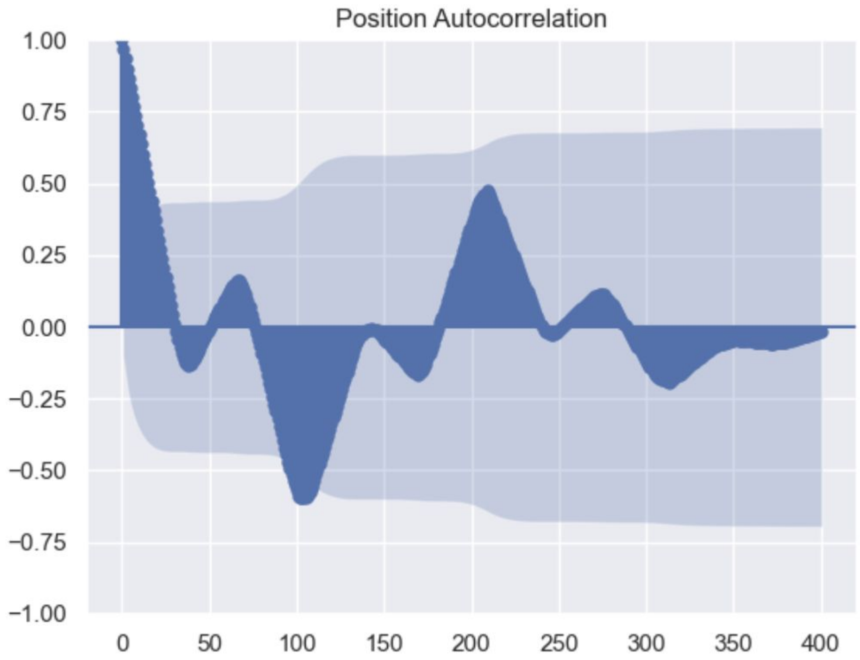
ARIMA - Position

Test set RMSE: 2.475

Model AIC: 313



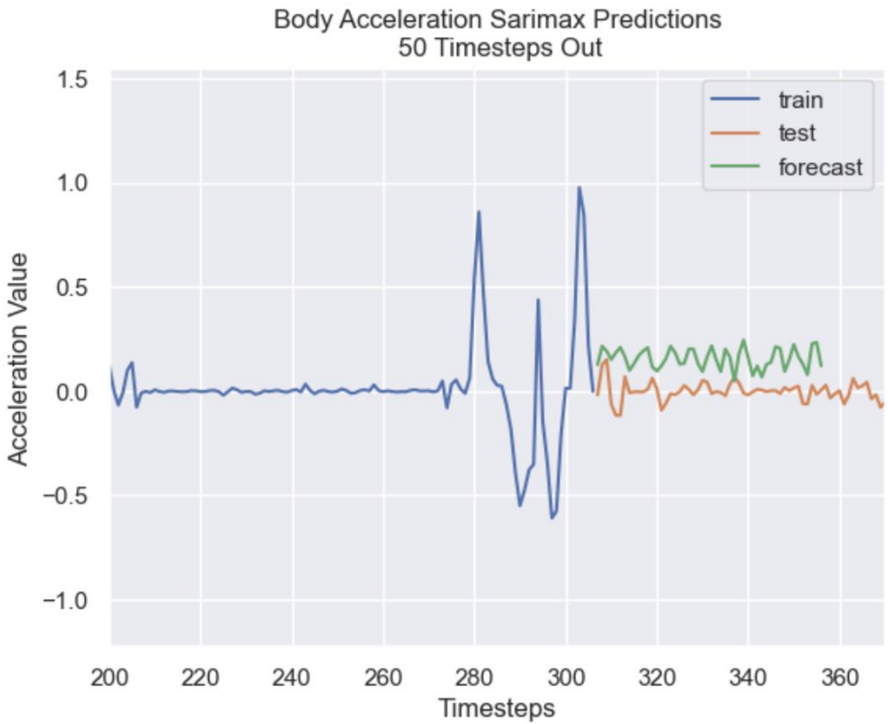
Observed Seasonality



SARIMAX - Body Acceleration

Test set RMSE: 0.17

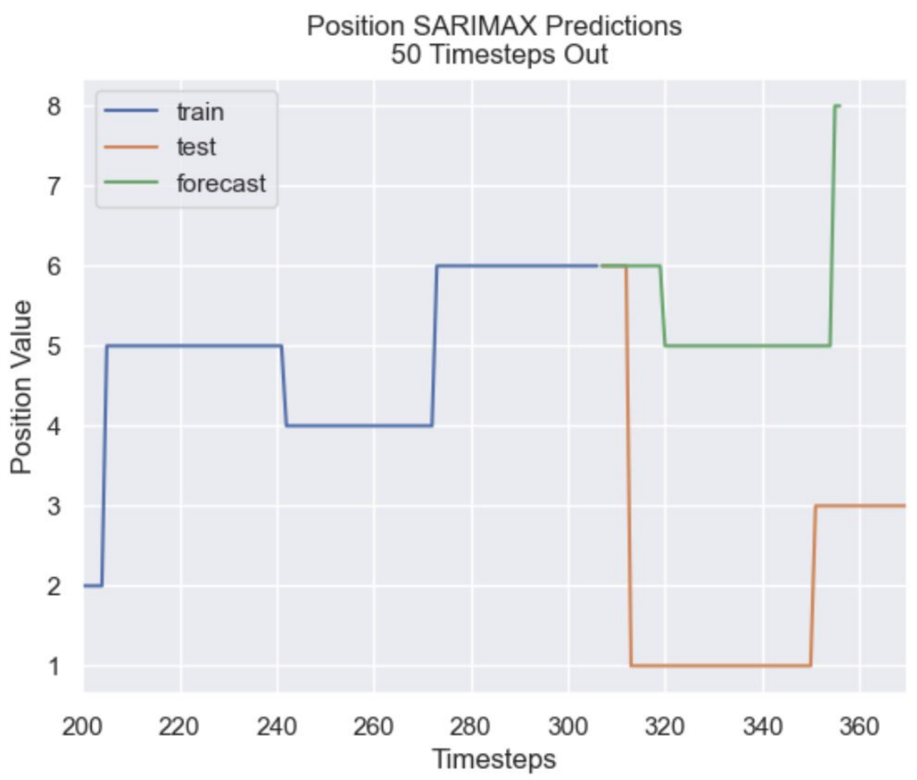
Model AIC: 8.7



SARIMAX - Position

Test set RMSE: 4.08

Model AIC: -77

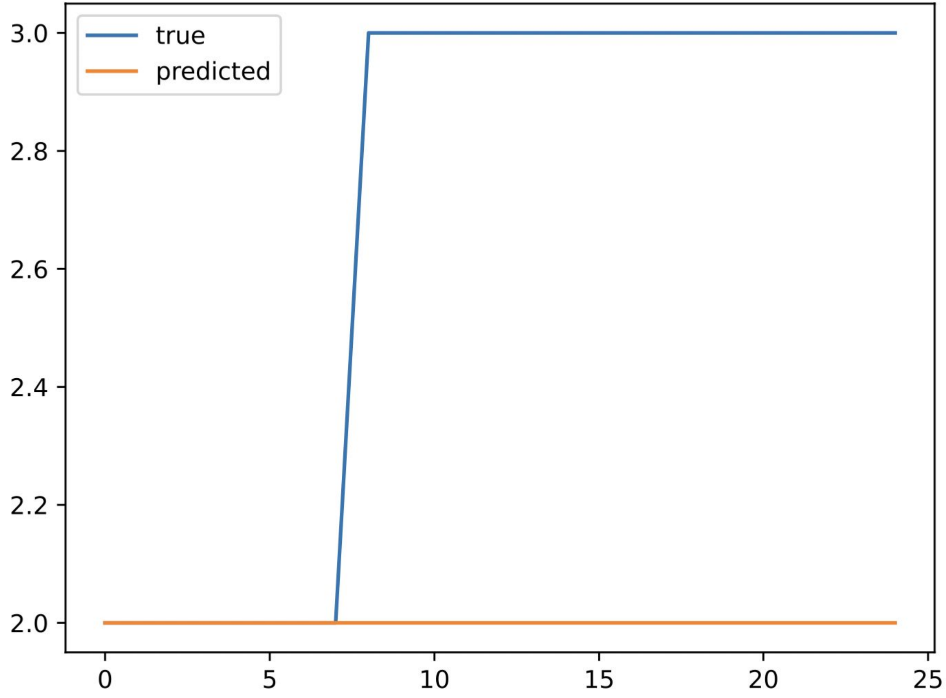


State-Based Models

Markov Chain Model: Method

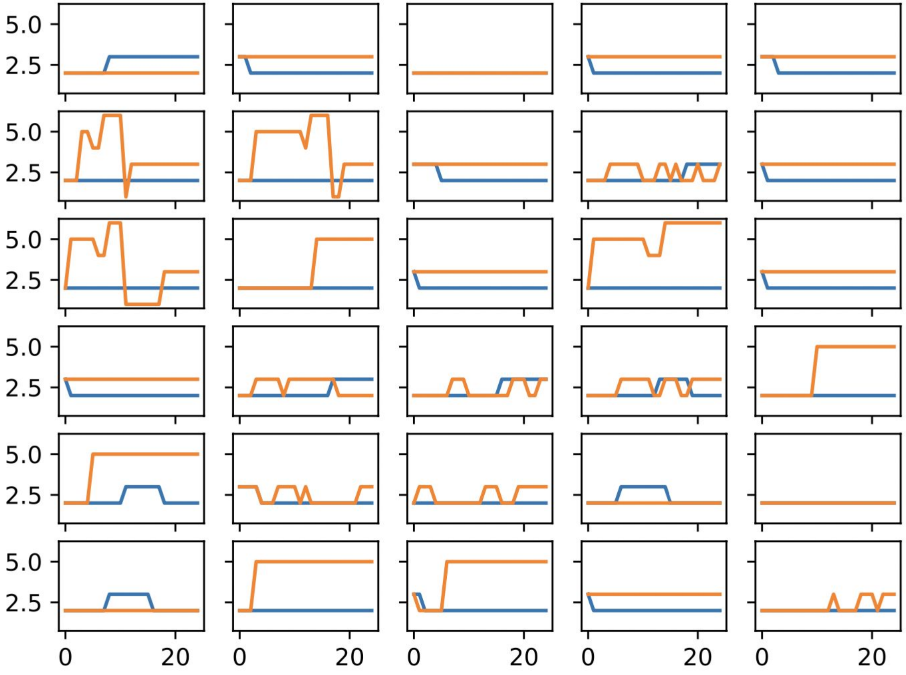
For each subject:

1. Create a transition matrix based on the training data of states
2. Run a simulation of last 25 states, based on last state in training data
3. Compare with test data

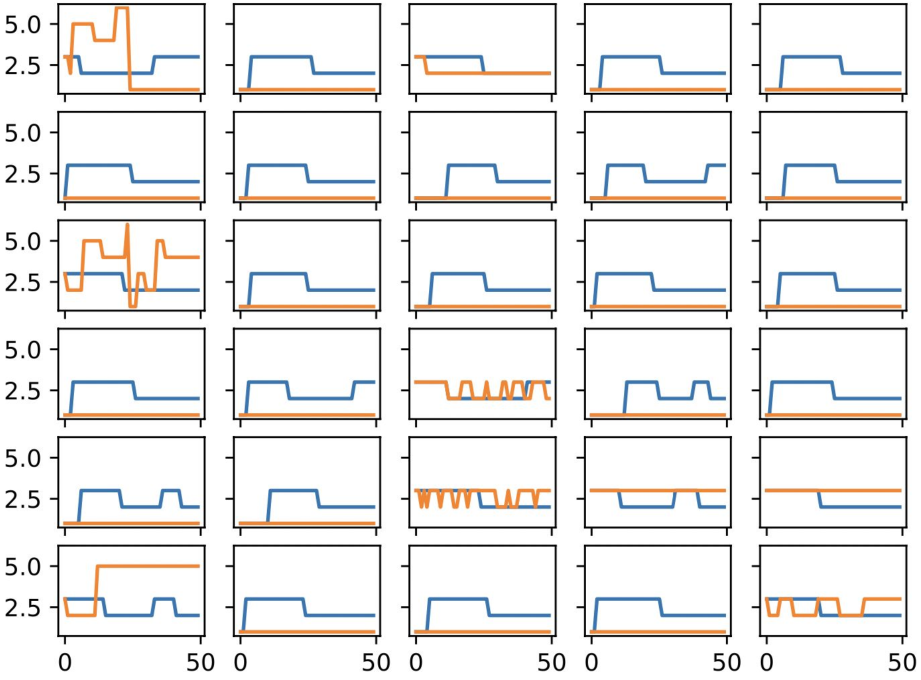


Markov Chain Model: Results

25 steps: average accuracy of 33%



50 steps: average accuracy of 17%



Markov Chain Model: Remarks

Issues

- Markov chain assumes that the next step only depends on the most recent step
- This was only one simulation
- Each model was trained on very little data
- Was each state reflected in each training set?
- Uncertainty in how data was obtained
    - Were state transitions assigned or chosen?

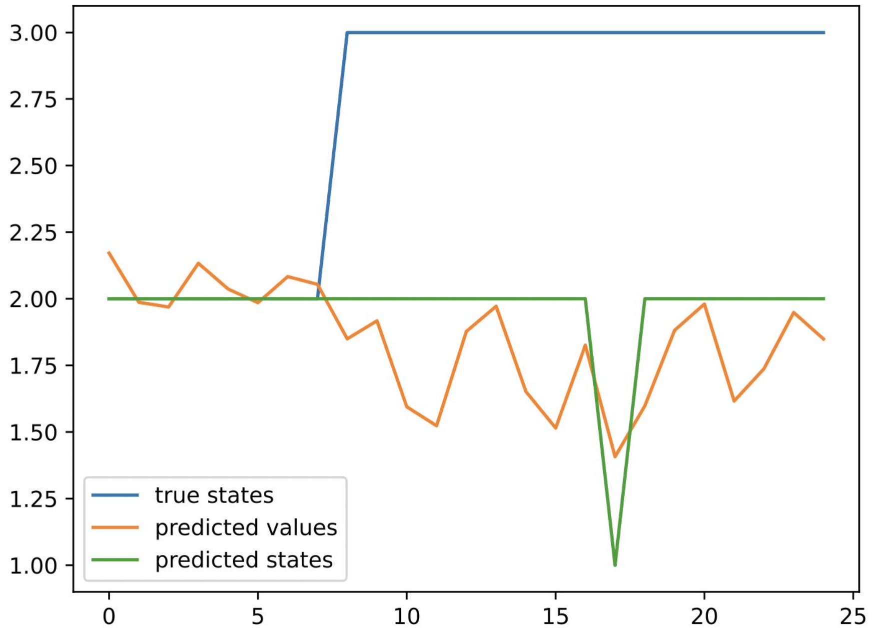
Possible Solutions

- **Second-order or higher markov chain model**
- **Multiple simulations, average accuracy per subject**
- **Ensure training data includes all states**
- **Combine all subjects into one dataset**
- **Find alternative dataset**

ARIMA Model: Method

For each subject:

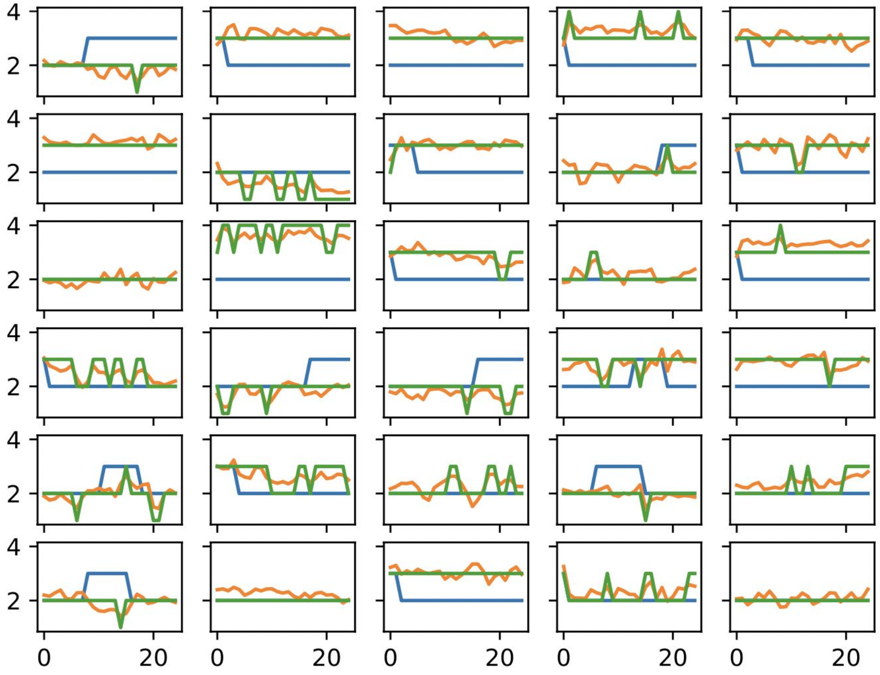
1. Use ARIMA to predict the states directly using training data
2. Auto-ARIMA function from pmdarima package for optimization of p,d,q
3. Round final predictions to nearest integer
4. Compare with test data



ARIMA Model: Results

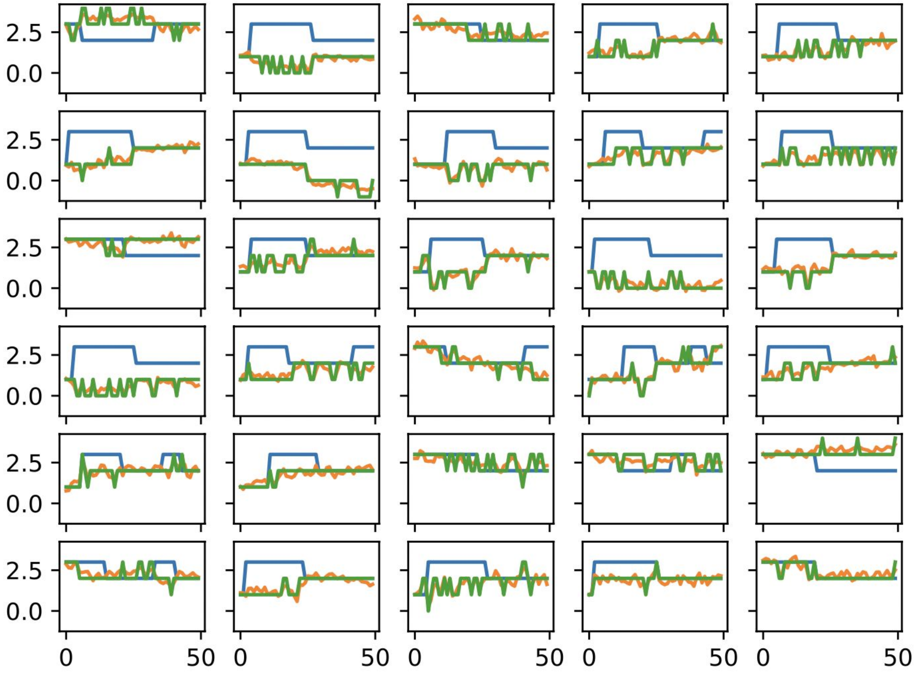
25 steps:

Average Accuracy = 43%; Average AIC = 32



50 steps:

Average accuracy of 47%; Average AIC: -10



ARIMA Model: Remarks

Issues

- ARIMA meant to predict continuous data types
- Each model was trained on very little data
- Was each state reflected in each training set?
- Uncertainty in how data was obtained
  - Were state transitions assigned or chosen?

Possible Solutions

- **Use ARIMA to predict the continuous variables, then classify into activity states**
- Ensure training data includes all states
- Combine all subjects into one dataset
- Find alternative dataset

Combining ARIMA & Classification

Approach Summary

1

Use PCA to reduce dimensions of the data

2

Use ARIMA to forecast future steps for different features

3

Train a classifier to use the selected features to predict the subject's current state

4

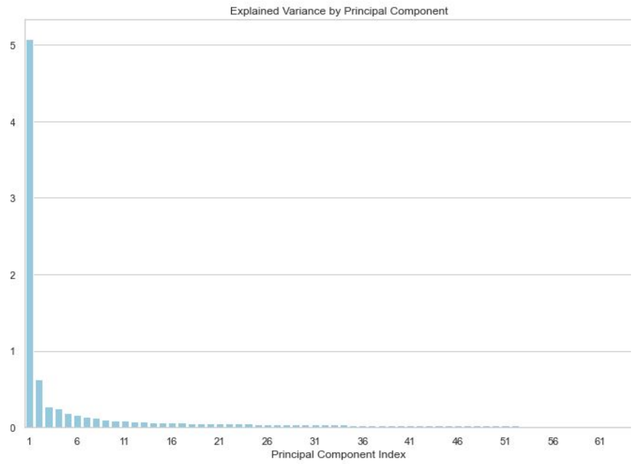
Use the classifier to predict state of a person based on forecasted data

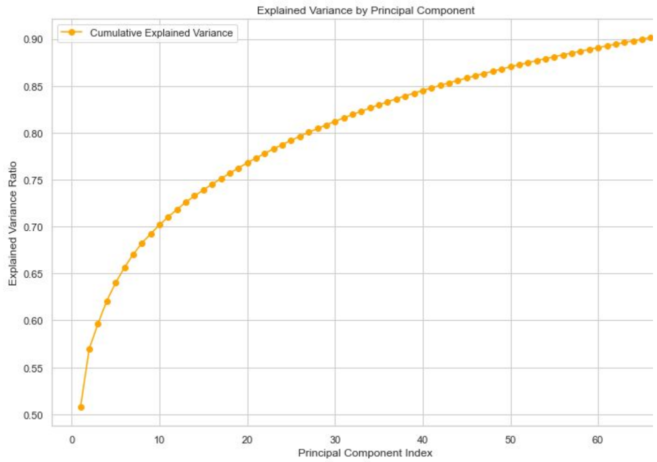
5

Evaluate the performance of combining these 2 methods to predict a person's future state

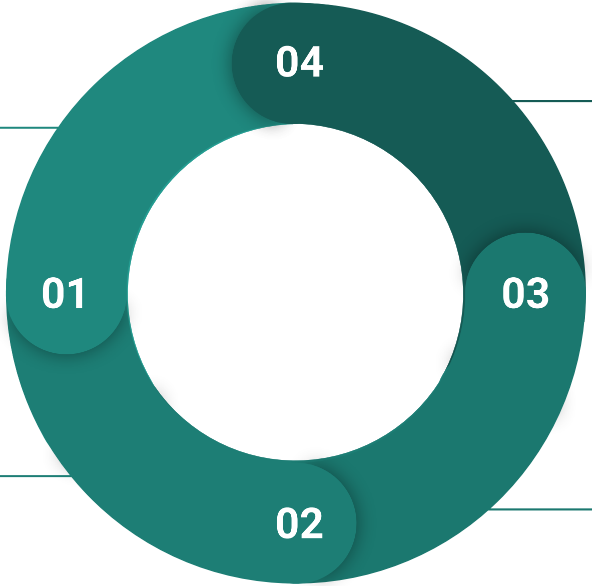
Results of the PCA

- Originally the data contained 562 variables
- Some of them contained means and non-useful information
- Used PCA to retain **90%** of the variance in the data
- This reduced the number of features we needed to model from **562 to 66**





Using ARIMA for Forecasting



Step 1

Extract a single feature (per subject) from the training set and treat it as it’s own time series

Step 2

Use the Auto ARIMA implementation from pmdarima to fit each of the features to a time series model and find it’s best parameters based on AIC

Step 3

Forecast the next N steps for each of the features in the dataset

Step 4

Compile these predictions into a forecasted data frame that mirrored the shape and dimensions of the actual test data

The Classifier

- We chose Logistic Regression (Non-Binary Implementation)

- Wanted to use a classifier that’s fairly baseline

- The goal was not to build the best classifier ever but to prove the concept that we could effectively predict future states using forecasted data

- On the actual test data the accuracy was generally >95% depending on the subject

Classification reports

- Real Test Data
  - Accuracy of 99.3%
  - Precision: 97%
  - Recall: 99%
  - F1-score: 98%

Classification Report:

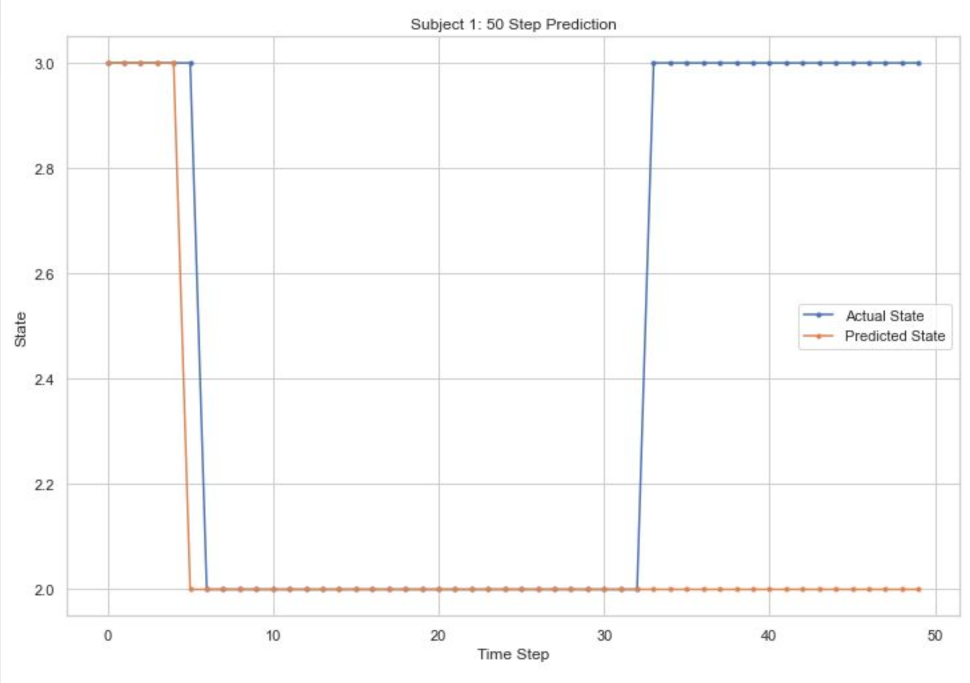
	precision	recall	f1-score	support
1	0.92	1.00	0.96	12
2	1.00	1.00	1.00	77
3	1.00	0.98	0.99	61
accuracy			0.99	150
macro avg	0.97	0.99	0.98	150
weighted avg	0.99	0.99	0.99	150

- Forecasted Test Data
  - Accuracy of 55.3%
  - Precision: 43%
  - Recall: 37%
  - F1-score: 29%

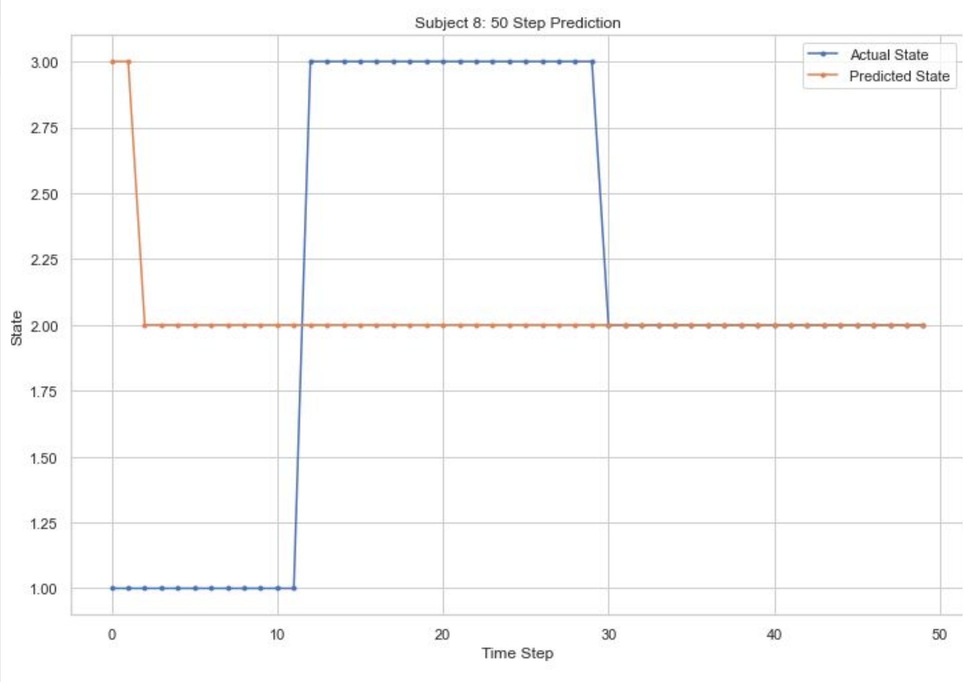
Classification Report:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	12
2	0.54	1.00	0.70	77
3	0.75	0.10	0.17	61
accuracy			0.55	150
macro avg	0.43	0.37	0.29	150
weighted avg	0.58	0.55	0.43	150

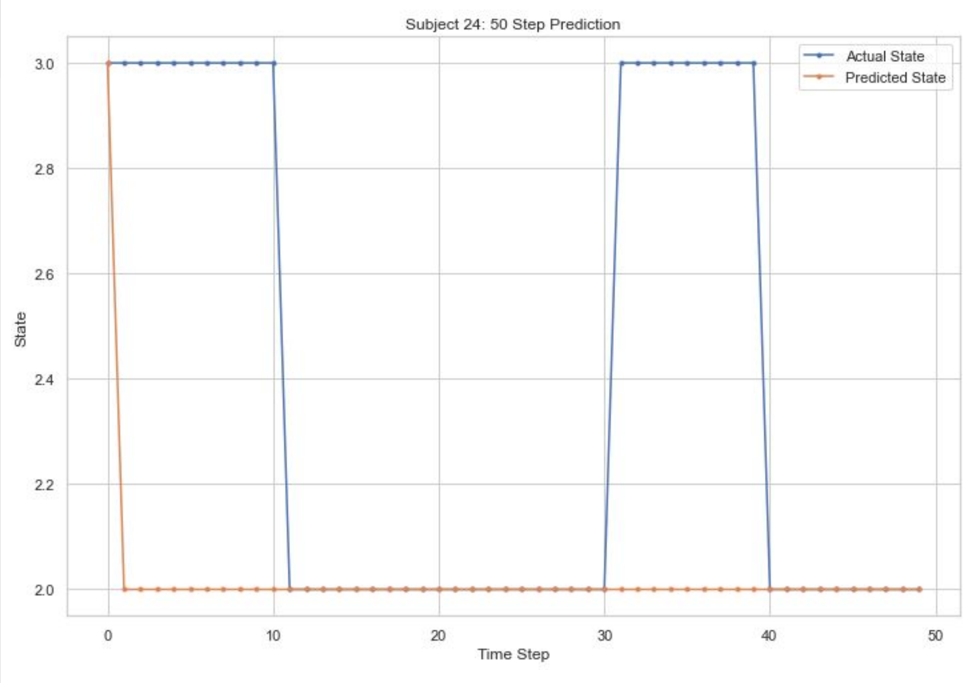
What Contributed to this Outcome?



What Contributed to this Outcome?



What Contributed to this Outcome?



What Went Wrong?

1. We treated each feature as an independent time series

2. Even small deviations from the truth could influence the meaning of the data to the classifier

3. The forecasted data seemed to fail to reflect state changes as much as the model needed it to

Possible Improvement

1. Use a moving window and Deep Learning approach to forecast the features

2. Try to account for the fact that each feature may not be independent of the other and there is some dependency

3. Try other classifiers that may be a little better at handling the small errors that the forecasting model makes

Questions?