<u>How to use the anomalies detecting option:</u>

The application has an option to alarm about anomalies that occurred during the flight.

It is allowed by an anomaly detecting algorithm that is loaded via dynamic linked library. The algorithm can learn a normal flight data file, find the features that are correlated do each other and by that, in a real time flight, it can detect any deviation from the normal pattern.

(while setting a certain threshold so it will not alarm on small deviations underneath that threshold).

We supplied two different detecting algorithms:

1. The algorithm using linear regression for predicting the future values and if the observed value is different from the prediction it will alert.
2. The algorithm finds the circle with the minimal radius that contains all the standard points so when getting a new point, it measures the distance from the center of the circle and if its bigger than the radius it will alert.

You can find those two algorithms in the pulgins folder. While running the application, if the user would like to see the alerts of the anomalies in his flight, he will be able to choose which algorithm he would like to use and also can upload to the application an algorithm of his own.

For that, he can upload a .dll file of the algorithm that must implement the functions bellow, and will also need to upload a csv file with data of a normal flight so the algorithm will be able to learn it.

```
IntPtr timeseries_Create([MarshalAs(UnmanagedType.LPStr)] string path);

IntPtr Anomaly_Detecor_Create();

void Anomaly_Detecor_LearnNormal(IntPtr sd, IntPtr ts);

IntPtr Anomaly_Detecor_Detect(IntPtr sd, IntPtr ts);

int Anomaly_Detecor_getReportSize(IntPtr report);

IntPtr Anomaly_Detecor_getAnomaly(IntPtr report, int index);

IntPtr Anomaly_Detecor_getAnomalyString(IntPtr cptr);

void Anomaly_Detecor_DeleteAnomaly(IntPtr cptr);
```

(documentation of the functions in the next pages)

```
/*

* Returns a pointer to a TimeSeries object.

* Basically it's an object that gets a path to a csv file with flight data

* when the first line is the features names, each line is a moment in the

* flight with data for each feature.

*/

IntPtr timeseries_Create([MarshalAs(UnmanagedType.LPStr)] string path);



/*

* Returns a pointer to an anomaly detector object.

*/

IntPtr Anomaly_Detecor_Create();


/*

* Gets a pointer to a timeseries object, that contains data of a normal flight,

* and a pointer to an anomaly detector object, so it will learn the normal flight

* for detecting devietions later.

*/

void Anomaly_Detecor_LearnNormal(IntPtr sd, IntPtr ts);


/*

* Gets a pointer to a timeseries object, that contains data of a flight to detect anomalies in
it,

* and gets a pointer to an anomaly detector object that will detect the anomalies in this
flight.

* The function returns a pointer to an anomaly report - a data structure which contains the
anomalies.

*/

IntPtr Anomaly_Detecor_Detect(IntPtr sd, IntPtr ts);
```

```
/*
* Gets a pointer to an anomaly report object an returns the size of it.
* (The amount of anomalies in the flight)
*/
int Anomaly_Detecor_getReportSize(IntPtr report);



/*
* Gets a pointer to an anomaly report and an index and returns
* a pointer the the anomaly in this index.
*/
IntPtr Anomaly_Detecor_getAnomaly(IntPtr report, int index);

/*
* Gets a pointer to an anomaly object and returns the string describing the anomaly.
* the format is: "feature1-feature2-anomalyTime"
*/
IntPtr Anomaly_Detecor_getAnomalyString(IntPtr cptr);

/*
* Gets a pointer to an anomaly object and frees the allocated memory.
*/
void Anomaly_Detecor_DeleteAnomaly(IntPtr cptr);
```