# DAFTAR PUSTAKA


Abdulgafoor M. Bachani, Margie Peden, G. Gururaj, Robyn Norton, and A. A. H. (2017). *Injury Prevention and Environmental Health. 3rd editio* ( et al Mock CN, Nugent R, Kobusingye O (ed.); 3rd ed.). https://www.ncbi.nlm.nih.gov/books/NBK525212/

Badan Pusat Statistik. (n.d.). *Jumlah Kecelakaan, Korban Mati, Luka Berat, Luka Ringan, dan Kerugian Materi Tahun 2019 - 2021*. Retrieved July 17, 2023, from https://www.bps.go.id/indicator/17/513/1/jumlah-kecelakaan-korban-mati-luka-berat-luka-ringan-dan-kerugian-materi.html

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. https://doi.org/10.1007/s11042-022-13644-y

Duman, E., & Erdem, O. A. (2019). Anomaly Detection in Videos Using Optical Flow and Convolutional Autoencoder. *IEEE Access*, *7*(December), 183914–183923. https://doi.org/10.1109/ACCESS.2019.2960654

Gao, X., Luo, H., Wang, Q., Zhao, F., Ye, L., & Zhang, Y. (2019). A human activity recognition algorithm based on stacking denoising autoencoder and lightGBM. *Sensors (Switzerland)*, *19*(4), 1–20. https://doi.org/10.3390/s19040947

Ghahremannezhad, H., Shi, H., & Liu, C. (2022). Real-Time Accident Detection in Traffic Surveillance Using Deep Learning. *IST 2022 - IEEE International Conference on Imaging Systems and Techniques, Proceedings*. https://doi.org/10.1109/IST55454.2022.9827736

Ijjina, E. P., Chand, D., Gupta, S., & Goutham, K. (2019). Computer Vision-based Accident Detection in Traffic Surveillance. *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*, 1–6. https://doi.org/10.1109/ICCCNT45670.2019.8944469

Latah, M. (2017). Human action recognition using support vector machines and 3D convolutional neural networks. *International Journal of Advances in Intelligent Informatics*, *3*(1), 47–55. https://doi.org/10.26555/ijain.v3i1.89

Maaloul, B. (2020). *Video-based algorithms for accident detections*. https://theses.hal.science/tel-02880464

Maaloul, B., Taleb-Ahmed, A., Niar, S., Harb, N., & Valderrama, C. (2017). Adaptive video-based algorithm for accident detection on highways. *2017 12th IEEE International Symposium on Industrial Embedded Systems, SIES 2017 - Proceedings*. https://doi.org/10.1109/SIES.2017.7993382

McCarty, D. A., Kim, H. W., & Lee, H. K. (2020). Evaluation of light gradient boosted machine learning technique in large scale land use and land cover classification. *Environments - MDPI*, *7*(10), 1–22. https://doi.org/10.3390/environments7100084

Pawar, K., & Attar, V. (2022). Deep learning based detection and localization of road accidents from traffic surveillance videos. *ICT Express*, *8*(3), 379–387. https://doi.org/10.1016/j.icte.2021.11.004

Prangga, S. (2017). *Optimasi Parameter Pada Support Vector Machine Menggunakan Pendekatan Metode Taguchi Untuk Data High-Dimensional Parameter Optimization of Support Vector Machine Using Taguchi Approach for High- Dimensional Data* [Institut Teknilogi Sepluh November]. https://repository.its.ac.id/3559/7/1315201017-Master-Theses.pdf

Rachmadi, R. R., Sudarsono, A., & Santoso, B. (2021). Implementasi Metode LightGBM Untuk Klasifikasi Kondisi Abnormal Pada Pengemudi Sepeda Motor Berbasis Sensor Smartphone. *Jurnal Komputer Terapan*, *7*(2), 218–227.

Rahman, Z., Ami, A. M., & Ullah, M. A. (2020). A Real-Time Wrong-Way Vehicle Detection Based on YOLO and Centroid Tracking. *2020 IEEE Region 10 Symposium, TENSYMP 2020*, *June*, 916–920. https://doi.org/10.1109/TENSYMP50017.2020.9230463

Rejitha, M. R., & George, S. N. (2019). An Unsupervised Abnormal Crowd Behavior Detection Technique using Farneback Algorithm. *2019 IEEE International Conference on Electronics, Computing and Communication Technologies, CONECCT 2019*. https://doi.org/10.1109/CONECCT47791.2019.9012845

Robles-Serrano, S., Sanchez-Torres, G., & Branch-Bedoya, J. (2021). Automatic detection of traffic accidents from video using deep learning techniques. *Computers*, *10*(11), 1–16. https://doi.org/10.3390/computers10110148

Ruwali, A., Kumar, A. J. S., Prakash, K. B., Sivavaraprasad, G., & Ratnam, D. V. (2021). Implementation of Hybrid Deep Learning Model (LSTM-CNN) for Ionospheric TEC Forecasting Using GPS Data. *IEEE Geoscience and Remote Sensing Letters*, *18*(6), 1004–1008. https://doi.org/10.1109/LGRS.2020.2992633

Saini, A., Suregaonkar, S., Gupta, N., Karar, V., & Poddar, S. (2018). Region and feature matching based vehicle tracking for accident detection. *2017 10th International Conference on Contemporary Computing, IC3 2017*, *2018-Janua*(August), 1–6. https://doi.org/10.1109/IC3.2017.8284322

Sanchez, S. A., Romero, H. J., & Morales, A. D. (2020). A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. *IOP Conference Series: Materials Science and Engineering*, *844*(1). https://doi.org/10.1088/1757-899X/844/1/012024

Sari, L., Romadloni, A., Lityaningrum, R., & Hastuti, H. D. (2023). *Implementation of LightGBM and Random Forest in Potential Customer Classification*. *4*(1). https://doi.org/10.38043/tiers.v4i1.4355

Seo, S. B., & Singh, D. (2022). Smart Town Traffic Management System using LoRa and Machine Learning Mechanism. *IEEE Technology Policy and Ethics*, *3*(6), 1–4. https://doi.org/10.1109/ntpe.2018.9778109

Septyanto, B. A., Wibowo, S. A., & Setianingsih, C. (2022). Implementasi Face Recognition Berbasis Deep Neural Network Sebagai Sistem Kendali Pada Quadcopter. *E-Proceeding of Engineering*, *8*(6), 3036–3050.

Shah, A. P., Lamare, J. B., Nguyen-Anh, T., & Hauptmann, A. (2019). CADP: A Novel Dataset for CCTV Traffic Camera based Accident Analysis. *Proceedings of AVSS 2018 - 2018 15th IEEE International Conference on Advanced Video and Signal-Based Surveillance*, *i*, 1–9. https://doi.org/10.1109/AVSS.2018.8639160

Tian, D., Zhang, C., Duan, X., & Wang, X. (2019). An Automatic Car Accident Detection Method Based on Cooperative Vehicle Infrastructure Systems. *IEEE Access*, *7*, 127453–127463. https://doi.org/10.1109/ACCESS.2019.2939532

Wang, C., Dai, Y., Zhou, W., & Geng, Y. (2020). A Vision-Based Video Crash Detection Framework for Mixed Traffic Flow Environment Considering Low-Visibility Condition. *Journal of Advanced Transportation*, *2020*. https://doi.org/10.1155/2020/9194028

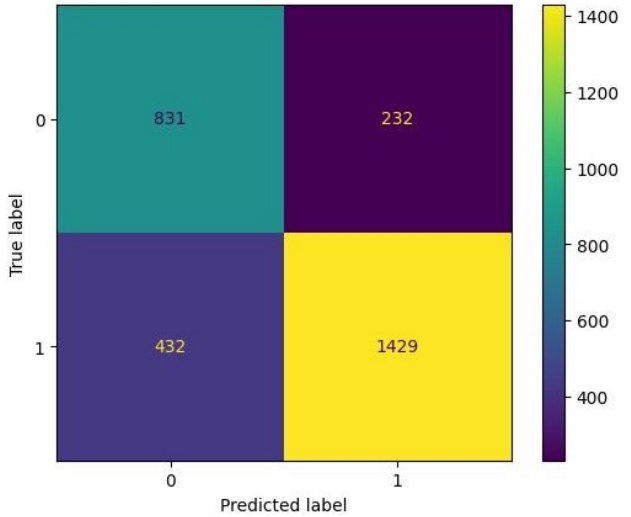
Yang, J., Han, S., & Chen, Y. (2023). Prediction of Traffic Accident Severity Based on Random Forest. *Journal of Advanced Transportation*, *2023*. https://doi.org/10.1155/2023/7641472

Zhang, Y., & Sung, Y. (2023). Hybrid Traffic Accident Classification Models. *Mathematics*, *11*(4). https://doi.org/10.3390/math11041050

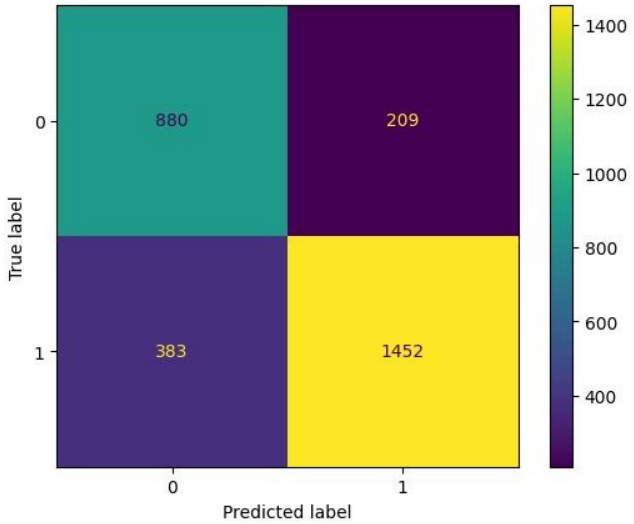
Zheng, M., Li, T., Zhu, R., Chen, J., Ma, Z., Tang, M., Cui, Z., & Wang, Z. (2019). Traffic accident's severity prediction: A deep-learning approach-based CNN network. *IEEE Access*, *7*(c), 39897–39910. https://doi.org/10.1109/ACCESS.2019.2903319

# LAMPIRAN

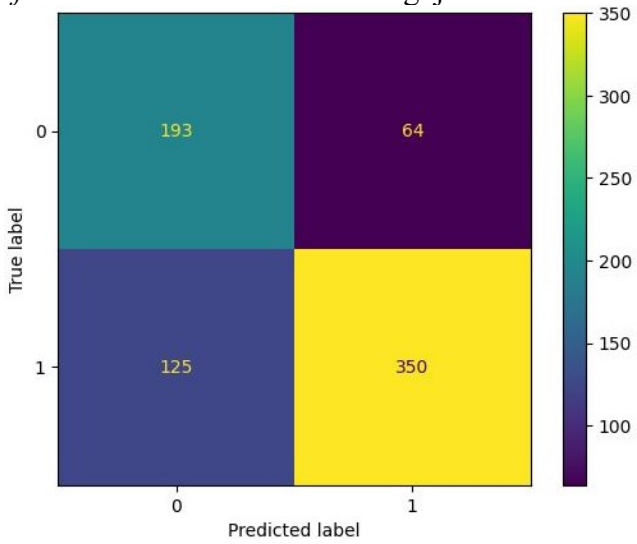
**Lampiran 1** *Confusion Matrix* SVM untuk Pembelajaran K-fold 5



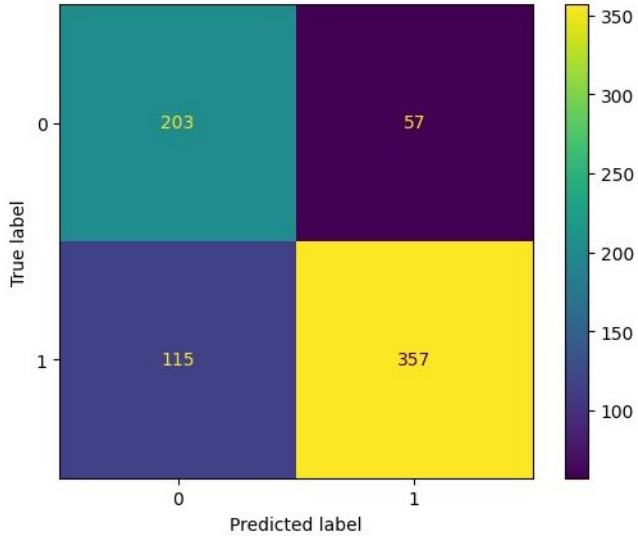

**Lampiran 2** *Confusion Matrix* SVM untuk Pembelajaran K-fold 10



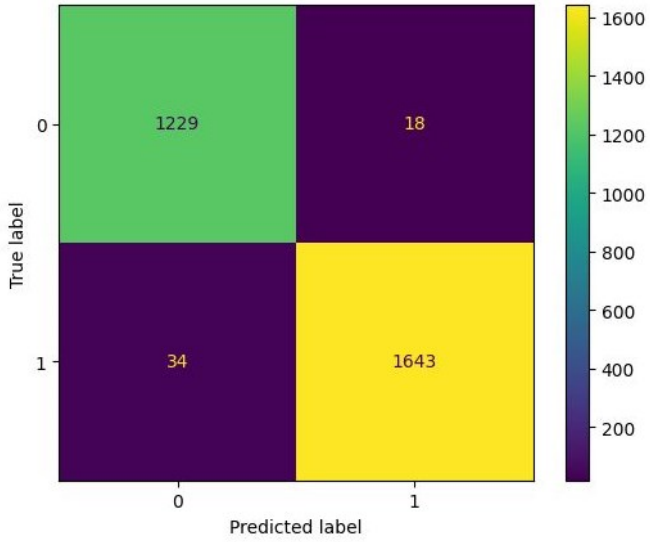

**Lampiran 3** *Confusion Matrix* SVM untuk Pengujian K-fold 5



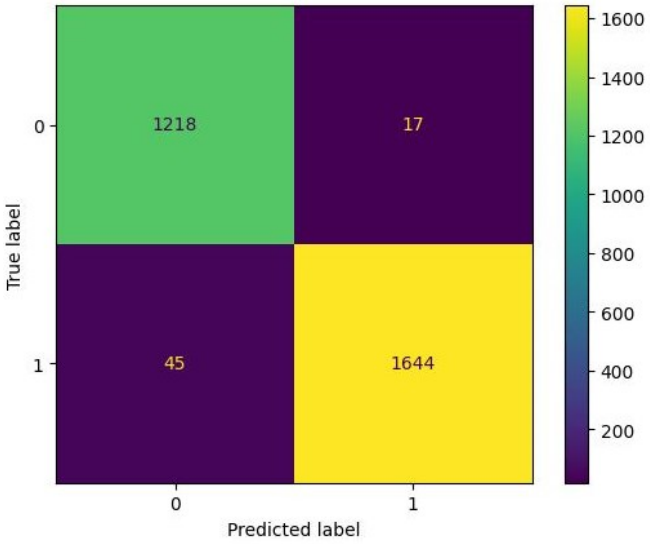
**Lampiran 4** *Confusion Matrix* SVM untuk Pengujian K-fold 10



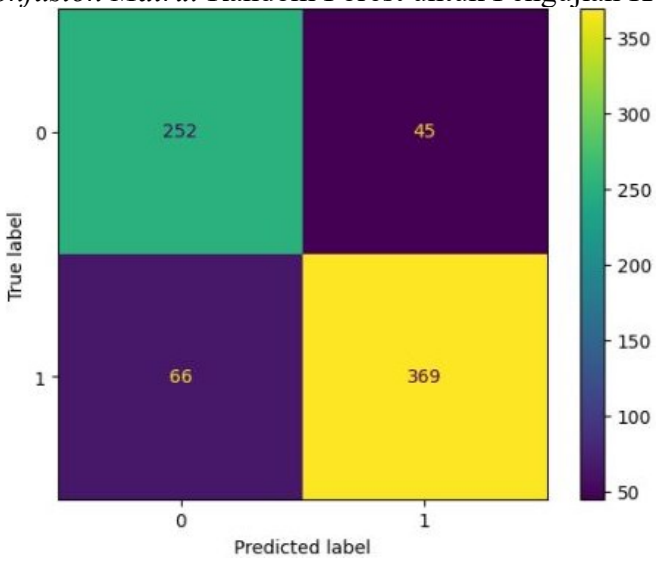

**Lampiran 5** *Confusion Matrix* Random Forest untuk Pembelajaran K-fold 5



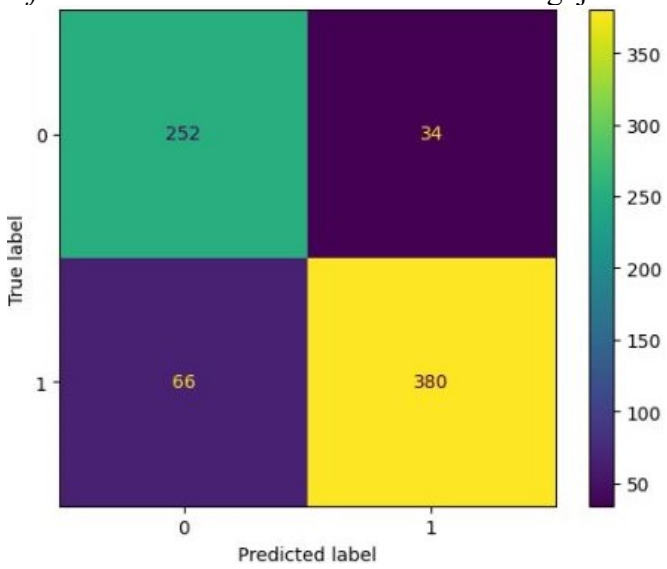

**Lampiran 6** *Confusion Matrix* Random Forest untuk Pembelajaran K-fold 10



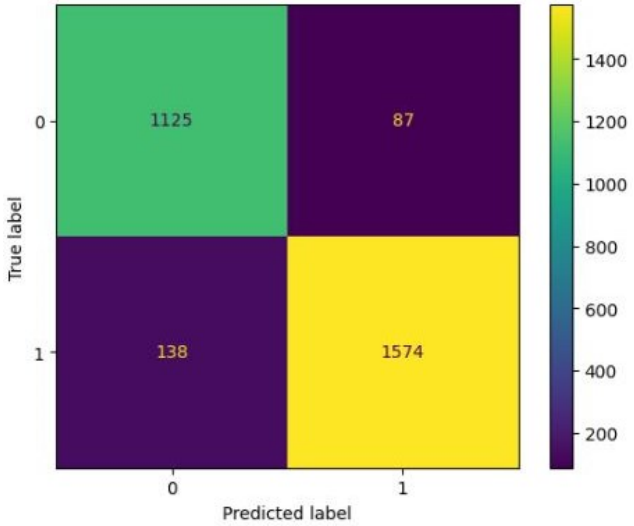
**Lampiran 7** *Confusion Matrix* Random Forest untuk Pengujian K-fold 5



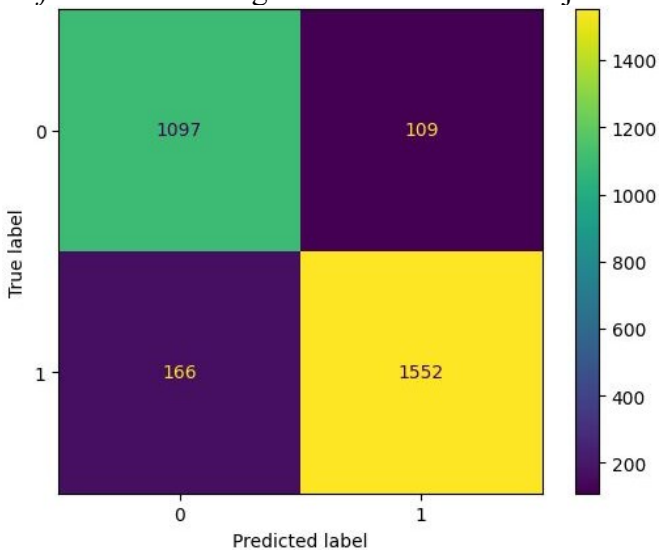

**Lampiran 8** *Confusion Matrix* Random Forest untuk Pengujian K-fold 10



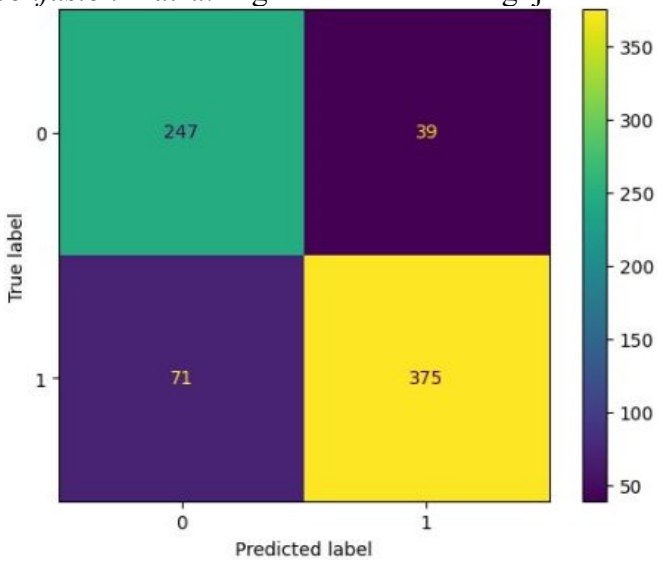

**Lampiran 9** *Confusion Matrix* LightGBM untuk Pembelajaran K-fold 5



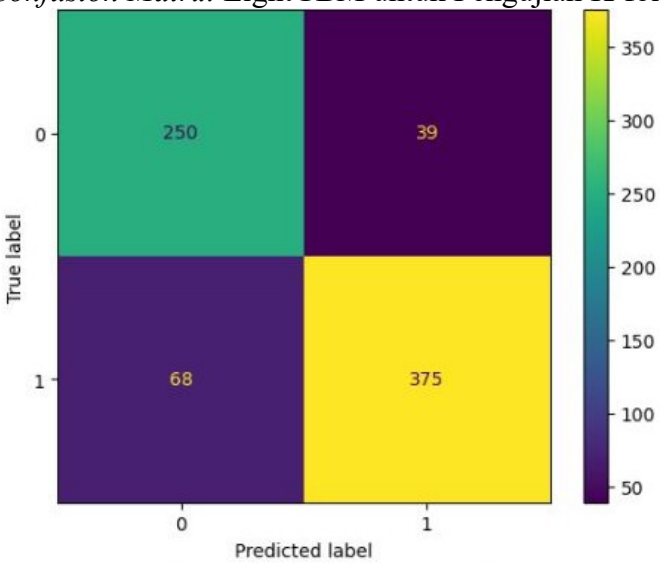
**Lampiran 10** *Confusion Matrix* LightGBM untuk Pembelajaran K-fold 10




**Lampiran 11** *Confusion Matrix* LightGBM untuk Pengujian K-fold 5




**Lampiran 12** *Confusion Matrix* LightGBM untuk Pengujian K-fold 10



**Lampiran 13** *Potongan frame kondisi lalu lintas pada video A01*


*Frame 119*

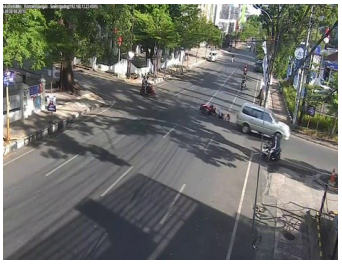

*Frame 123*


Frame 136

**Lampiran 14** *Potongan frame kondisi lalu lintas pada video A02*


*Frame 138*


Frame 171


Frame 181

**Lampiran 15** *Potongan frame kondisi lalu lintas pada video A03*

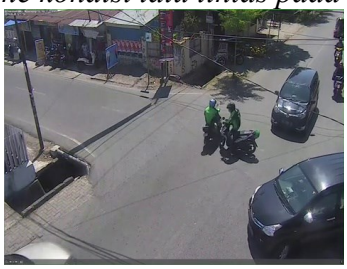
*Frame 169*

*Frame 177*

Frame 271

**Lampiran 16** *Potongan frame kondisi lalu lintas pada video A04*


*Frame 275*


*Frame 286*


Frame 305

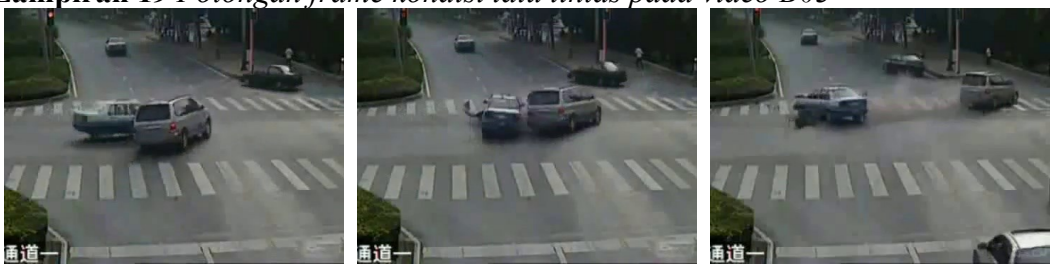
**Lampiran 17** *Potongan frame kondisi lalu lintas pada video B01*

*Frame 181* *Frame 188* Frame 216

**Lampiran 18** *Potongan frame kondisi lalu lintas pada video B02*

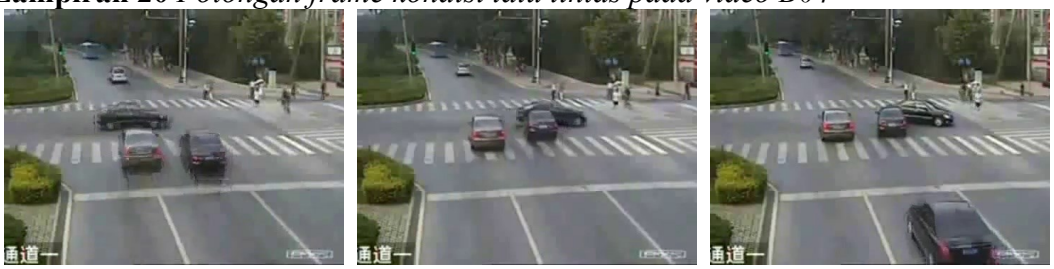
*Frame 309* Frame 321 Frame 345

**Lampiran 19** *Potongan frame kondisi lalu lintas pada video B03*



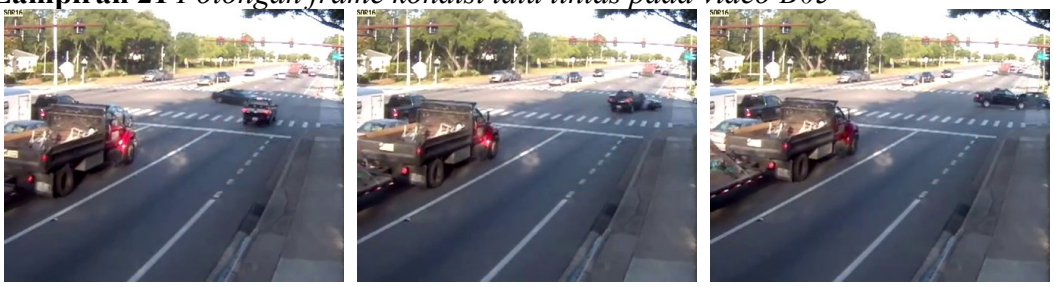
*Frame 178* Frame 191 Frame 220

**Lampiran 20** *Potongan frame kondisi lalu lintas pada video B04*



*Frame 250* Frame 275 Frame 311

**Lampiran 21** *Potongan frame kondisi lalu lintas pada video B05*

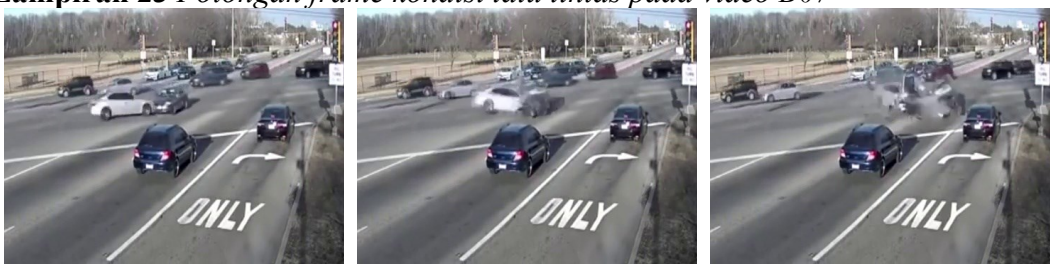


*Frame 138* Frame 151 Frame 170

**Lampiran 22** *Potongan frame kondisi lalu lintas pada video B06*

*Frame 191* *Frame 199* Frame 236

**Lampiran 23** *Potongan frame kondisi lalu lintas pada video B07*



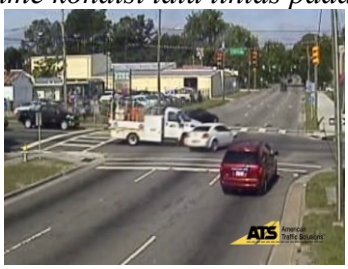

*Frame 400* Frame 404 Frame 412

**Lampiran 24** *Potongan frame kondisi lalu lintas pada video B08*

*Frame 31* Frame 41 Frame 61

**Lampiran 25** *Potongan frame kondisi lalu lintas pada video B09*


*Frame 61*


Frame 73


Frame 86

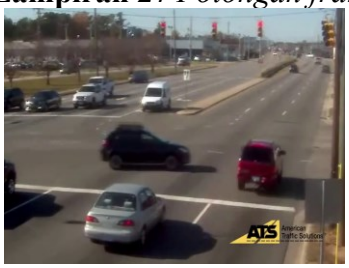
**Lampiran 26** *Potongan frame kondisi lalu lintas pada video B10*

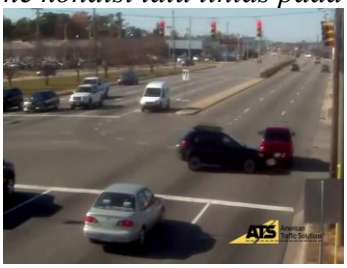

*Frame 31*

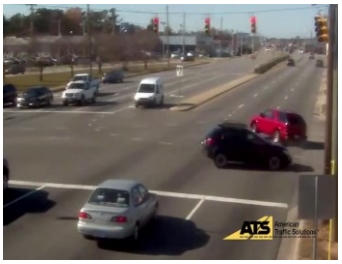

Frame 61


Frame 86

**Lampiran 27** *Potongan frame kondisi lalu lintas pada video B11*


*Frame 41*


*Frame 56*


Frame 71

**Lampiran 28** *Potongan frame kondisi lalu lintas pada video B12*


*Frame 66*


Frame 76


Frame 91

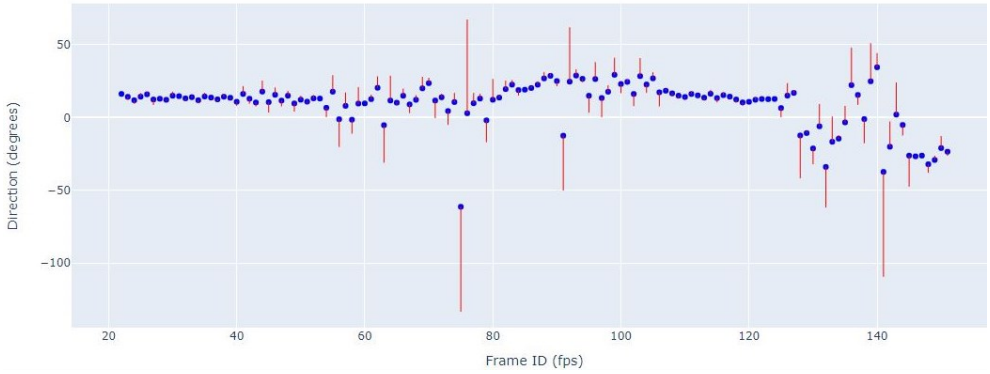
**Lampiran 29** *Potongan frame kondisi lalu lintas pada video B13*

*Frame 38* Frame 63 Frame 76

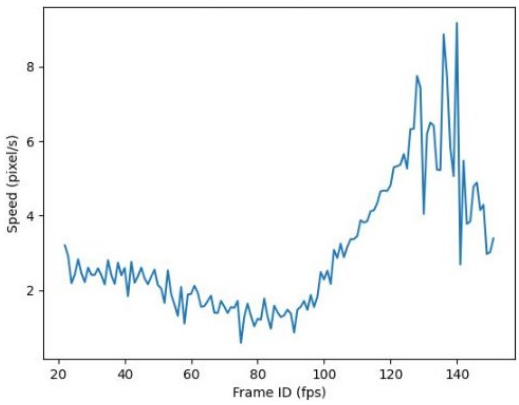
**Lampiran 30** *Potongan frame kondisi lalu lintas pada video B14*

*Frame 51* Frame 71 Frame 96

**Lampiran 31** *Pola berkendara objek ID A01_2*

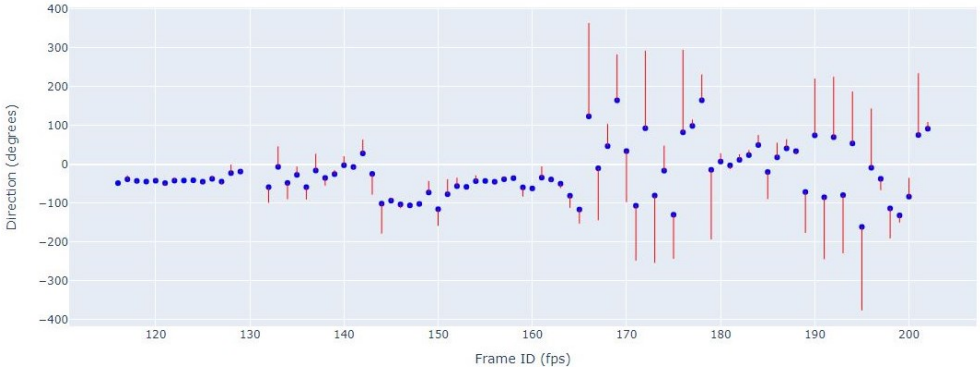



Fitur Arah

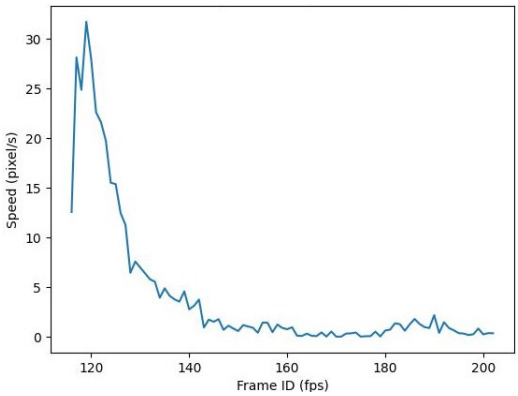



Fitur Kecepatan

**Lampiran 32** *Pola berkendara objek ID A01 7*

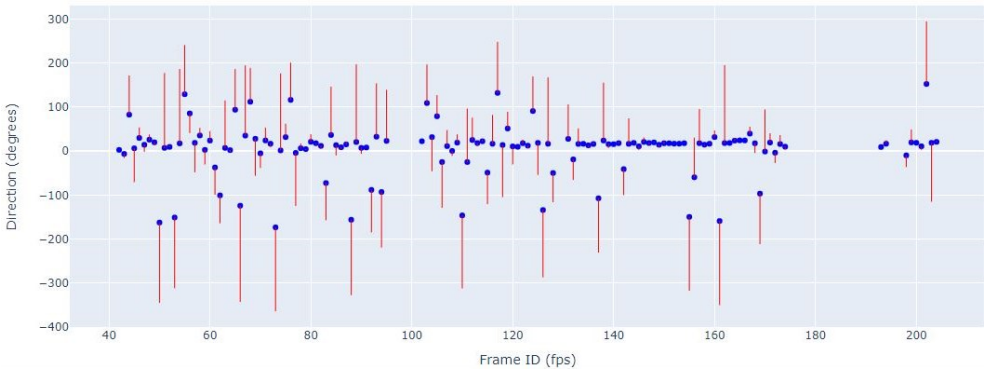



Fitur Arah

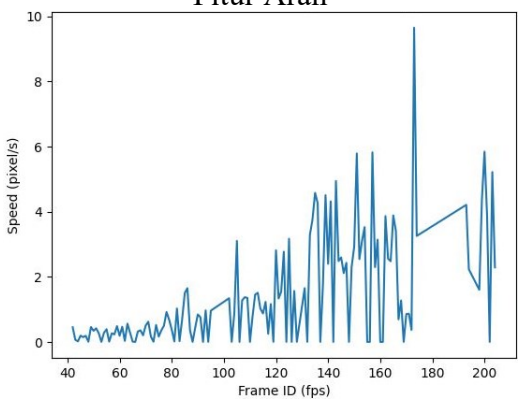



Fitur Kecepatan

**Lampiran 33** *Pola berkendara objek ID A02 5*

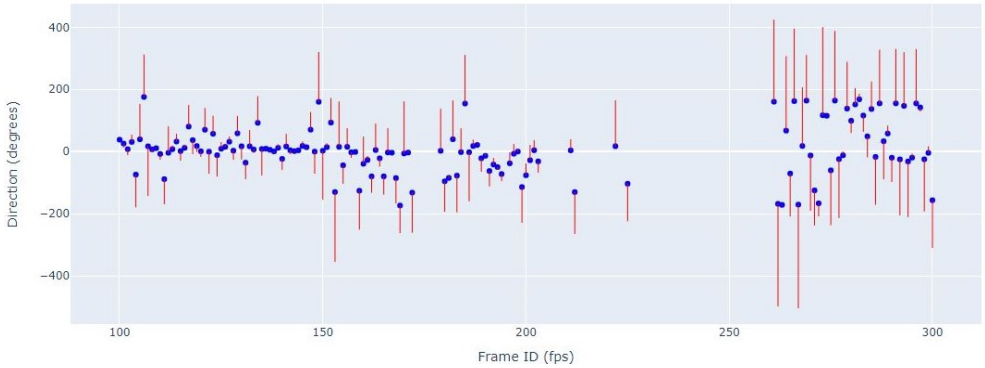



Fitur Arah

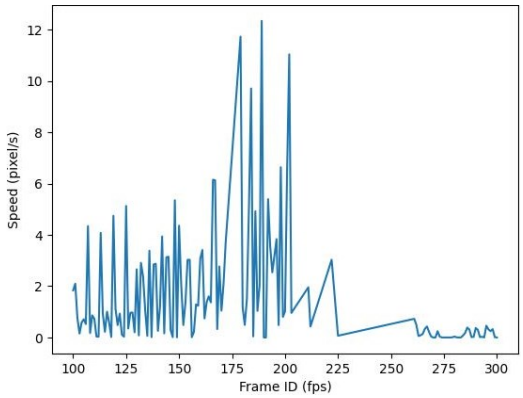



Fitur Kecepatan

**Lampiran 34** *Pola berkendara objek ID B01_6*

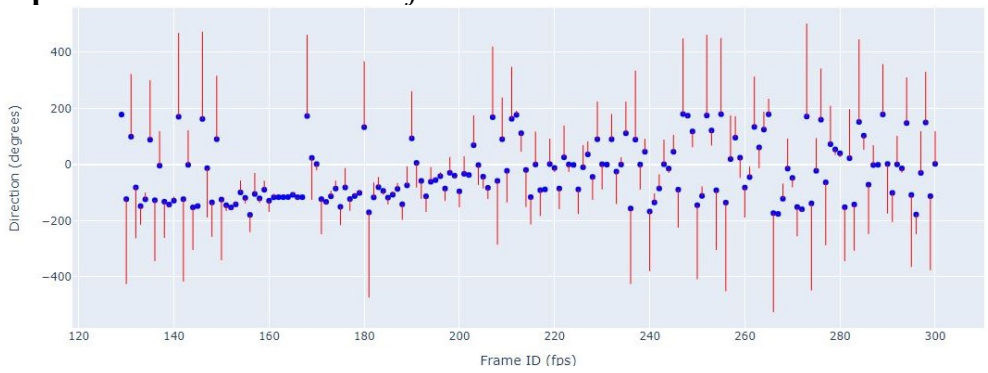



Fitur Arah

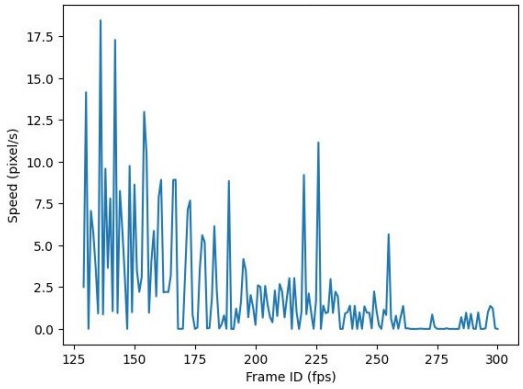



Fitur Kecepatan

**Lampiran 35** *Pola berkendara objek ID B01_8*

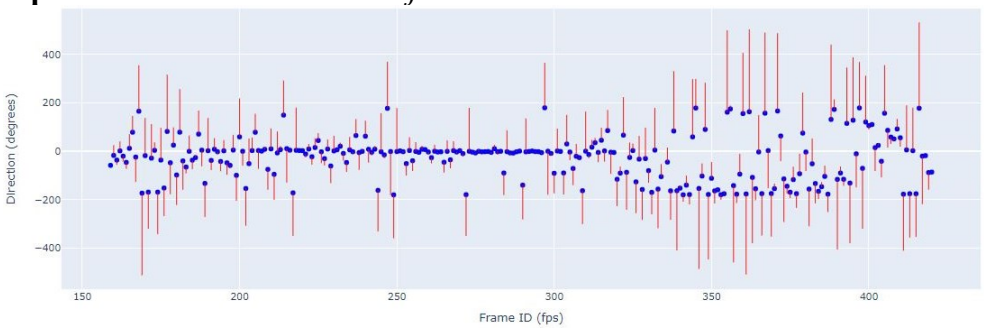



Fitur Arah

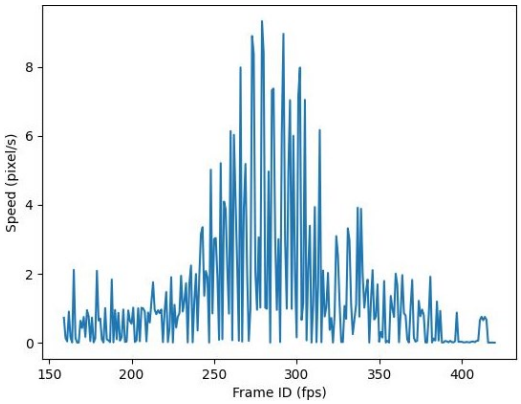



Fitur Kecepatan

**Lampiran 36** *Pola berkendara objek ID B02_11*

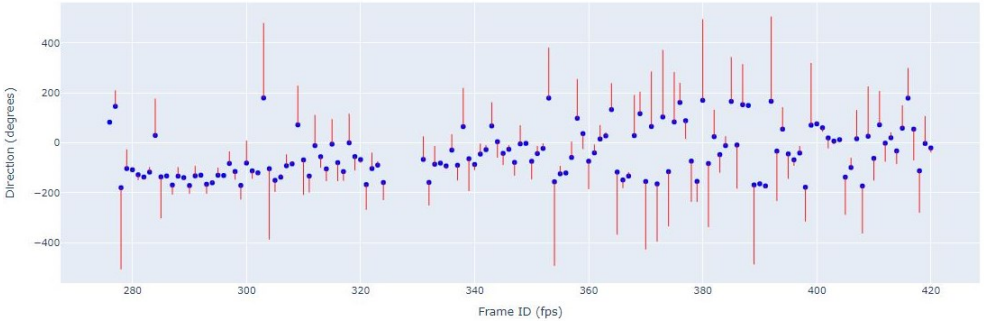



Fitur Arah

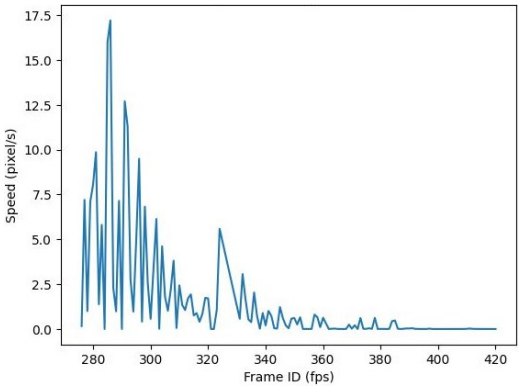



Fitur Kecepatan

**Lampiran 37** *Pola berkendara objek ID B02_14*

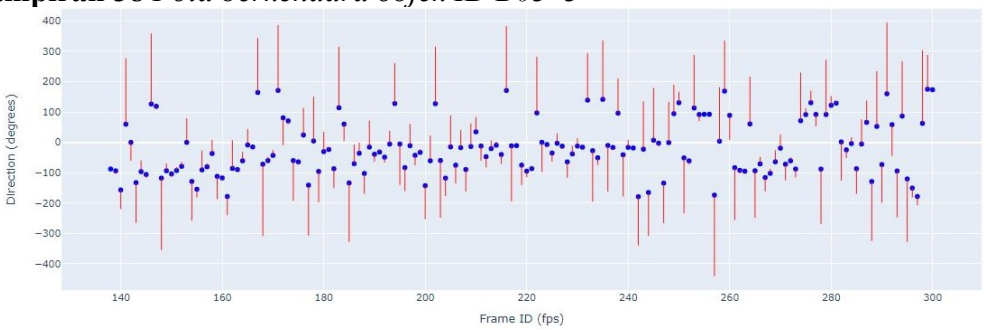



Fitur Arah

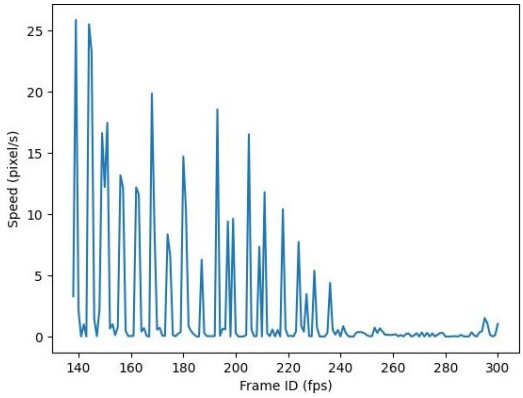



Fitur Kecepatan

**Lampiran 38** *Pola berkendara objek ID B03  3*

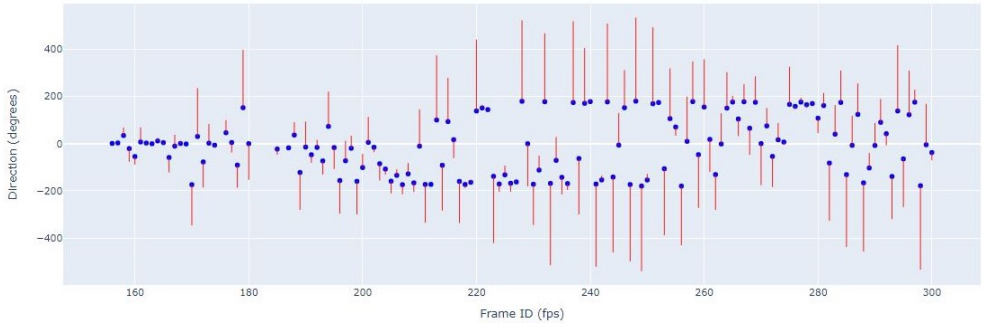



Fitur Arah

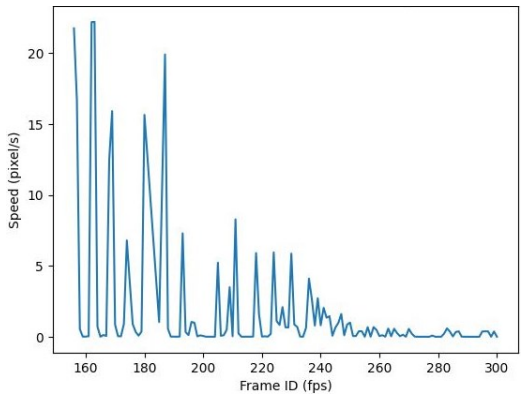



Fitur Kecepatan

**Lampiran 39** *Pola berkendara objek ID B03  4*

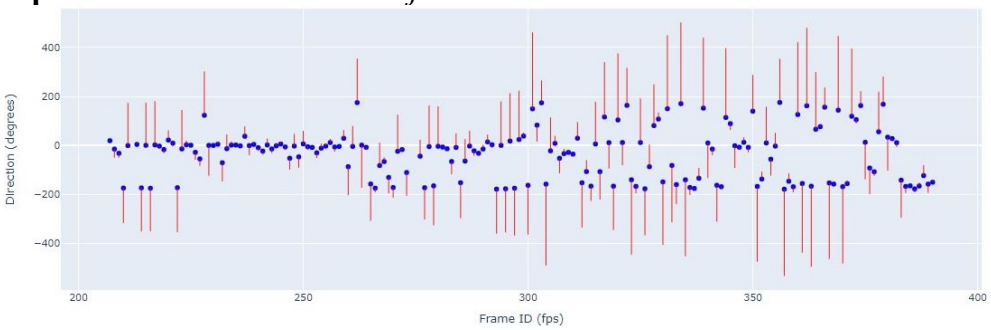



Fitur Arah

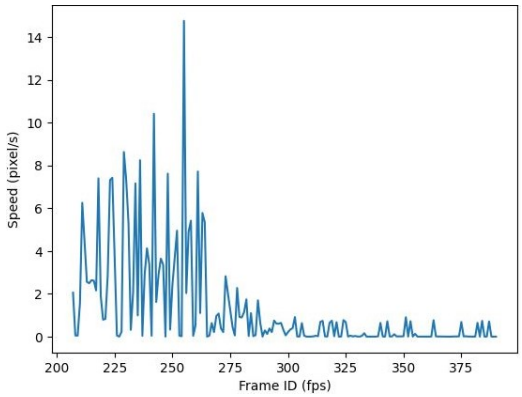



Fitur Kecepatan

**Lampiran 40** *Pola berkendara objek ID B04_6*

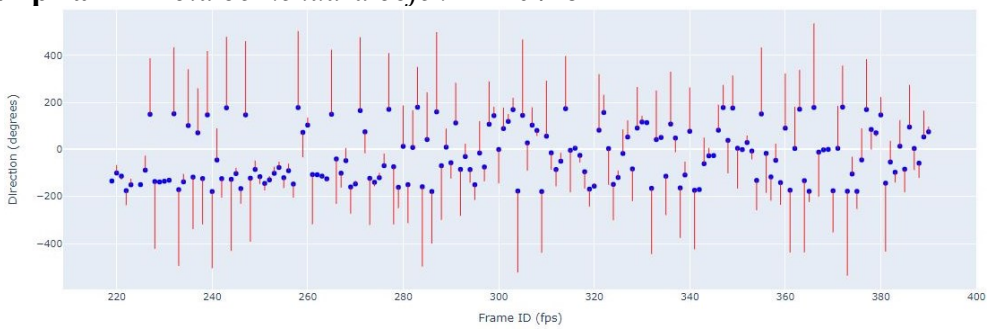



Fitur Arah

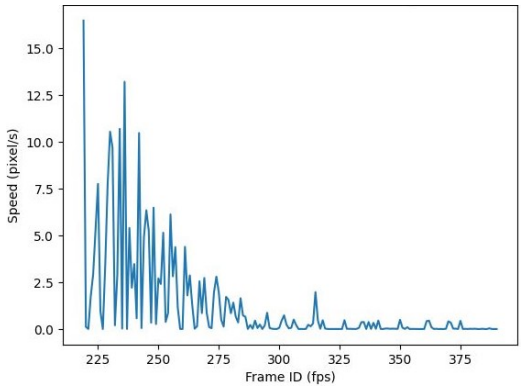



Fitur Kecepatan

**Lampiran 41** *Pola berkendara objek ID B04_8*

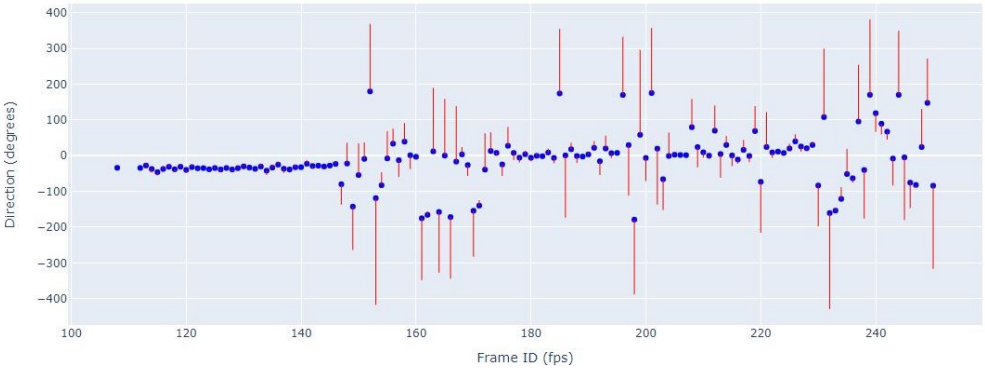



Fitur Arah

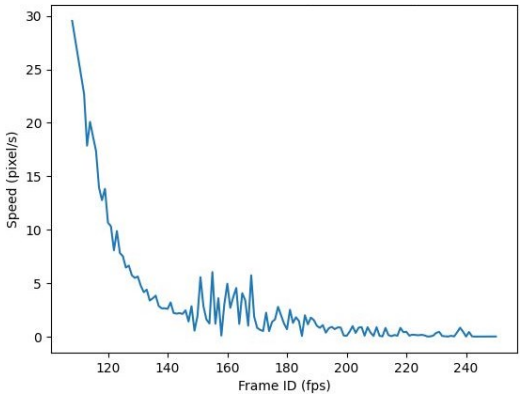



Fitur Kecepatan

**Lampiran 42** *Pola berkendara objek ID B05_9*

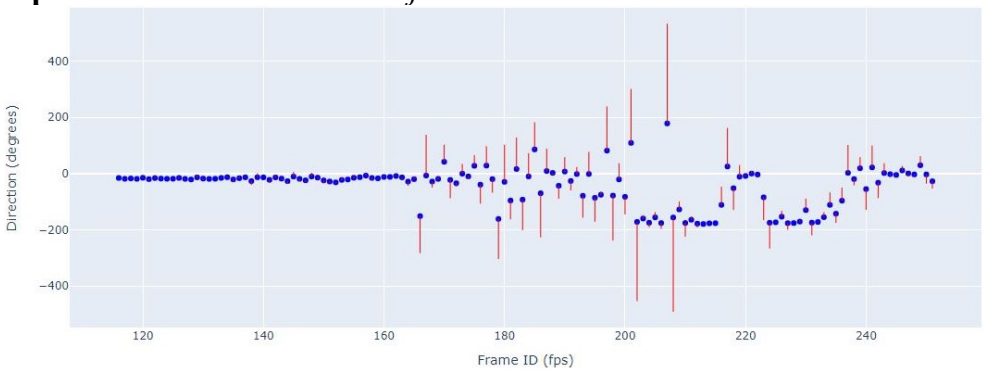



Fitur Arah

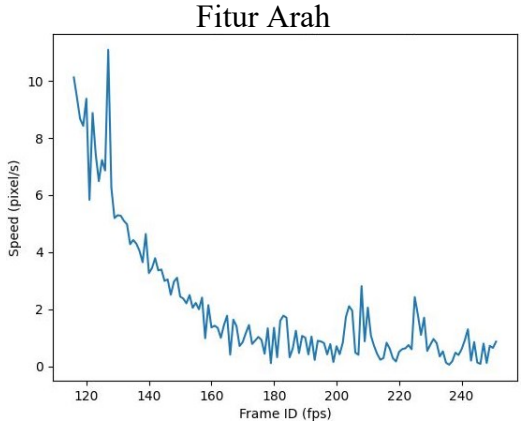



Fitur Kecepatan

**Lampiran 43** *Pola berkendara objek ID B06_7*

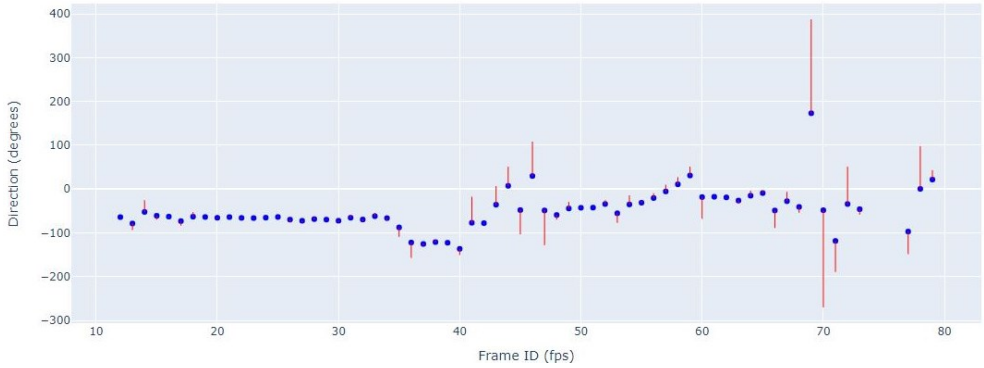



Fitur Arah

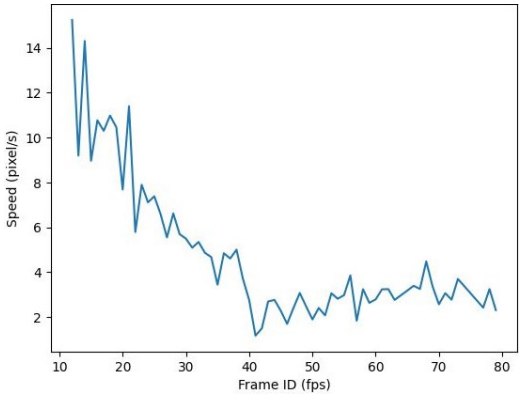



Fitur Kecepatan

**Lampiran 44** *Pola berkendara objek ID B08_3*

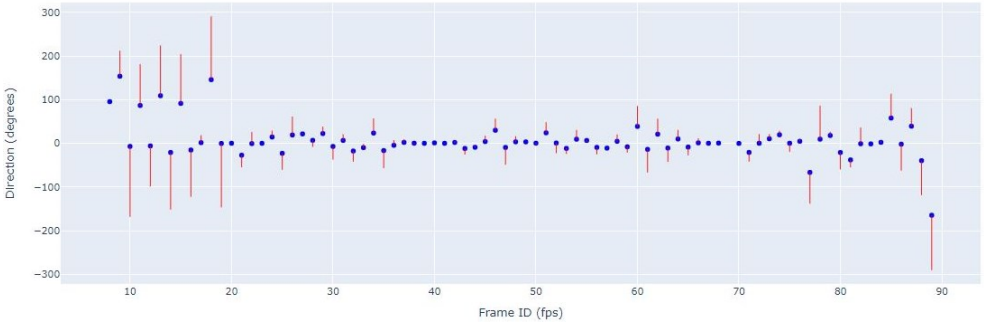



Fitur Arah

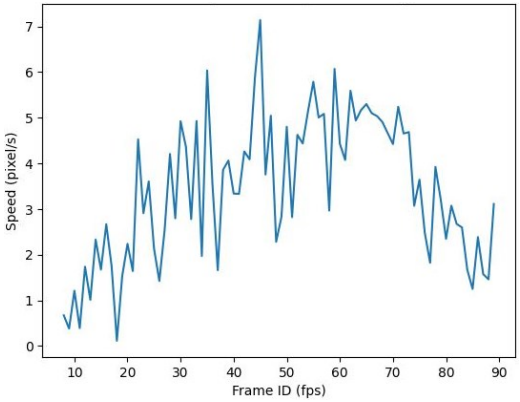



Fitur Kecepatan

**Lampiran 45** *Pola berkendara objek ID B09_3*

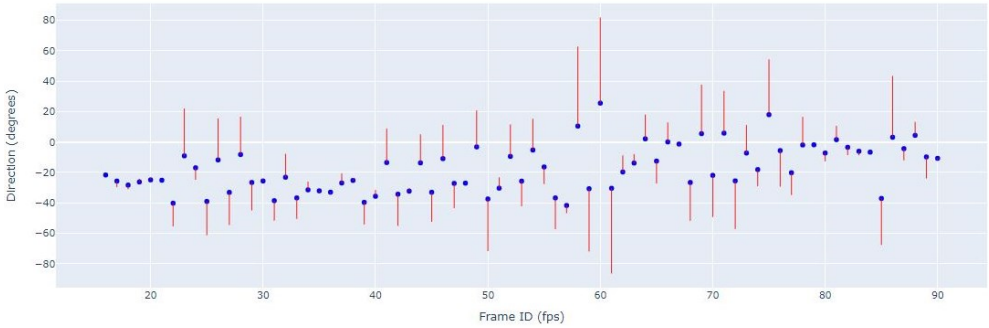



Fitur Arah

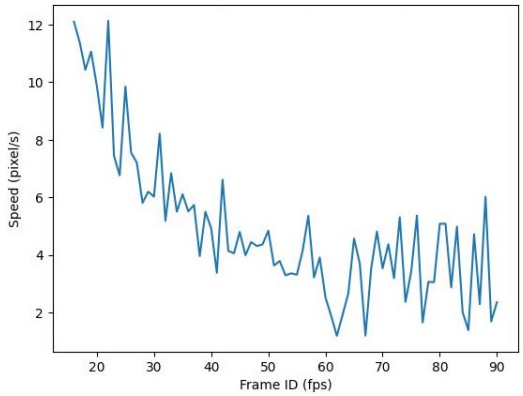



Fitur Kecepatan

**Lampiran 46** *Pola berkendara objek ID B10  6*

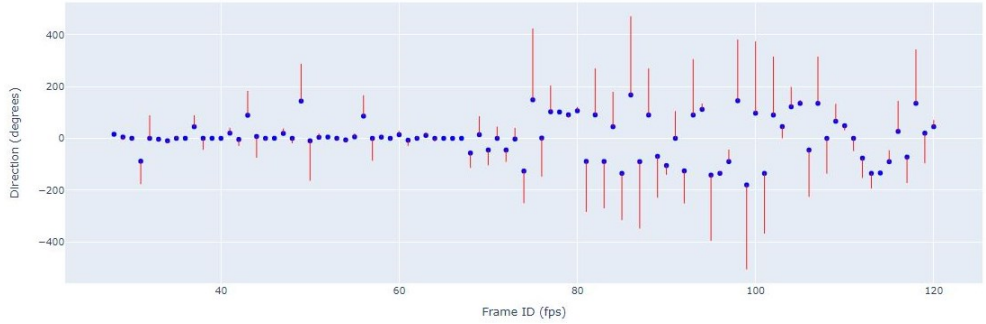



Fitur Arah

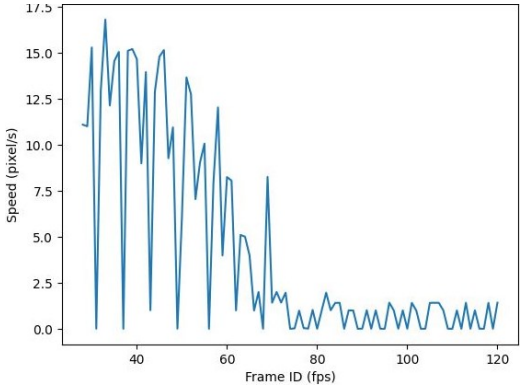



Fitur Kecepatan

**Lampiran 47** *Pola berkendara objek ID B11  8*

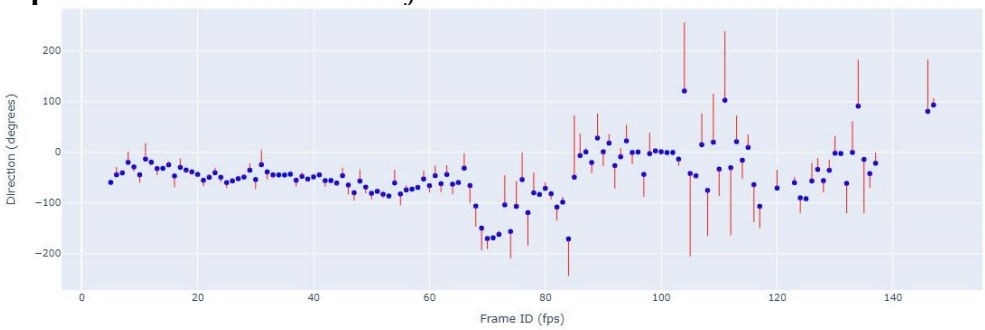



Fitur Arah

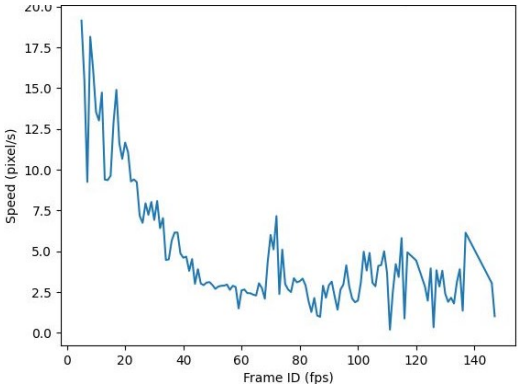



Fitur Kecepatan

**Lampiran 48** *Pola berkendara objek ID B12  3*

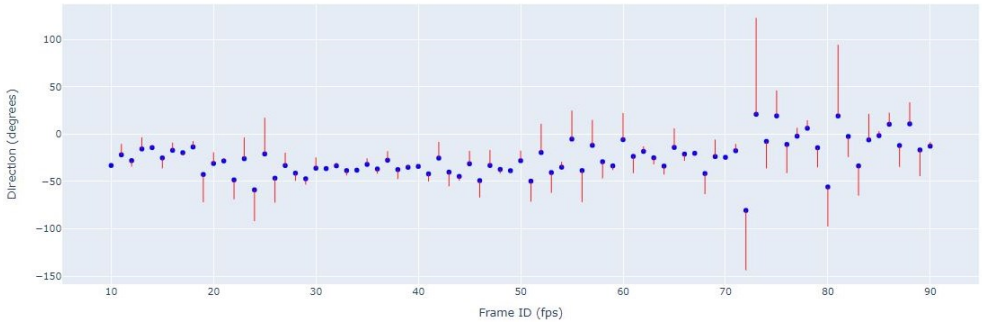



Fitur Arah

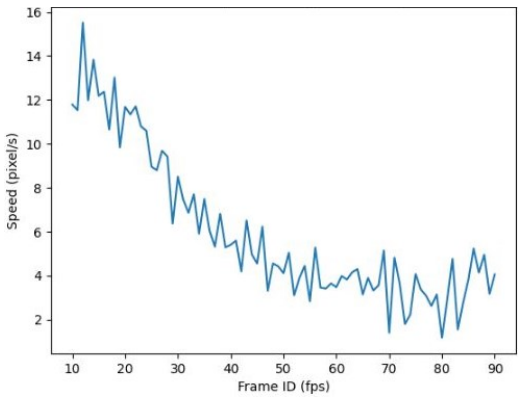



Fitur Kecepatan

**Lampiran 49** *Pola berkendara objek ID B13  7*

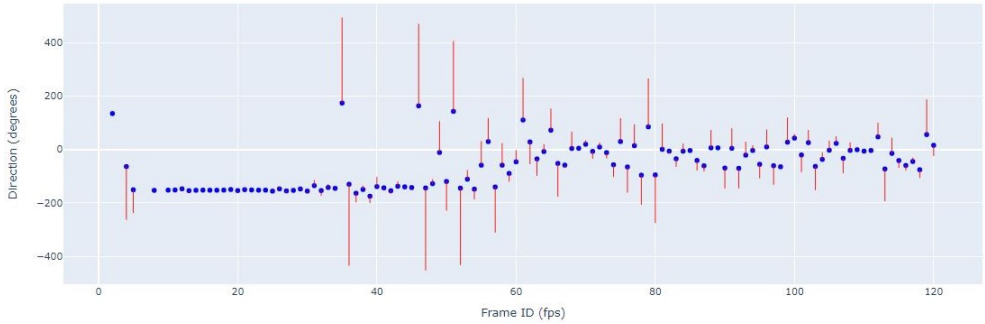



Fitur Arah

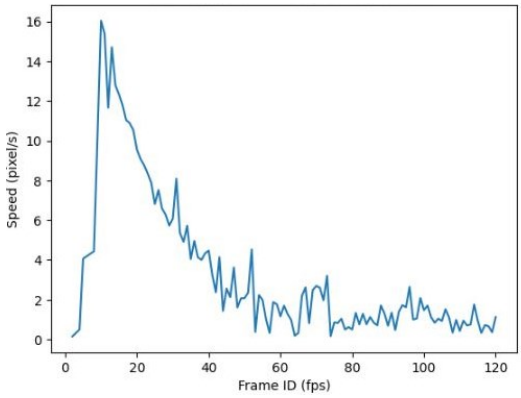



Fitur Kecepatan

**Lampiran 50** *Pola berkendara objek ID B14  1*




Fitur Arah




Fitur Kecepatan

**Lampiran 51** *Contoh code proses deteksi objek dengan model YOLOv5s*

```
    # Draw bounding boxes and labels on the frame
    for (x, y, w, h) in filtered_boxes:
        # Draw bounding box rectangle
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    return frame, filtered_boxes

# Load the YOLOv5 model
# model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
# model.eval()

# Modify the video_path according to the location of the video you want to detect
video_path = "/content/drive/MyDrive/YOLOV5/video/18_SIANG_ANTAR MOTOR_TABRAK_BLUR 1.mp4"
output_path = "/content/drive/MyDrive/YOLOV5/percobaan/output_video18.mp4"  # Path to save the output video

cap = cv2.VideoCapture(video_path)
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

# Initialize count
count = 0
center_points_prev_frame = []

tracking_objects = {}
track_id = 1
```

**Lampiran 52** *Contoh code proses pemberian ID kendaraan*

```
# Increment idle frames for unmatched objects and remove objects with too many idle frames
tracking_objects_copy = tracking_objects.copy()
for object_id in tracking_objects_copy.keys():
    if object_id not in matched_ids:
        idle_frames[object_id] += 1
        if idle_frames[object_id] > max_idle_frames:
            tracking_objects.pop(object_id)
            idle_frames.pop(object_id)

# Assign new IDs to unmatched objects
for pt in center_points_cur_frame:
    if pt[1] > 50:
        # Check if the object is a shadow or irrelevant object based on width-to-height ratio
        is_shadow = False
        width_height_ratio = pt[0] / pt[1]
        if width_height_ratio > 3 or width_height_ratio < 0.3:
            is_shadow = True

        # Find the smallest available ID
        if not is_shadow:
            while track_id in tracking_objects:
                track_id += 1
            tracking_objects[track_id] = pt
            idle_frames[track_id] = 0
            matched_ids.append(track_id)
            track_id += 1
```

**Lampiran 53** *Contoh code proses pelacakan fitur menggunakan Farneback*

```
    # Calculate optical flow
    flow = cv2.calcOpticalFlowFarneback(prev_gray, curr_gray, None, 0.5, 3, 15, 3, 5, 1.2, 0)

    # Update tracking results with optical flow
    for object_id, pt in tracking_objects.items():
        if object_id in matched_ids:
            # Get the coordinates for optical flow calculation
            x = int(pt[0])
            y = int(pt[1])
            dx = flow[y, x, 0]
            dy = flow[y, x, 1]

            # Calculate speed and direction
            speed = math.sqrt(dx**2 + dy**2)
            angle = math.atan2(dy, dx) * 180 / math.pi

            # Write tracking results to CSV file
            csv_writer.writerow([count, object_id, speed, angle, 0])
```

**Lampiran 54** *Contoh code proses klasifikasi menggukan SVM Gaussian RBF*

```
# import HalvingGridSearchCV
from sklearn.experimental import enable_halving_search_cv  # Perlu diimpor untuk mengaktifkan fitur ini
from sklearn.model_selection import HalvingGridSearchCV
from sklearn.svm import SVC

# instantiate classifier with default hyperparameters
svc = SVC()

# declare parameters for hyperparameter tuning

parameters = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf','linear'],
              'degree':[1,2,3,4,5,6]}

SVM = HalvingGridSearchCV(estimator=svc,
                                    param_grid=parameters,
                                    scoring='accuracy',
                                    cv=5,
                                   random_state=0,
                                    factor=3,
                                    verbose=2)

SVM.fit(X_train, y_train)

print("Best parameters found:")
print(SVM.best_params_)
print("Best accuracy found:", SVM.best_score_)
```