**Capstone Project**
Machine Learning Engineer Nanodegree

**Andrejs Zujevs**
Apr 21st, 2017
Riga

**Quadcopter Automatic Landing Detection**

# I. Definition

## Project Overview

Drones are more formally known as unmanned aerial vehicles. Essentially, a drone is a flying robot and a quadcopter is a specific type of drone, propelled by four motors (see Figure 1). Quadcopters and other multicopters can fly autonomously for different purposes, for example, to deliver a pizza. Usually, an autonomous drone's take-off is from a home position (turn-on the motors and rise up into the air, in this project described as the drone's status 'Take-off'), flying to a defined destination point (described as the drone's status 'Flight'), then returning back to the home location and landing (described as the drone's status 'Land'). In this project, the word 'landing' means the drone's act of coming to the land/surface and then turning off the motors in time (simultaneously with touching the landing surface).

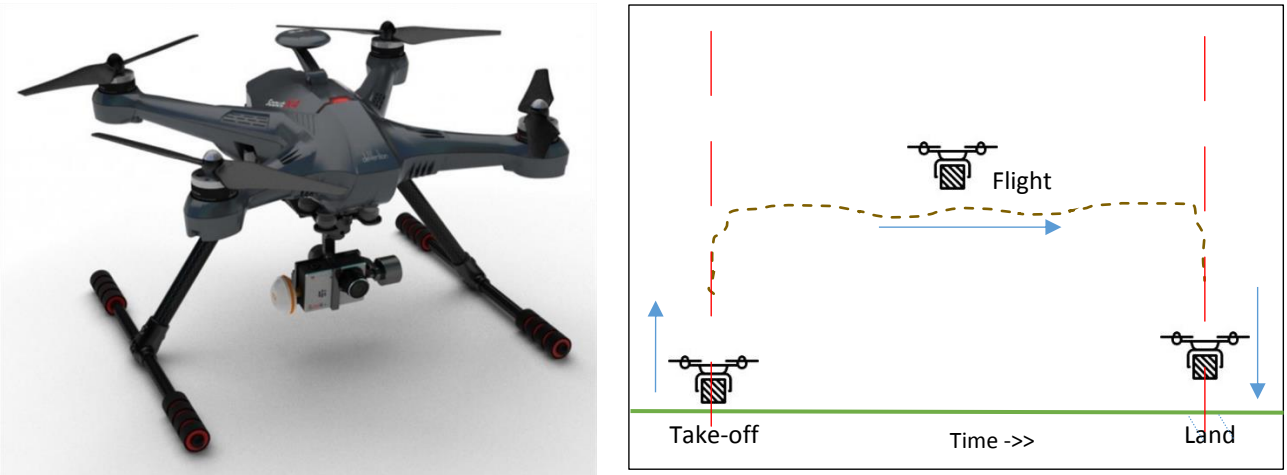



**Figure 1.** Quadcopter (from: http://www.hobbyflip.com) (left), drone' states (right)

In the reality, the autonomous landing is not an easy task for a drone due to the noise in sensors. Usually drones use a single sensor or a combination of them to measure the actual distance to the landing surface and turns off the motors only when it touches the landing surface or is very close to it. For example, the ultrasound, LIDAR, and an optical sensor (stereo vision) are generally used to measure the distance between the drone and the surface. The most common problems for a drone's autonomous landing are as follows:

1) noise in the sensor(-s) due to the impact of the other electronics in a drone;
2) uneven surface at the landing location; sensors may measure various altitudes, although in truth the drone is at the same altitude;
3) slope of the landing surface and the drifting of the drone to other location;
4) inaccuracy in detecting the location of the landing caused by the inaccuracy of GPS sensor or low GPS signal;
5) dirt on the sensors (dust, water, etc.) due to environmental conditions.

The above-mentioned problems can cause the following issues for drone developers and users:

1) the drone's propellers may be damaged, due to its drifting to one side and probable collision with an obstacle or landing in a high grass, which in turn would damage the propellers;

2) drifting of the drone (propellers not turned off in time) may cause an injury (cut, bruise, etc.) to its user, other people, or animals;
3) the drone may cause damage to the property of a third party if the its landing position is measured inaccurately due to GPS signal problems or noise in sensors.

The dataset available for this project consists of 25 short-time flights (with one or several landing events during a flight). The total size of the dataset is ~88K records with 21 features (various sensors' data and data derived from sensors as well as from motors). A large size of the dataset is explained with frequency of data recordings (approximately every 30$^{th}$ millisecond). Each flight has three categories of data records which are correspond to the drone's states: Take-off, Flight and Land (in accordance with the example in Figure 1). The sequence of the drone statuses may be repeated many times if a drone landed or its landing was canceled more than once during a flight. Also, the moment the drone touches the landing surface (time interval ~0.5 sec) is manually labeled for each landing event where the drone's status is Land.

## Problem Statement

**The aim of this project** - create an algorithm/approach which will detect the time moment of a drone touching a surface during the landing. The final algorithm should use only the actual/current data from sensors and motors in the given moment **without using** data from the external sensors, such as LIDAR, the ultrasonic, or the vision sensor due to the previously mentioned problems. Also, the final algorithm should be used real-time, with limited computation resources.

The defined problem may be solved by supervised learning; having the touch moment labeled, we would only need to construct a classification model that can predict the touch moment with the landing surface.
First, we need to explore the dataset, then detect outliers and remove theirs, if necessary. Factor Analysis (FA) or Independent Component Analysis (ICA) will reduce the number of features, allowing us to construct a classification model by using K-Means or GMM (Gaussian Mixture Models).

## Metrics

In the case of the supervised learning, we need to take into account FP (False Positives) and FN (False Negatives), because FP will prematurely turn off the motors and the drone may crash from high altitude. Then again, FN will delay the motors' turn-off and the drone may drift and/or cause injury to people or animals. Finally, a more appropriate metric is F1-score due to its weighted average of precision and recall. Precision score tp/(tp + fp) is partly appropriate of this problem, however it does not give any measurements for FN (than a touch moment need to be predicted). Thanks for recall score – tp/(tp +fn).

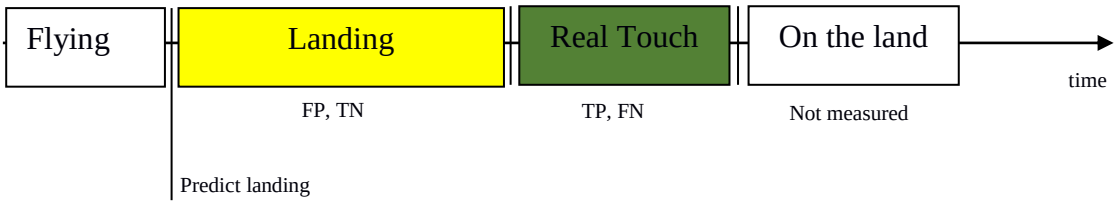



**Figure 2.** Predicting the touch moment during the drone landing

# II. Analysis

*Data Exploration*

The dataset consists of 88084 records with 21 features:

```
BAR             float64 # Barometer's sensor data
MOTOR1            int64 # Rotation speed of motor 1, rpm
MOTOR2            int64 # Rotation speed of motor 2, rpm
MOTOR3            int64 # Rotation speed of motor 3, rpm
MOTOR4            int64 # Rotation speed of motor 4, rpm
LIDAR           float64 # LIDAR's sensor data, m
THRUST          float64 # Thrust of motors (mixed), rate
ACC_X           float64 # Accelerometer's sensor data (acceleration in X axis) m/s^2
ACC_Y           float64 # Accelerometer's sensor data (acceleration in Y axis) m/s^2
ACC_Z           float64 # Accelerometer's sensor data (acceleration in Z axis) m/s^2
MAG_Z           float64 # Magnetometer's sensor data (magnetic flow in Z axis), rate
GYRO_X          float64 # Gyroscope's sensor data (orientation in X axis), rate
GYRO_Y          float64 # Gyroscope's sensor data (orientation in Y axis), rate
GYRO_Z          float64 # Gyroscope's sensor data (orientation in Z axis), rate
PITCH           float64 # Drone pitch - internally estimated using different sensors
ROLL            float64 # Drone roll - internally estimated using different sensors
YAW             float64 # Drone yaw - internally estimated using different sensors
TIME            float64 # Log's record time in microseconds from the logging start
LOG_ID            int64 # Log file ID
LAND_STATUS       int64 # Status - has the drone actually touched the surface or not (in
~0.5s time frame)
STATUS           object # Drone's state
```

Samples of the dataset:

| | BAR | MOTOR1 | MOTOR2 | MOTOR3 | MOTOR4 | LIDAR | THRUST | ACC_X | ACC_Y | ACC_Z | MAG_Z | GYRO_X | GYRO_Y | GYRO_Z | PITCH | ROLL | YAW | TIME | LOG_ID | LAND_STATUS | STATUS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1019.44 | 1298 | 1298 | 1298 | 1298 | 0.047 | 0.000 | -0.906 | 0.055 | -9.837 | 0.489 | 0.015 | 0.027 | -0.003 | -0.081 | 0.016 | 2.633 | 64927667 | 0 | 0 | take_off |
| **100** | 1019.29 | 1517 | 1624 | 1507 | 1531 | 0.298 | 0.686 | -0.116 | -0.083 | -13.008 | 0.457 | -0.221 | 0.003 | -0.055 | 0.083 | -0.015 | 2.614 | 66566750 | 0 | 0 | take_off |
| **200** | 1018.91 | 1437 | 1486 | 1445 | 1466 | 3.263 | 0.521 | 0.334 | -0.291 | -9.822 | 0.483 | -0.008 | 0.057 | -0.003 | 0.001 | 0.033 | 2.628 | 68336653 | 0 | 0 | take_off |
| **1073** | 1019.23 | 1425 | 1515 | 1396 | 1481 | 0.495 | 0.512 | -0.498 | -0.075 | -10.002 | 0.474 | 0.039 | 0.024 | -0.035 | -0.056 | 0.021 | -1.934 | 84077889 | 0 | 1 | land |
| **9999** | 1019.35 | 1513 | 1616 | 1509 | 1572 | 0.177 | 0.622 | -0.233 | -0.001 | -12.335 | 0.463 | -0.156 | -0.133 | -0.032 | 0.090 | -0.011 | 3.000 | 42166481 | 5 | 0 | take_off |
| **32050** | 1012.42 | 1535 | 1525 | 1461 | 1462 | 3.150 | 0.546 | 0.091 | 0.199 | -8.652 | 0.498 | 0.016 | 0.054 | -0.022 | 0.002 | -0.031 | -1.020 | 550829712 | 10 | 0 | flight |

The dataset consists of data time series, where each corresponds to the particular drone flying event. The feature's TIME values are measured in microseconds. The records' TIME when the drone's status is at TAKE-OFF has some shift due to a log file which started recording slightly before the take-off. Also, you can see that the velocity of the motors is represented with integer numbers (rpm), LIDAR (0 m – 10 m calibrated) and BAR sensor provides their raw scalar values. Thrust is scalar, and in the range [0.0, 1.0], 0.0 is the minimum value and the corresponding motors' rotation speed is 910 rpm.

Descriptive statistics of the dataset:

| | BAR | MOTOR1 | MOTOR2 | MOTOR3 | MOTOR4 | LIDAR | THRUST | ACC_X | ACC_Y | ACC_Z | MAG_Z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 | 88084.000 |
| **mean** | 1014.537 | 1466.298 | 1510.502 | 1441.252 | 1476.070 | 5.890 | 0.537 | -0.064 | -0.064 | -9.802 | 0.482 |
| **std** | 2.757 | 97.355 | 100.373 | 99.777 | 99.054 | 8.340 | 0.105 | 0.659 | 0.528 | 1.357 | 0.021 |
| **min** | 1009.240 | 910.000 | 910.000 | 910.000 | 910.000 | 0.000 | 0.000 | -20.925 | -7.646 | -47.693 | -0.519 |
| **25%** | 1012.170 | 1434.000 | 1480.000 | 1406.000 | 1445.000 | 0.059 | 0.509 | -0.378 | -0.361 | -10.153 | 0.472 |
| **50%** | 1013.980 | 1477.000 | 1517.000 | 1450.000 | 1483.000 | 4.887 | 0.546 | -0.057 | -0.097 | -9.775 | 0.484 |
| **75%** | 1017.410 | 1514.000 | 1559.000 | 1492.000 | 1526.000 | 7.956 | 0.585 | 0.235 | 0.191 | -9.383 | 0.494 |
| **max** | 1019.680 | 1879.000 | 1879.000 | 1819.000 | 1878.000 | 65.324 | 1.000 | 16.850 | 15.148 | 9.816 | 0.561 |

From the descriptive statistics you can see that the BAR sensor's values are in the range of 1009-1019, meaning that flight events were recorded in the same place and that this sensor's value is relative to the destination of the drone's flight. The LIDAR sensor's values are in the range of 0.0-65.32, although the sensor can accurately measure the altitude only within the range of 0.2 – 10 m. The values beyond these limits are incompatible with the sensor's sensitivity. The feature ACC_Z values are mostly negative due to the gap between the drone's acceleration on the Z axis and the gravity constant ($g \approx 9.8 m/s^2$).

*Exploratory Visualization*

The figures below depict sensors' values from the log ID=1. Figure 3 represents the motors, sensors' and barometer's values. As you can see, the barometer's value significantly decreased after the 80$^{th}$ second (drone is taking-off) and increased after the 185$^{th}$ second (drone is landing). In the 201$^{st}$ second, the drone touched the land surface and all four motors after that time point were rotating with relatively low velocity. For visualization purposes, the BAR sensor's values were shifted by -1.002+5e and scaled by 100.0.

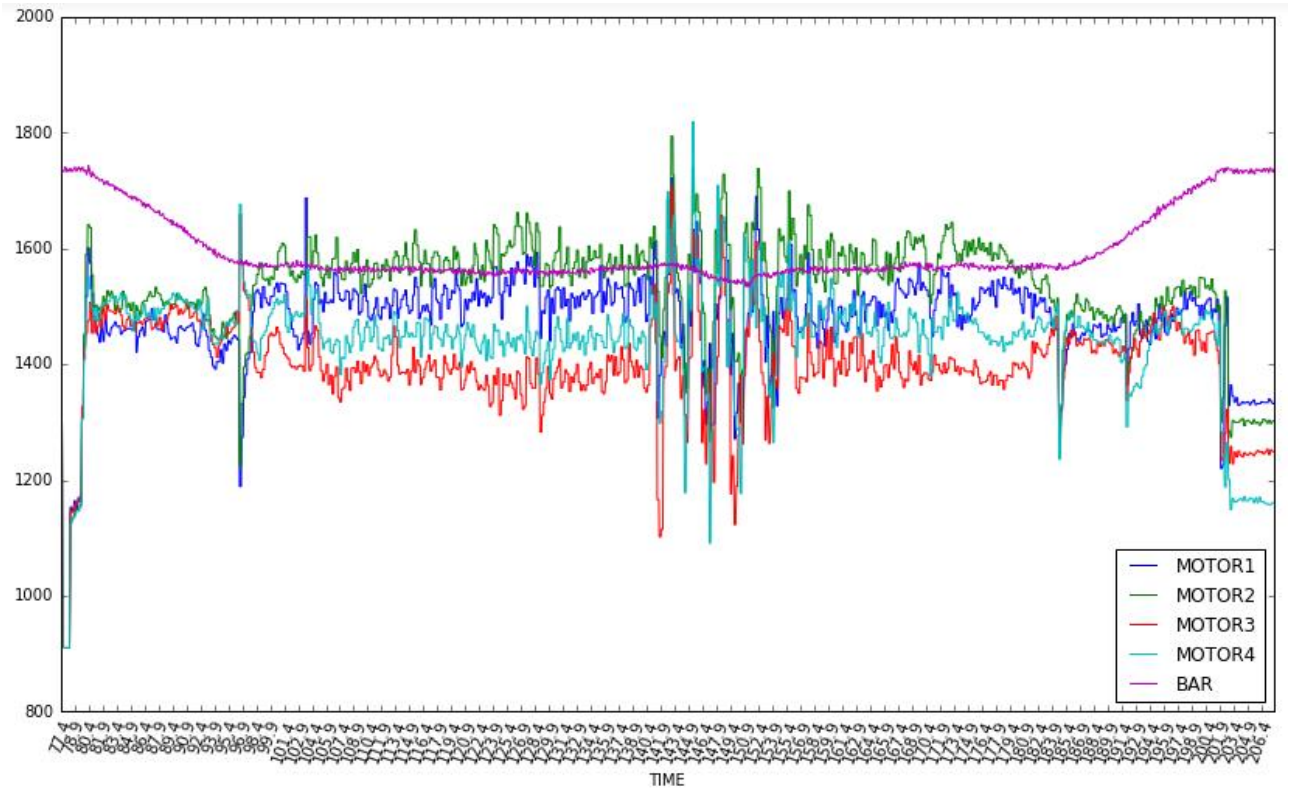



**Figure 3.** Log ID1's motors and barometer data during take-off, flight and landing

In Figure 4, values of the feature ACC_Z (accelerometer' value for the z-axis) are shifted by 8.0 (for the visualization purpose). The land-touching moment was recorded at the 201$^{st}$ second. There you can see high spikes for the ACC_X and the ACC_Y feature.

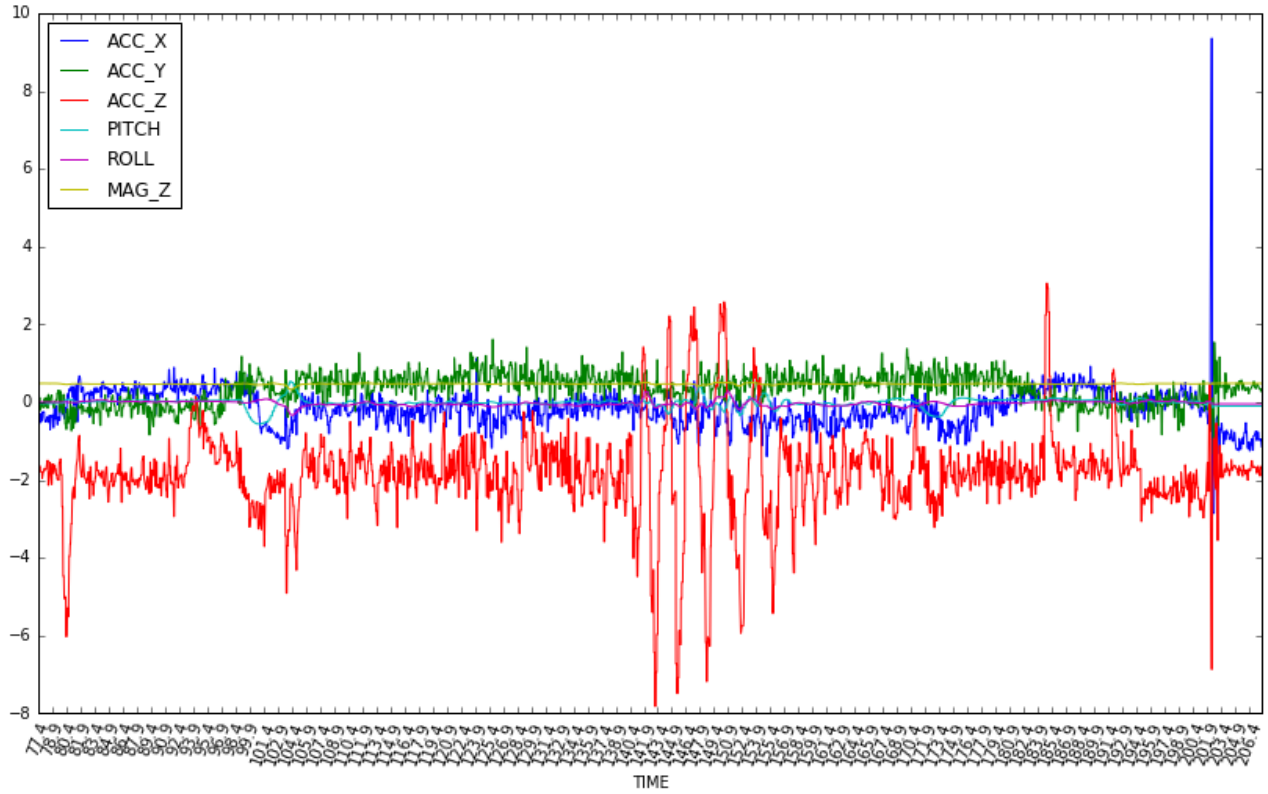
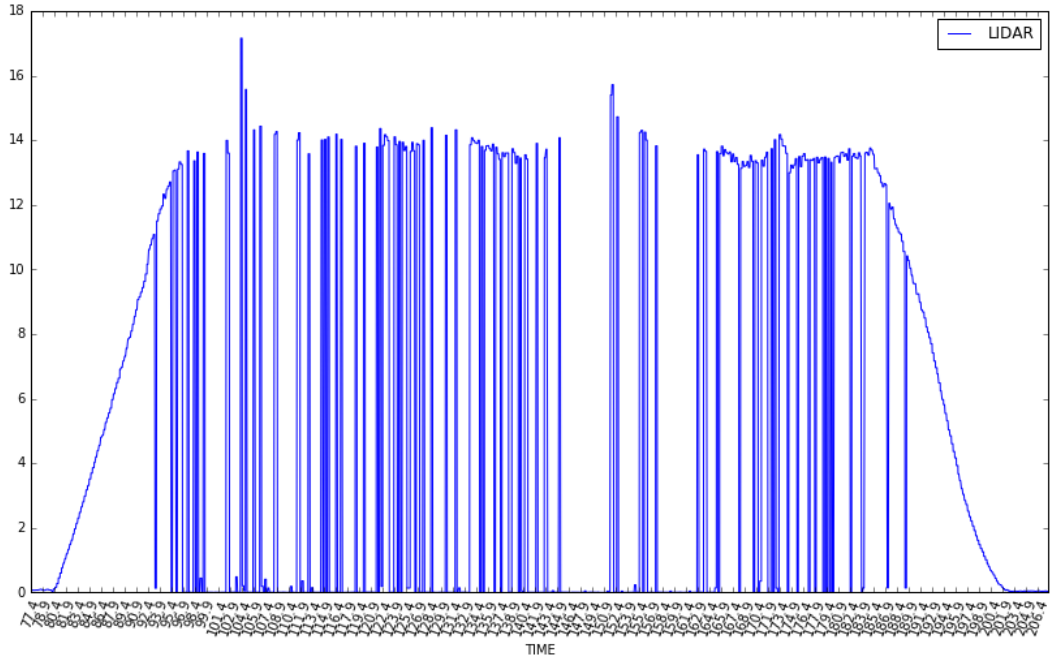



**Figure 4.** Log ID1's accelerometer and gyroscope data during take-off, flight, and landing

In the next figure, the LIDAR sensor's data are presented (also for log ID=1). Here, the sensor's values are scaled using 1.5 rate (for visualization purposes). The abundance of spike lines can be explained by the sensor's calibration range, where the sensor interpreted the altitude above 12 m




incorrectly.

**Figure 5.** Log ID1's LIDAR sensor data during take-off, flight, and landing

The figures below represent scatter plots with the estimation of each feature value's frequencies in the diagonal and data correlation between different features.

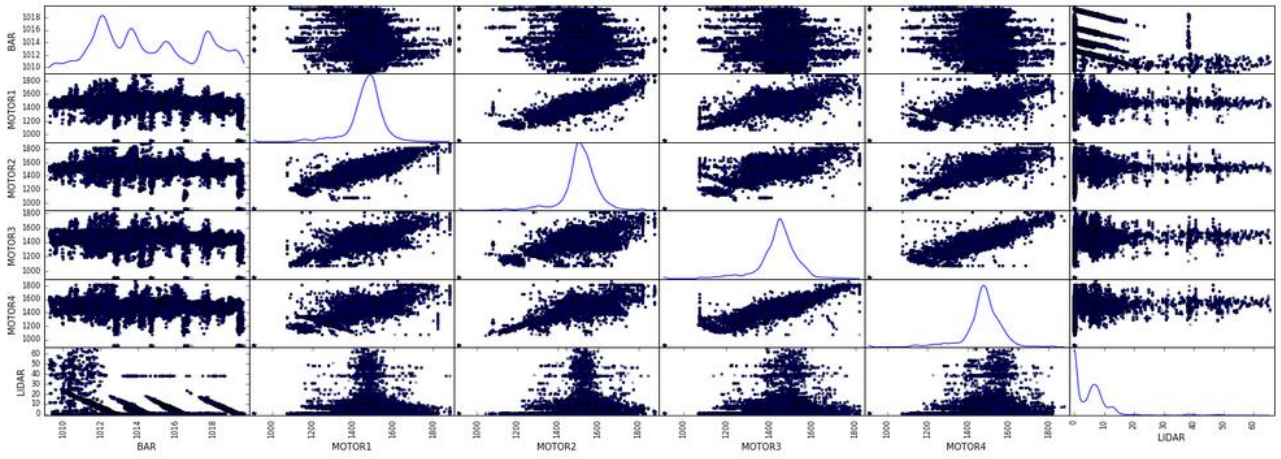



**Figure 6.** KDE plot for BAR, MOTOR[1-4], and LIDAR

In the figure above, some correlation of motors exists due to the motors rotations being synchronized by the flight controller.

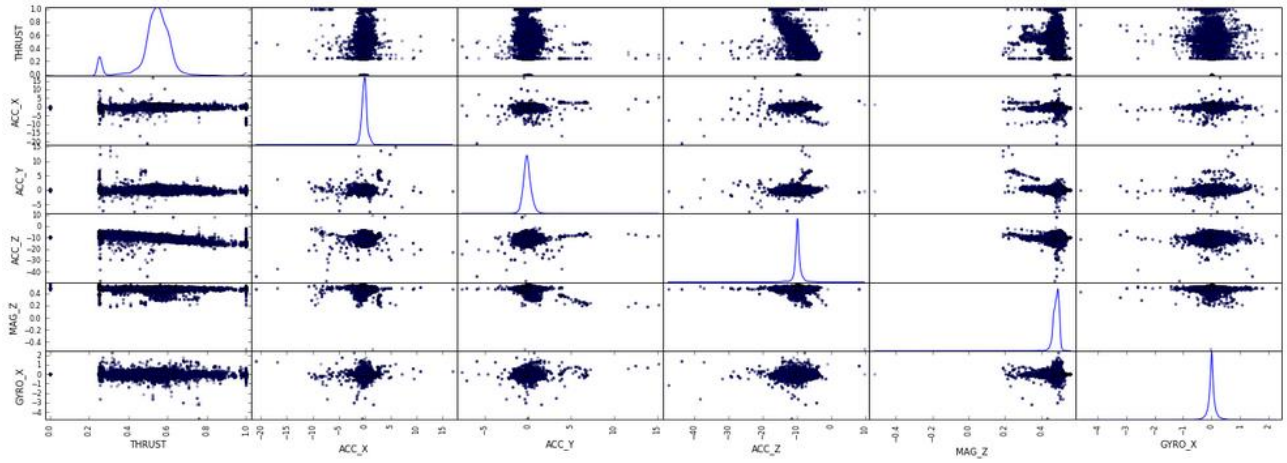



**Figure 7.** KDE plot for TRUST, accelerometer, magnetometer, and gyroscope

In the figure above, some correlation is present between the THRUST and ACC_X, ACC_Y, ACC_Z features.

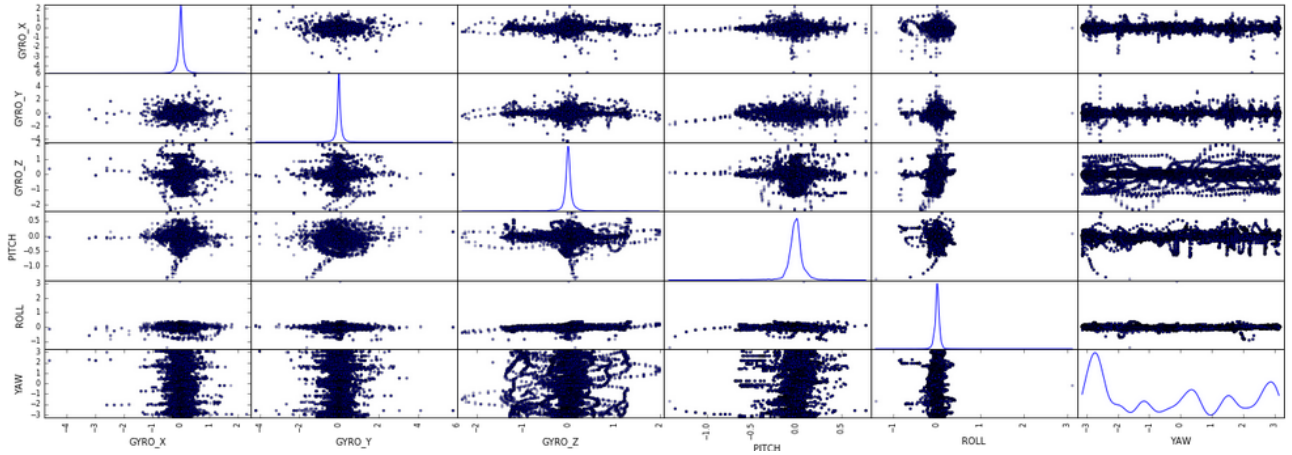



**Figure 8.** KDE plot for gyroscope, pitch, roll, and yaw

In the figure above, some correlation exists between all features due to the fact that PITCH, ROLL, and YAW are controlled by the drone flight controller, and the gyroscope state depends on it.

### *Algorithms and Techniques*

Firstly, data were cleared from noise using outlier detection technique and data visual analysis. Then, data were normalized using logarithm function, and some features were scaled to the positive value range as a result. Secondly, the analysis of relative features was carried out, and more relevant features were selected for further analysis. Because of the necessity to find an approach for the drone touch-moment detection, the drone being in state Landing, the Independent component analysis (ICA) and Factor analysis (FA) were used. Finally, the FA was used prior to the application of K-Means clustering.

**Dimension reduction** – ICA was applied to the dataset to reduce dimensionality, yet ICA requires normally distributed values and Shapiro-Wilk test verified that features' data are not normally distributed. Hence, FA was used for the reduction of dimensionality.

**Clustering** – K-means and Gaussian mixture models (GMM) clustering methods were used due to their simplicity, performance and they are most used clustering algorithms. Also, they are scalable for large datasets and use distances metric between points. For the project problem, number of clusters is known – 2 (whether the drone touches the landing surface during landing or not), and prior to the use of clustering we applied FA to the dataset in order to reduce the number of features.

**Performance measurement** – during the project, our own implementation of the F1-score, accuracy and recall score were provided. The main reason to write our own metrics measurement functionality is the time series which represents the state of the drone and particularly the touch moment during a landing.

### *Benchmark*

If the drone's automatic landing detection works with accuracy, equal or over 0.9 for the F1 score metric, this project solution will be acceptable and useful. If the accuracy is less than desired, the automatic landing will make a lot of False Positive events and the drone will crash from a high altitude during the landing, damaging itself, or drifting, and thus potentially causing injuries (cut, bruise, etc.) to the user, other people, or animals.

## III. Methodology

### Data Preprocessing

During this project, we need to select relevant features for further analysis and data transformations. Also, we need to take into account the final solution performance of the drone to be able to process the real-time data. For ICA and FA, the dataset was normalized using the logarithm function. Since the logarithm function requires values over zero, the selected features were scaled to the positive values' range (in case originally they were negative). Also, the outliers detection was applied to some features, using the visual dataset analysis.
The feature "Status" (the drone's state) was transformed into two separate binary features: STATUS_flight, STATUS_land, and STATUS_take_off.

Linear regression (using DecisionTreeRegressor) was applied to determine the relevant features (see Table 1).

**Table 1.** Relevant features using linear regression

| Nr. | Feature | Score value | Nr. | Feature | Score value |
|---|---|---|---|---|---|
| 1 | BAR | 0.981 | 11 | MAG_Z | 0.972 |
| 2 | MOTOR1 | 0.977 | 12 | GYRO_X | 0.718 |
| 3 | MOTOR2 | 0.983 | 13 | GYRO_Y | 0.755 |
| 4 | MOTOR3 | 0.979 | 14 | GYRO_Z | 0.914 |
| 5 | MOTOR4 | 0.983 | 15 | PITCH | 0.910 |
| 6 | LIDAR | 0.759 | 16 | ROLL | 0.895 |
| 7 | THRUST | 0.948 | 17 | YAW | 0.953 |
| 8 | ACC_X | 0.823 | 18 | STATUS_flight | 1.000 |
| 9 | ACC_Y | 0.831 | 19 | STATUS_land | 1.000 |
| 10 | ACC_Z | 0.871 | 20 | STATUS_take_off | 1.000 |

For further analysis, these were used: MOTOR1, MOTOR2, MOTOR3, MOTOR4, THRUST, ACC_X, ACC_Y, ACC_Z, MAG_Z, PITCH, and ROLL. The selection of all the features were generally based on data exploratory visualization and some relevant assumptions from prior experience or expert knowledge, the linear regression analysis relatively provides the same results for all features; and only few features, such as LIDAR, ACC_*, GYRO_*, have smaller correlation values, and yet they are not as insignificant to claim that other features do not impact them.
The feature BAR was not used for further analysis because its values depends on a place where the drone flying. The LIDAR sensor's values are too noised to be trusted in the proximity of one meter (due to the specification of the sensor). The values of the GYRO_* features were not used as well, since the pitch, roll, and yaw features can be more precisely estimated and depends on mentioned features.
The feature STATUS_* are only needed for the data separation and results validation, yet they are not primary used for the data analysis. All features values scaled to the range of 0.0 – 255.0.
Alter all the outliers detected using the following rules: 'MOTOR1': (40.868, 250.447), 'MOTOR2': (53.579, 264.316), 'MOTOR3': (23.536, 274.104), 'MOTOR4': (40.489, 259.557), 'THRUST': (34.542, 235.443), 'ACC_X': (75.0, 190.0), 'ACC_Y': (40.0, 175.0), 'ACC_Z': (75.0, 220.0), 'MAG_Z': (150.0, None), 'PITCH': (80.0, 245.0), 'ROLL': (25.0, 125.0), where in parenthesis the actual range of values is given. Afterwards, all the data were scaled (see Fig. 8 and 9).

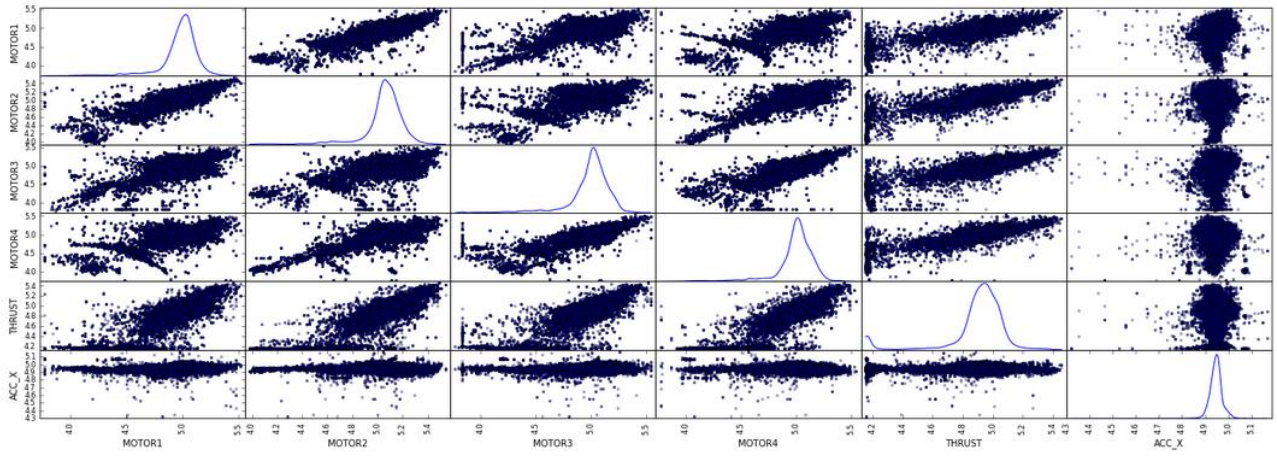



**Figure 9.** Data with removed outliers. KDE plot for BAR, MOTOR[1-4], and LIDAR

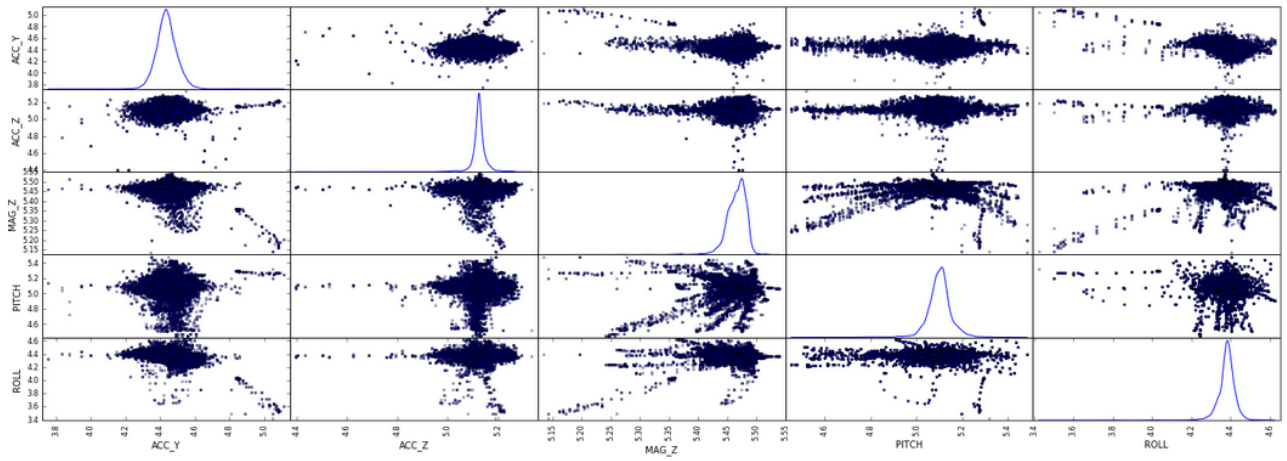



**Figure 10.** Data with removed outliers. KDE plot for gyroscope, pitch, roll, and yaw

FA was applied on all preprocessed data (for drone state Landing) and only the initial 5 components were used for further analysis, since their features' values have relatively high impact values (see Figure 11 and example of log after FA in Figure 12).

In the Figure 11, in the 1-th component, you can see strong correlation between the thrust and the speed of the motors. As the thrust decreases, so the motors' speed also decreases. The second and third component have a more diverse picture. The ACC_Z values increase for these components. Therefore, being described only by the properties of the sensor calibration process, we need to multiply ACC_Z value with a negative one to better interpret them. After all, in the first five components, the motors' values change differently, because of Flight Controller will always trying to stabilize the drone horizontally during the landing. On the other hand, changes in other features (pitch, roll, yaw, acc*) usually differs before and after the drone touches the landing surface.

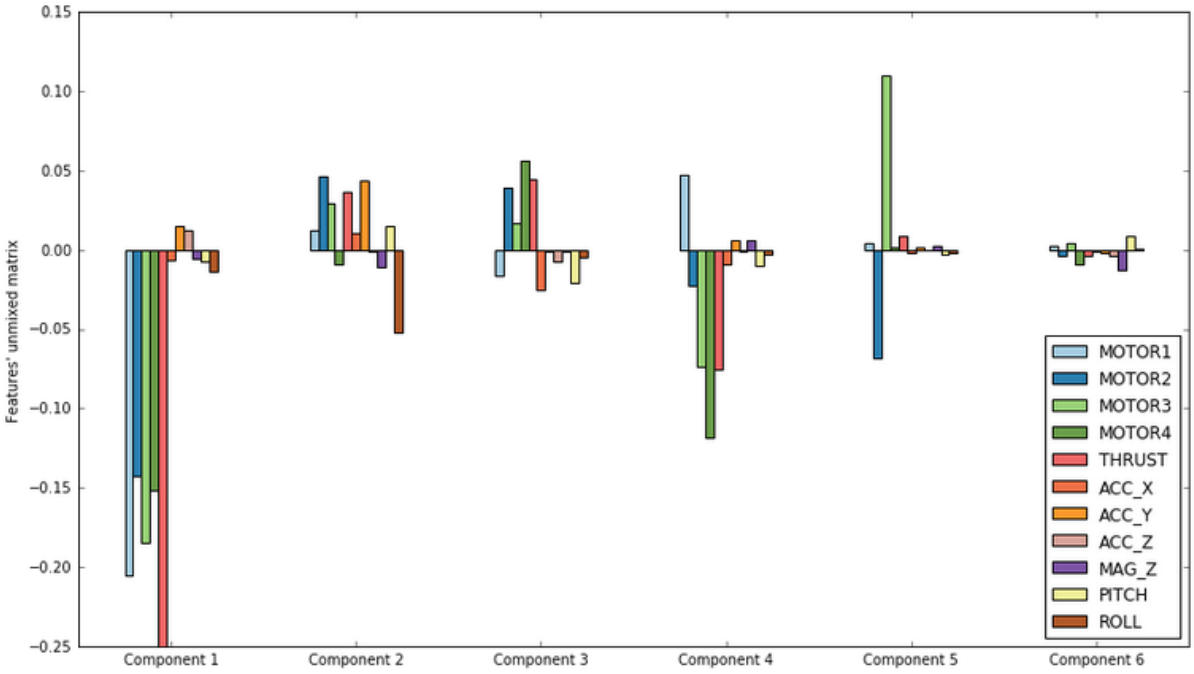



**Figure 11.** FA results

In the Figure below, you can see the FA transformation result for the data of logID=19, where the drone touched the landing surface in 164.80-165.34 sec.

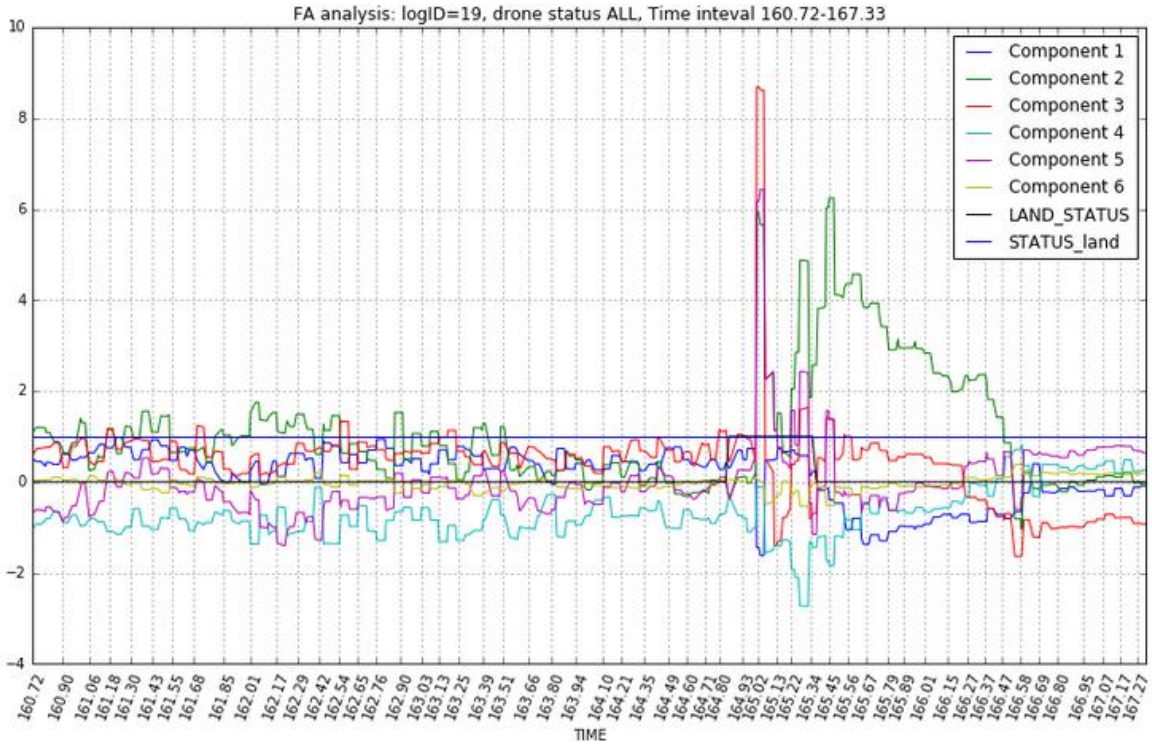



**Figure 12.** Example: LOGID=19 data representation after applying FA.

## Implementation

### Performance metrics

As was previously mentioned, the following scores were used for the performance measurement: the F1 score, the precision and the recall scores. In this project, the author uses their own implementation of the all such metrics. This is due to the landing detection which may arise during a period of time (the data of the time series). These are the implementation details (see implementation in the *measure_accuracy()* method):

a) FP – estimated during the landing, before the drone touched the surface. Only single FP prediction value was necessary to satisfy a singular FP. Otherwise, a singular TN should be satisfied;
b) FN – estimated during the real touch moment. If no landings were predicted, the singular FN should be satisfied, otherwise only a singular TP.

Clustering

K-Means and GMM were applied after FA transformation on the whole data. The dataset was divided into the training (logid - [0, 11]) and testing (logid - [12, 24]) sub-sets. In the figure below, the clustering results are shown:

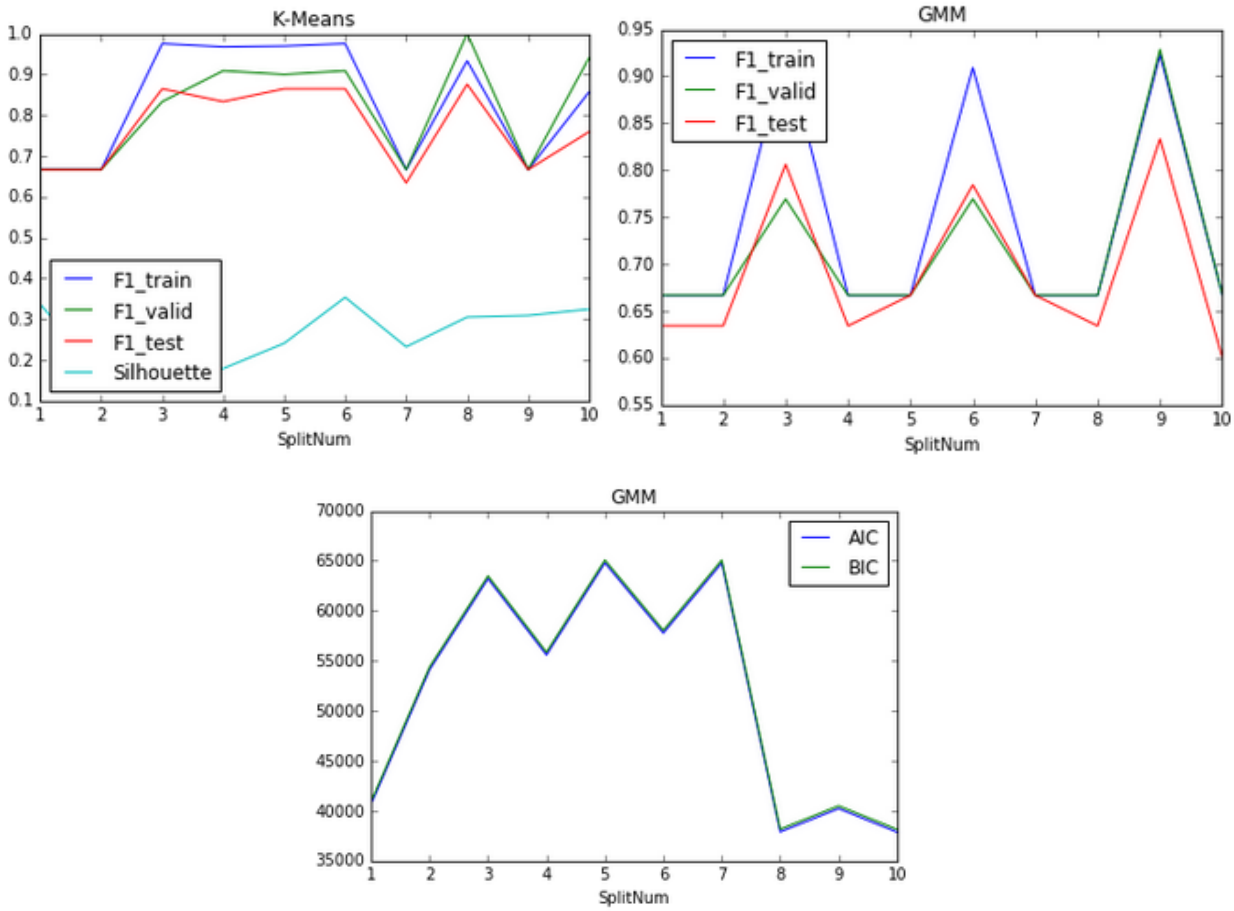



**Figure 13.** K-Means and GMM clustering results for the 10 splits. Metrics reported: f1_score, recall, and precision. For testing, only data from the testing subset were used.

The best results (testing) with K-Means were obtained in the 8-th split by the F1 score (0.88); and with GMM it was done in the 9-th split by the F1 score (0.83). The visual representation of the clustering results is shown below.

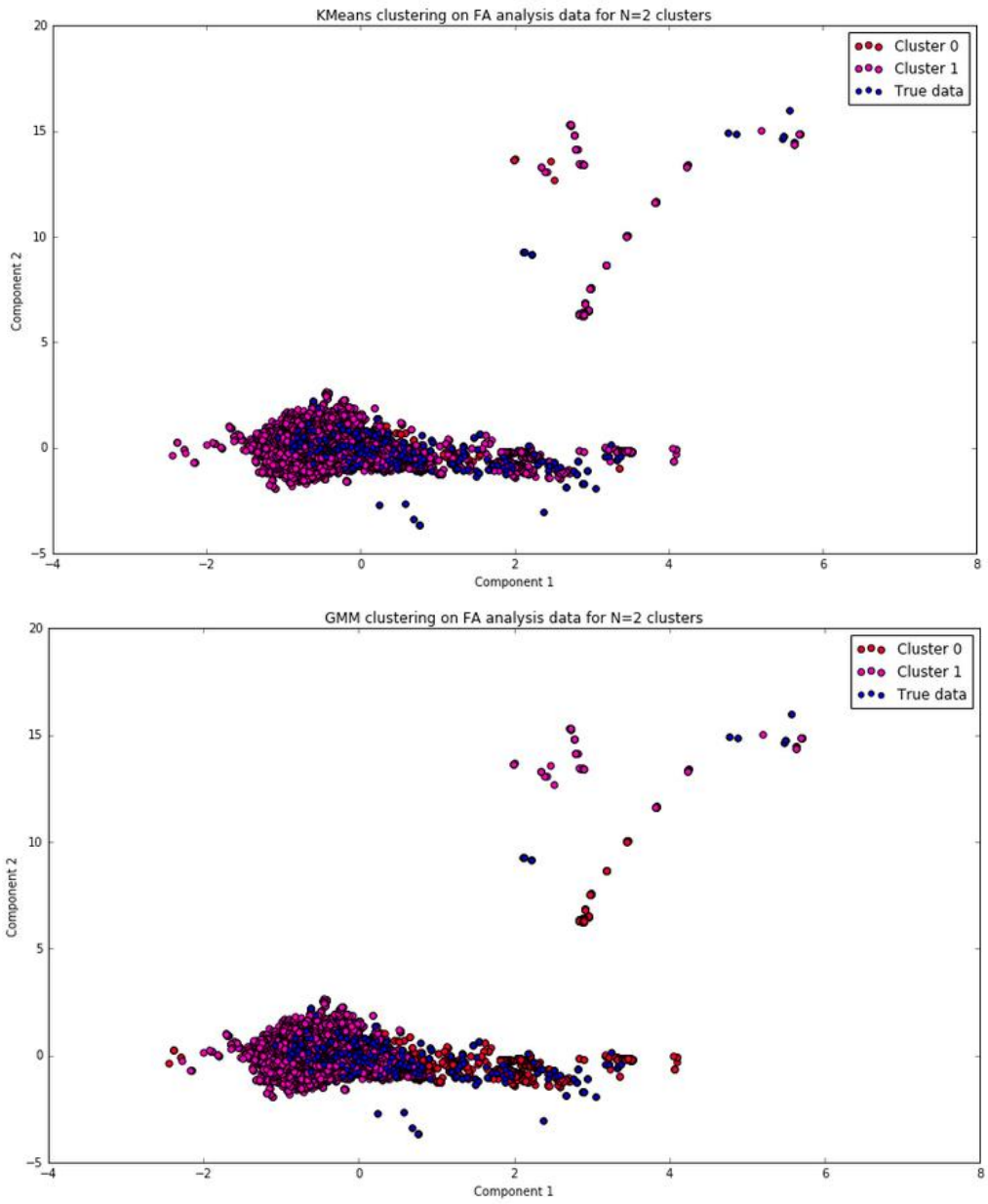



**Figure 14.** Initial K-Means and GMM clustering results and component's 1,2 visualization, where true data is marked for real-time touching

### *Refinement*

K-Means is the type of a GMM. During refinement, the covariance type ‘tied’ for GMM was selected for F1 score value, which is finally equal to 0.941. This result was obtained within 200 iterations.

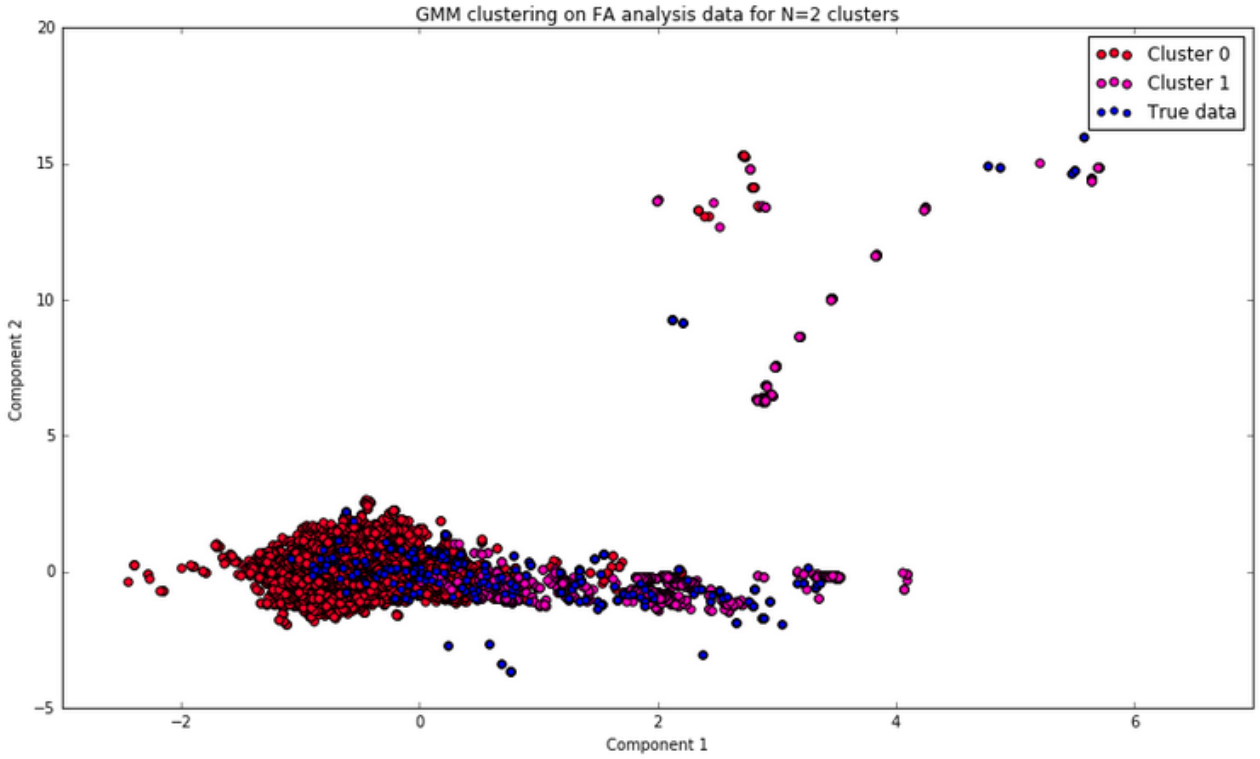



**Figure 15**. GMM clustering results after refinement, where true data is marked for real touching moment

As you can see in Figure 15, true data are smoothed for both clusters, but we also need to take into account that such true point also represents drone landings, as data has a time series type.

## IV. Results

### Model Evaluation and Validation

The final model (using GMM clustering) completely aligns with the expectations of the project and provides satisfactory performance results at 0.941 (the F1 score) on unseen data within 10 logs with 12 real- touch moments. There were small changes in the training data set, they do not give high impact of the results (see Table 2).

**Table 2.** GMM clustering results during 10 folds for F1 score

| **Description/ Dataset** | **Training dataset, logs - [0, 11]** | **Validation dataset, logs - [0, 11] only 35%** | **Testing dataset, logs [12, 24]** |
|---|---|---|---|
| **Mean** | 0.89 | 0.84 | 0.85 |
| **Std** | 0.12 | 0.09 | 0.10 |
| **Max** | 0.98 | 0.92 | 0.93 |
| **Min** | 0.67 | 0.67 | 0.67 |

### Justification

The final results are stronger than benchmark results, since the GMM was trained with 'tied' covariance type and K-Means is only a case of GMM. Although K-Means clustering was provided better results during the final model tuning.
F1 score value of 0.941 for unseen data is satisfactory and completely fitting with the benchmark requirements. Also, a small variation in the training data did not significantly impact the model training results and the standard deviation for the testing dataset was determined only at 0.1 of the F1 score for all 10 folds.
Since the dataset of this project belongs to the time series data type, the clustering techniques are a more adequate approach than, for example, SVM or Decision Trees.

## V. Conclusion

In Figure 16, you can see how well the final model works with the unseen data. There are no False Positives before the real touch moment other than logid=14, where the touch moment is detected a little earlier, by~0.20 sec. However, the mistake is not actually significant, if we consider the drone's landing speed to be commonly low.

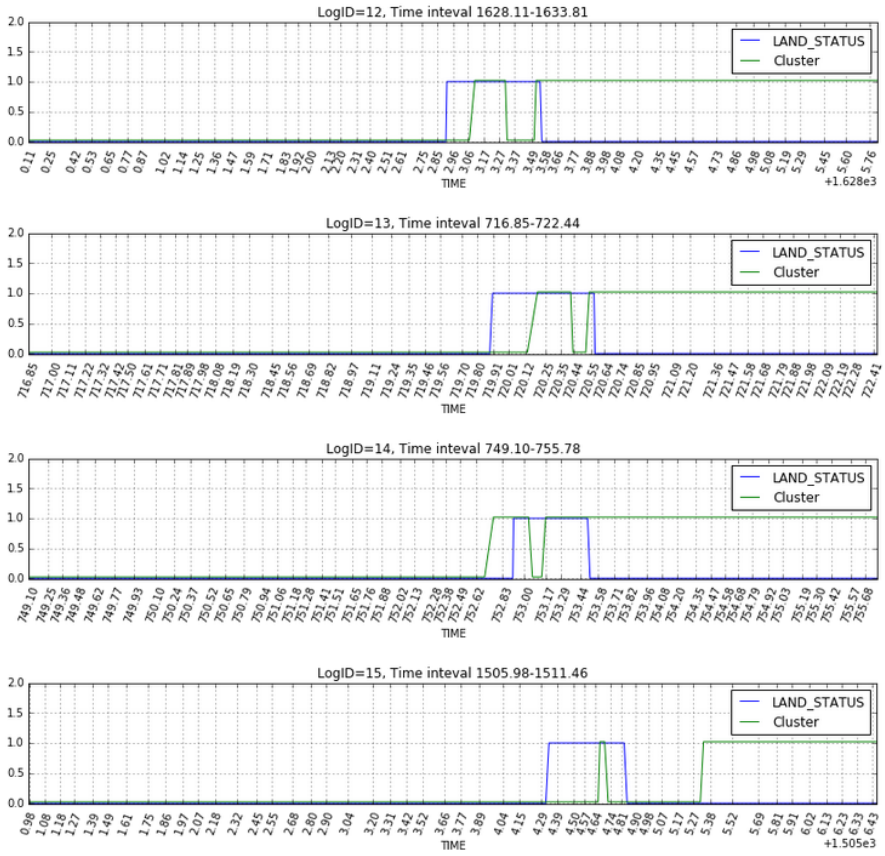



**Figure 16.** The drone touching moment prediction on unseen data for logs 12, 13, 14, and 15

### Reflection

This project's problematics are still topical in the drone industry. More and more drone manufactures are trying to use machine learning techniques and algorithms to make decisions on real-time cases and having minimal computation resources at their disposal. This project is no exception and attempts to solve the automatic landing detection problem.

One of the most problematic and interesting aspects of this project is the data type time series, but since we need to detect the touch moment we cannot use time series as input data for prediction purposes due to the fact that the decision time is strongly limited and we need to detect the touching moment with the landing surface as soon as possible.
Dataset preparation and preprocessing takes a great deal of time, yet the modeling task required the largest amount of time. Different algorithms were tried: SVM, Decision Making models, K-Means, GMM (the first two were rejected due to bad benchmark results). The author also implemented the performance metrics in this project.
The final model fit with the project expectations and the benchmark of the project, and finally provides a 0.94 F1 score on the unseen data. The critical moment for this project might have been to find a trade-off between False Positives and True Positives. The first could cause the drone to crash, and the second would cause its drifting, both being negative occurrences.
GMM is a fast clustering algorithm, and scaling training dataset will not significantly impact the training time. This gives more opportunities for future improvements.

### Improvement

I believe that improvements may occur in the modeling stage - where training data will have additional features and as such would allow to do clustering better. Also, the largest training and testing dataset will provide more robust results.
Techniques that would be interesting to implement in the future are the time series pattern-matching and the learning algorithms. If we use a relatively small-time window, for example, 0.3 sec for predicting the drone’s touching moment, the current final benchmark value could be improved.