# LINK TO GITHUB

https://github.com/RockFZ/CS542-Summer2022.git

# ANOMALY DETECTOR MODELS

## USING KEYSTROKE DYNAMICS ANALYSIS

By Fucheng Zhu

# KEYSTROKE DYNAMICS

# THE DATA

Shape: 20400 * 31

```
       subject  sessionIndex  rep  H.period  DD.period.t  UD.period.t     H.t
0         s002             1    1    0.1491       0.3979       0.2488  0.1069
1         s002             1    2    0.1111       0.3451       0.2340  0.0694
2         s002             1    3    0.1328       0.2072       0.0744  0.0731
3         s002             1    4    0.1291       0.2515       0.1224  0.1059
4         s002             1    5    0.1249       0.2317       0.1068  0.0895
...        ...           ...  ...       ...          ...          ...     ...
20395     s057             8   46    0.0884       0.0685      -0.0199  0.1095
20396     s057             8   47    0.0655       0.0630      -0.0025  0.0910
20397     s057             8   48    0.0939       0.1189       0.0250  0.1008
20398     s057             8   49    0.0923       0.1294       0.0371  0.0913
20399     s057             8   50    0.0596       0.1310       0.0714  0.0992

       DD.t.i  UD.t.i     H.i  ...     H.a  DD.a.n  UD.a.n     H.n  DD.n.l  \
0      0.1674  0.0605  0.1169  ...  0.1349  0.1484  0.0135  0.0932  0.3515
1      0.1283  0.0589  0.0908  ...  0.1412  0.2558  0.1146  0.1146  0.2642
2      0.1291  0.0560  0.0821  ...  0.1621  0.2332  0.0711  0.1172  0.2705
3      0.2495  0.1436  0.1040  ...  0.1457  0.1629  0.0172  0.0866  0.2341
4      0.1676  0.0781  0.0903  ...  0.1312  0.1582  0.0270  0.0884  0.2517
...       ...     ...     ...  ...     ...     ...     ...     ...     ...
20395  0.1290  0.0195  0.0945  ...  0.1219  0.1383  0.0164  0.0820  0.1329
20396  0.1148  0.0238  0.0916  ...  0.1008  0.0512 -0.0496  0.1037  0.0868
20397  0.1122  0.0114  0.0721  ...  0.0913  0.1169  0.0256  0.0689  0.1311
20398  0.0990  0.0077  0.0992  ...  0.0882  0.0821 -0.0061  0.0576  0.0697
20399  0.1103  0.0111  0.0998  ...  0.0969  0.0784 -0.0185  0.0790  0.1133
...
20398  0.0121  0.0979       0.1917       0.0938    0.0931
20399  0.0343  0.0807       0.1993       0.1186    0.1018

[20400 rows x 34 columns]
```

# THE DATA

- **Collecting data**
  - 51 Subjects corresponding to 51 typist
    - 8 sessions
      - 50 reps each session per person
- **Features**
  - Hold time for a key
  - keydown-keydown time for a digraph

# GOALS

- To develop models that detect anomaly imposters
- To compare different models in terms of performance metrics
- To try to reasoning those performance differences

# THE DATA

```
YTrain = D[(D['sessionIndex'] <= 4 ) & (D['subject'] == evalSubject)]
YTrain = np.array(YTrain)[:,3:]

YScore0 = D[(D['sessionIndex'] <= 4 ) & (D['subject'] == evalSubject)]
YScore0 = np.array(YScore0)[:,3:]

YScore1 = D[(D['sessionIndex'] == 1 ) & (D['subject'] != evalSubject) & (D['rep'] <= 5)]
YScore1 = np.array(YScore1)[:,3:]
```

- YTrain: first half of a subject's reps—TrainingSet
  - shape: 200 * 31
- YScore0: second half of a subject's reps—TestingSet
  - shape: 200 * 31
- YScore1: first 5 reps of all other subjects—ImposterSet (Validation)
  - shape:  250 * 31

# MODELS

- Statistic models:
  - Euclidean distance detector
  - Manhattan distance detector
  - Mahalanobis distance detector
- Classification model
  - One class SVM with linear kernel
  - One class SVM with other kernels
  - NN

# EUCLIDEAN DETECTOR

$$d(x,\ y)\ =\ \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

# MANHATTAN DETECTOR

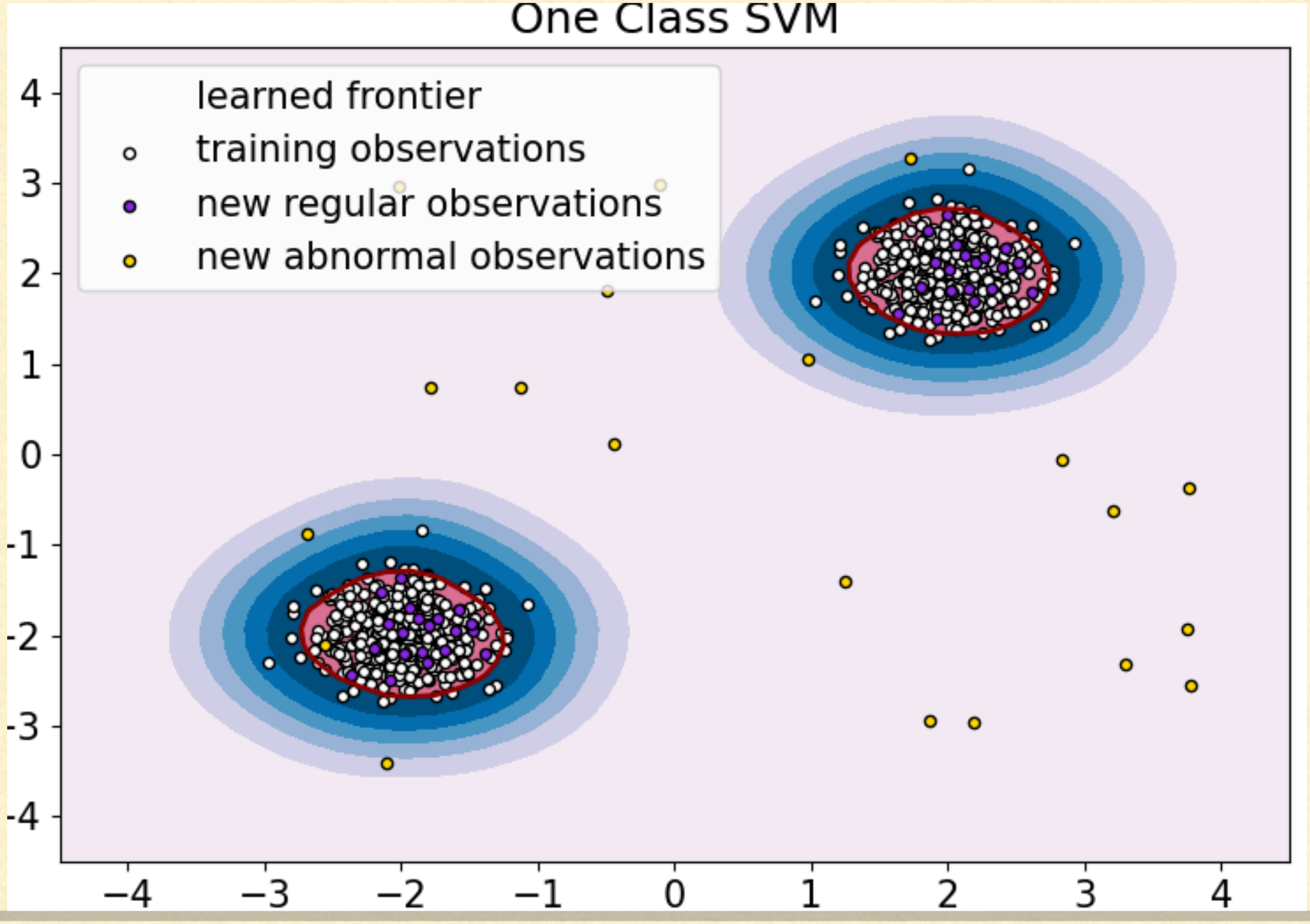
$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

# MAHALANOBIS DETECTOR

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}.$$

# ONE CLASS SVM



# IMPLEMENTATION

- For each detector:
  - For each subjects:
    - calculate user_score from testing_set;
    - calculate imposter_score from imposter_set;
    - calculate and draw ROC using user_score and imposter_score
- evaluate using average AUCs

# MODELS-TWO PHASES

- During training phase, the detector method calculates the mean_vector or fit the data using the TrainingSet

- During scoring phase, giving the TestingSet or ImposterSet, the model calculates the Manhattan/Euclidean/Mahalanobis distance between each training vector and the mean vector, or the SVM predictions. Return the scoring vector.

# RESULTS IN ROC

- After training, use the true positive rate and false positive rate to draw ROC

- In this scenario,  fnr represents the miss rate, and fpr represents false-alarm

- There are many ways to analyze this…

# IMPLEMENTATION

- For each detector:
  - For each subject:
    - Train the model with TrainingSet
    - Get user_score from TestingSet
    - Get imposter_score from ImposterSet
    - Calculate & Draw ROC
- Calculate average AUC

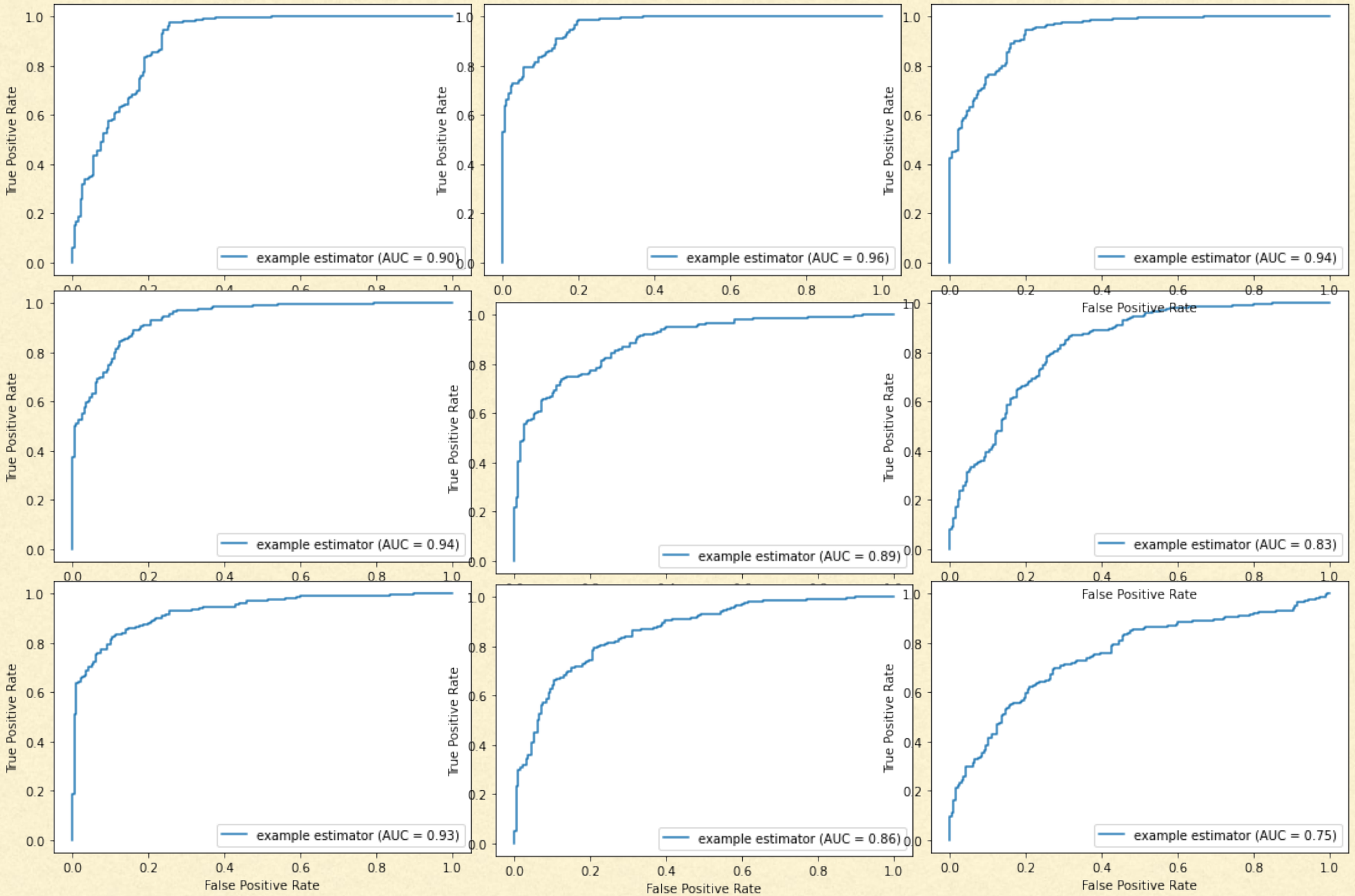



Figure 1. Euclidean Detector ROC from 9 random subjects

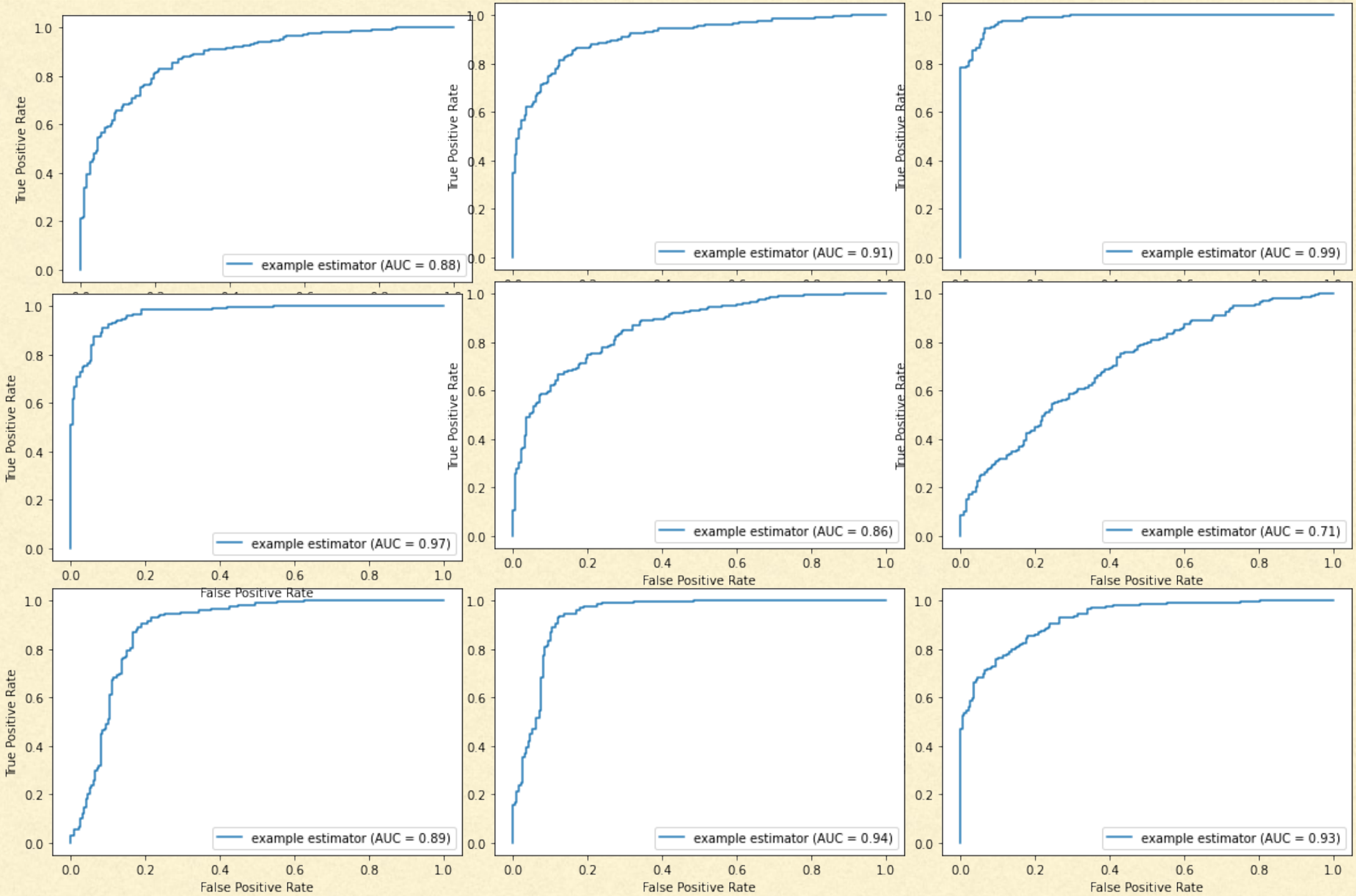



Figure 2. Manhattan Detector ROC from 9 random subjects

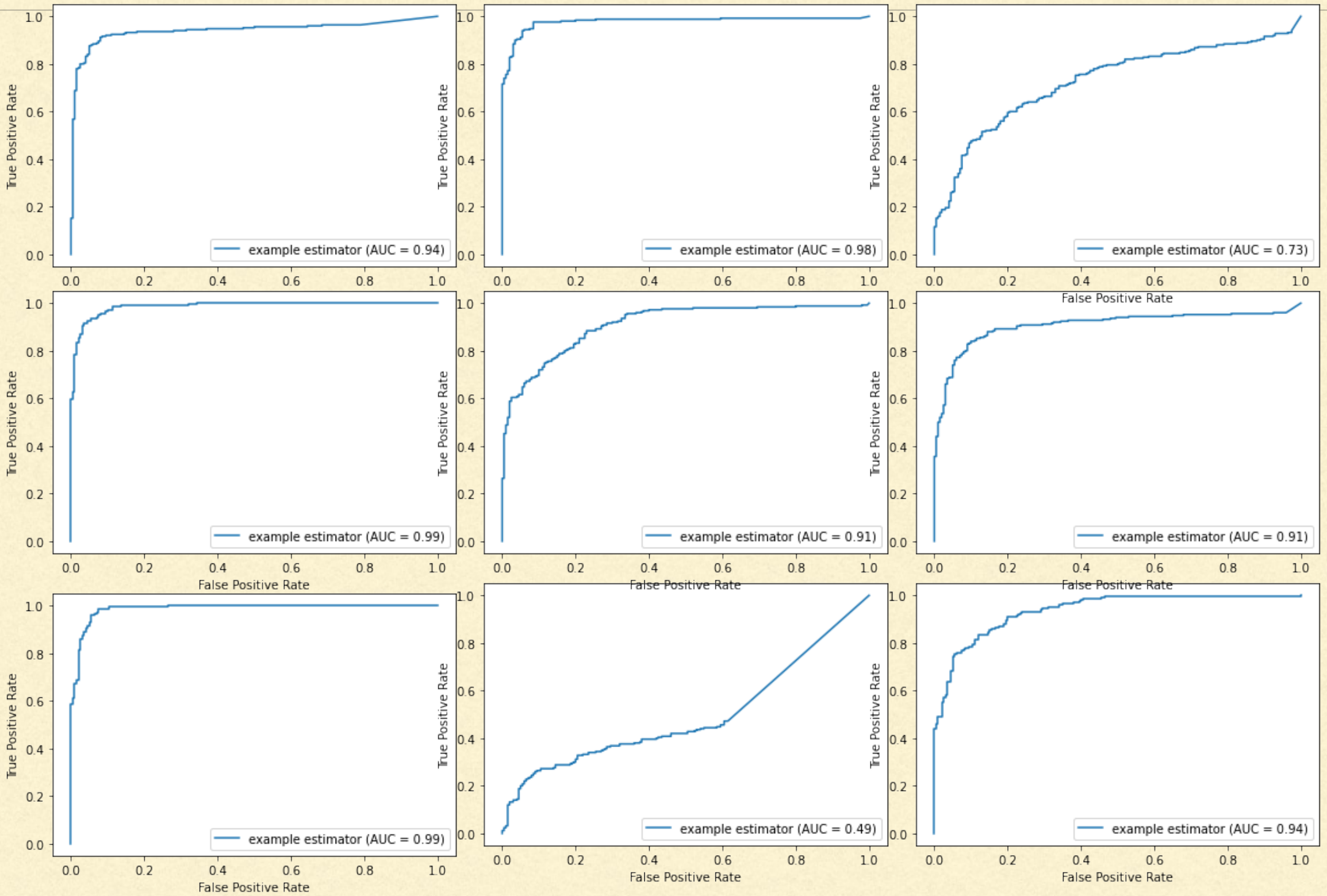



Figure 3. Mahalanobis Detector ROC from 9 random subjects

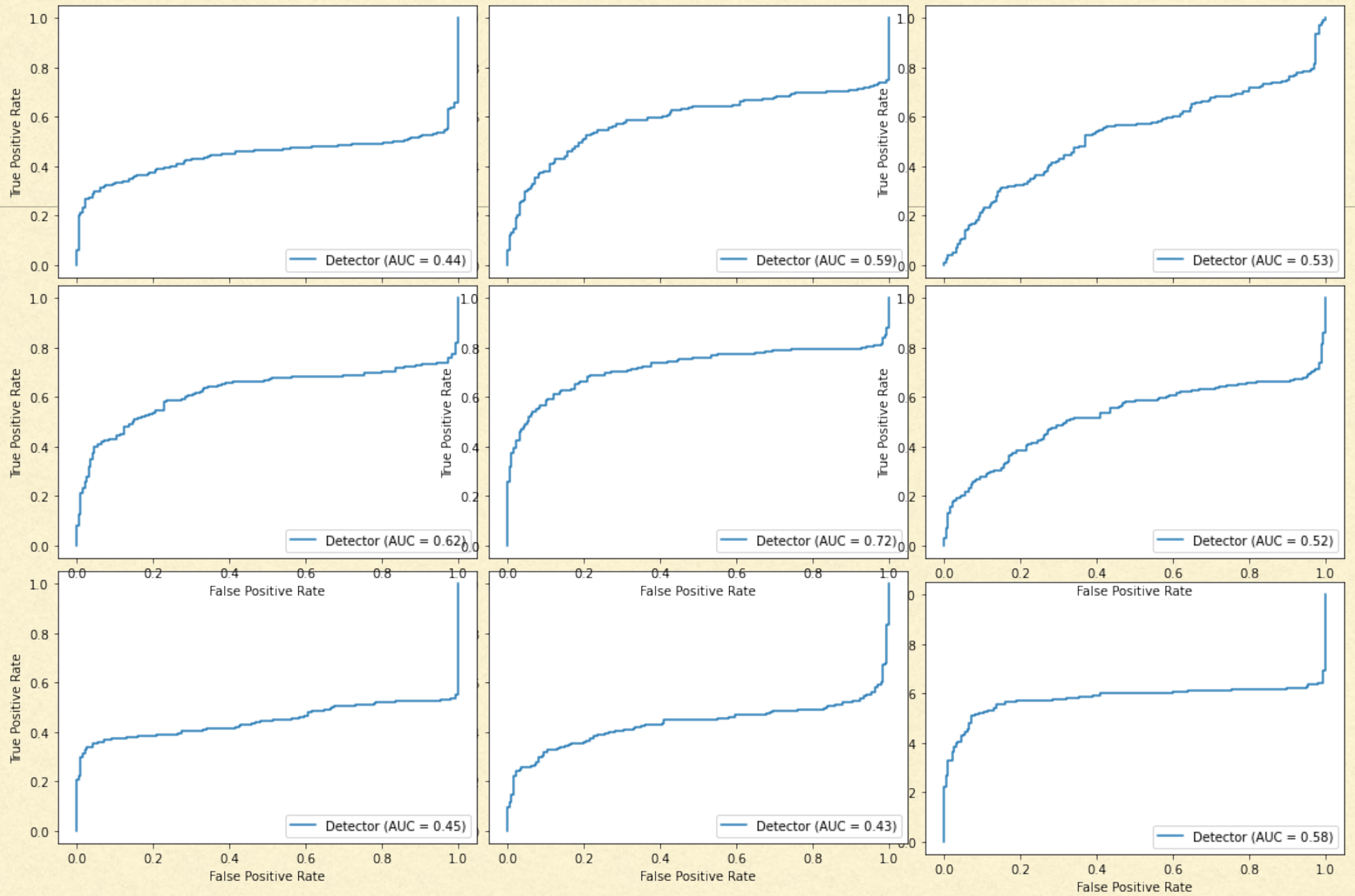



Figure 4. SVM detector with RBF kernel from 9 random subjects

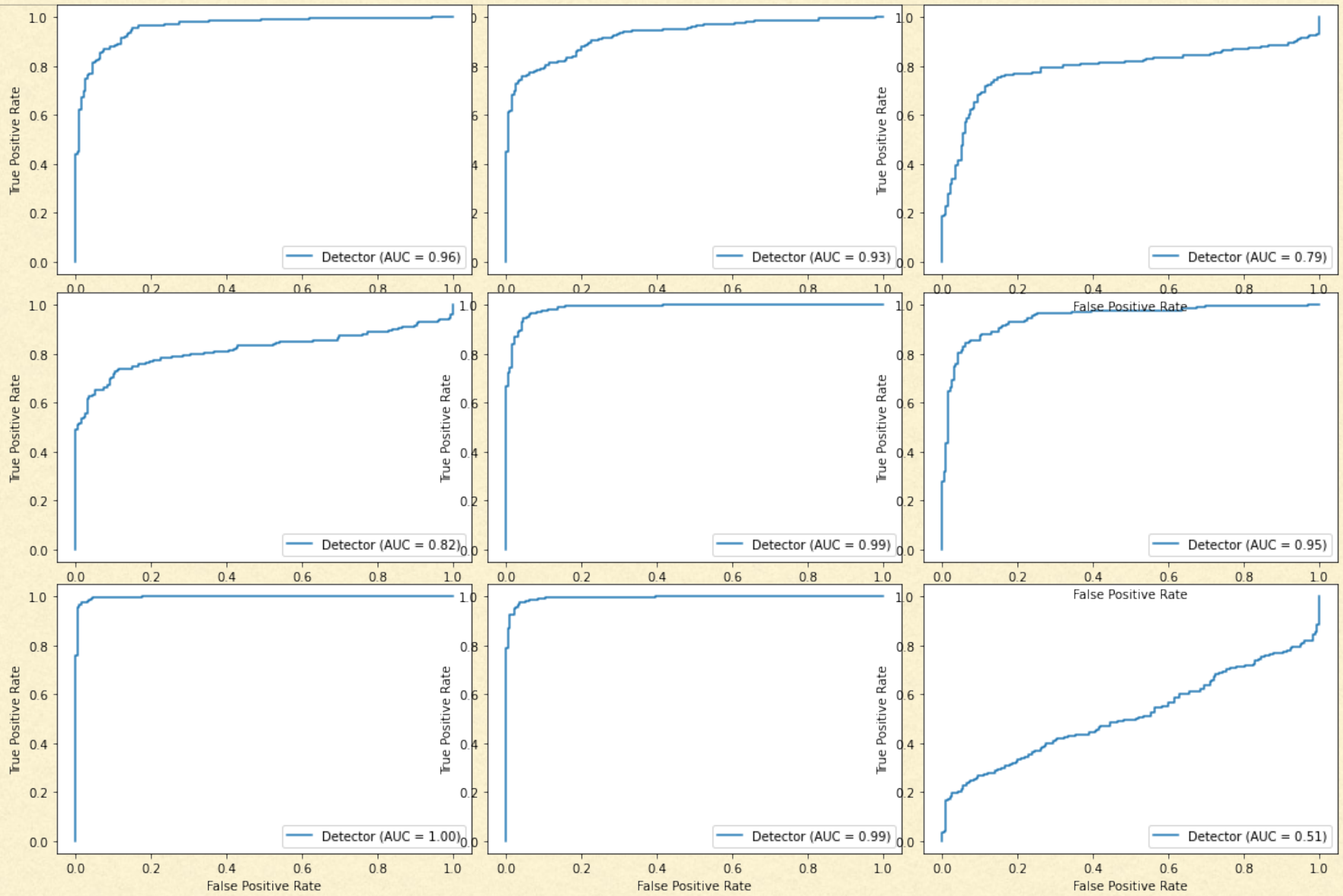



Figure 5. SVM with linear kernel ROC from 9 random subjects

# COMPARING AVERAGE SCORES

| Detector Name | Average AUC |
|---|---|
| Manhattan | 0.902622745 |
| Euclidean | 0.879335294 |
| Mahalanobis | 0.849603872 |
| SVM linear | 0.858285098 |
| SVM rbf | 0.476107451 |
| SVM polynomial | 0.85914 |
| SVM sigmoid | 0.857907451 |

# OBSERVATIONS

- Using statistic tools can guarantee a certain degree of robustness of the model.
  - Manhattan > Euclidean > Mahalanobis—>each features is more “proportional” to each other
- SVMs as the top regression algorithm—> robust with relatively small dataset.
- Determining Kernel functions by the “shape” of the dataset.

# THANK YOU

# Q&A

# REFERENCES


- https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html
- https://en.wikipedia.org/wiki/Keystroke_dynamics
- https://www.cs.cmu.edu/~maxion/pubs/KillourhyMaxion09.pdf