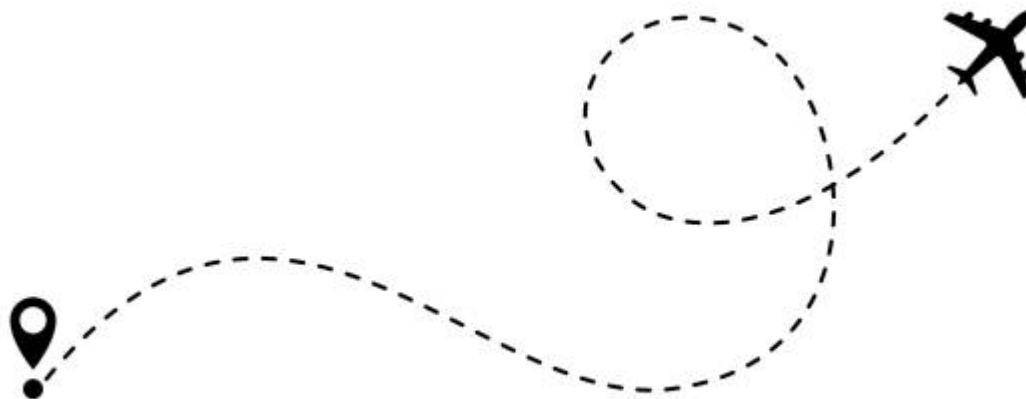


Predicting True Air Speed from Other Sensors

This example shows how to use historical data to train a regression model that is capable of predicting the True Air Speed (TAS) of a flight from the measurements of other sensors. Such models can be useful in controls applications where it may be desirable to use a regression model as a nonlinear state estimator for a non-observable or costly-to-observe state. These types of models can also be used as surrogate models in simulations and trade-studies, in cases where a physics-based model is too complex to create or takes too long to simulate.



Import data from Excel

Start by reading in the data with code generated by the Import Tool. Also remove data recorded when the aircraft was stationary on the ground as that will not be useful for predicting airspeed.

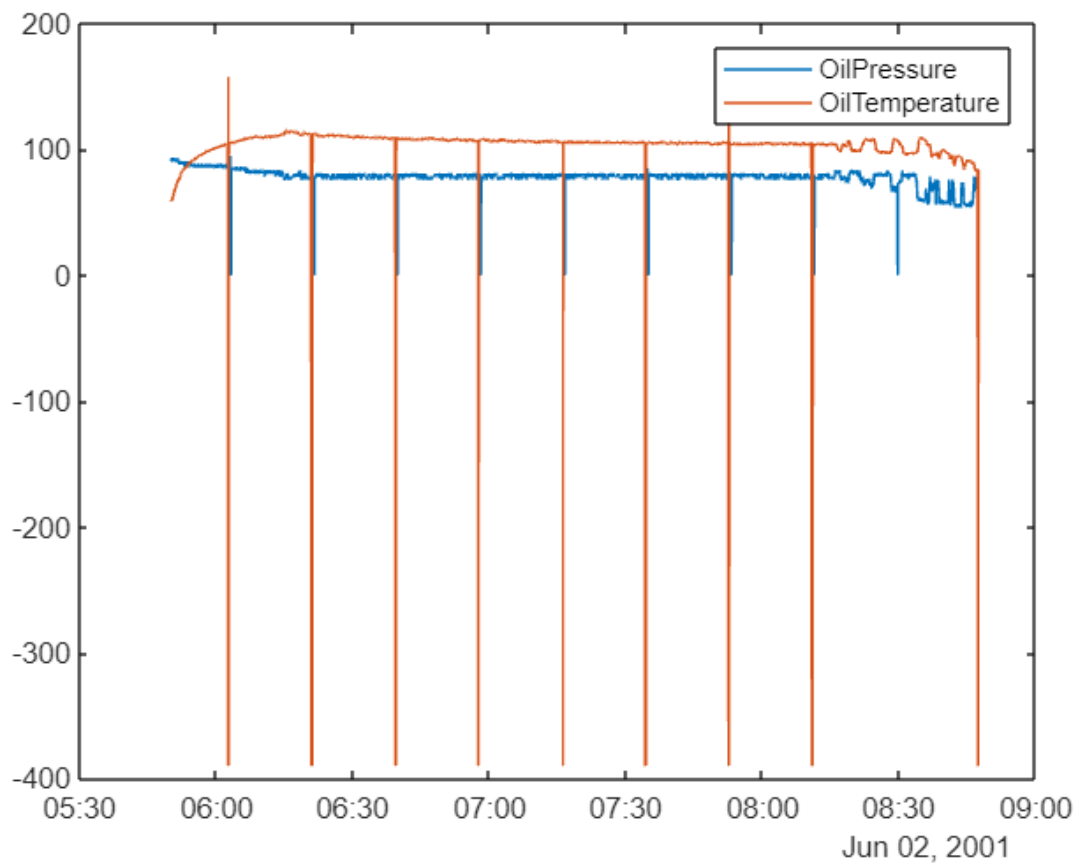
Time	FuelQuantity	OilPressure	OilTemperature	LatitudePosition	LongitudePosition
02-Jun-2001 05:50:04	7966.8	90.354	58.805	44.891	-63.508
02-Jun-2001 05:50:05	7968	90.354	58.595	44.891	-63.508
02-Jun-2001 05:50:05	7968	90.354	58.595	44.891	-63.508
02-Jun-2001 05:50:05	7968	90.354	58.595	44.891	-63.509
02-Jun-2001 05:50:05	7968	90.354	58.595	44.891	-63.509
02-Jun-2001 05:50:06	7968	90.354	58.595	44.891	-63.509
02-Jun-2001 05:50:06	7968	90.354	58.595	44.891	-63.509
02-Jun-2001 05:50:06	7968	90.354	58.595	44.891	-63.509

Visualize data

We can see the flight path by viewing a geoplot of latitude and longitude.

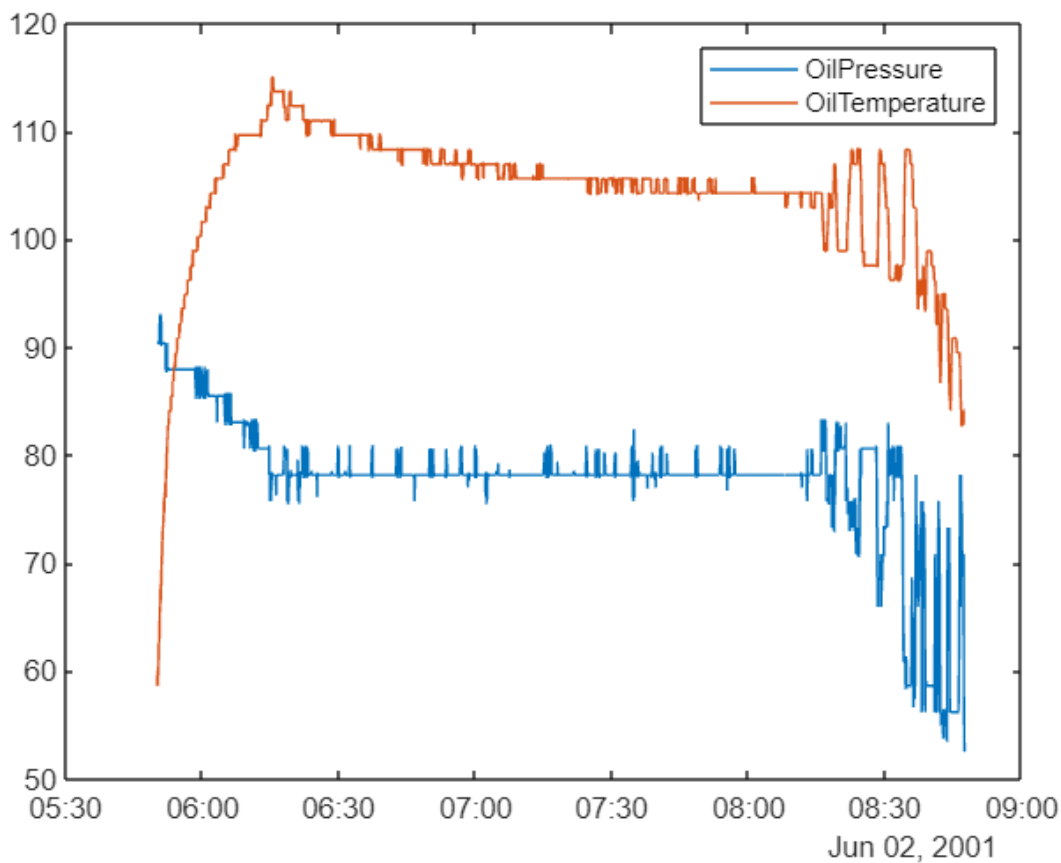


Some of the sensors appear to have dropouts that we'll need to clean.



Clean sensor data

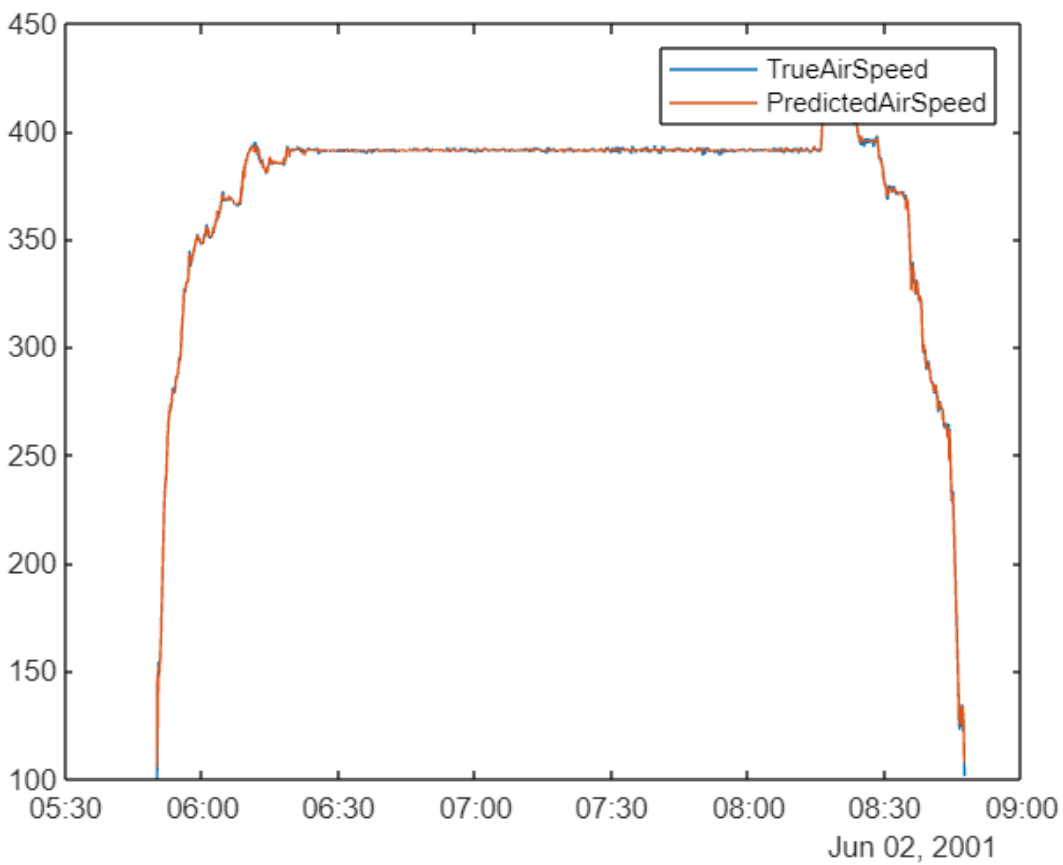
Use the Data Cleaner to remove the sensor dropouts for oil pressure and temperature.



Model True Air Speed

We can model TAS with a regression model trained in the Regression Learner app. We'll use a bagged decision tree with default settings, which produces very accurate cross-validated results. Normally in a machine learning workflow we'd use a separate hold-out test set to verify results, but we'll skip that here for brevity.

```
mdl = struct with fields:
    predictFcn: @(x)ensemblePredictFcn(predictorExtractionFcn(x))
    RequiredVariables: {'Altitude' 'ExhaustTemperature' 'FanSpeed' 'FuelFlow' 'FuelQuantity' 'OilPressure' 'OilTemperature' 'TrueAirSpeed'}
    RegressionEnsemble: [1x1 classreg.learning.regr.RegressionBaggedEnsemble]
    About: 'This struct is a trained model exported from Regression Learner R2022a.'
    HowToPredict: 'To make predictions on a new table, T, use: yfit = c.predictFcn(T) replacing 'c' with mdl'
```



Predictor importance

Which measurements are important for making predictions? One benefit of the bagged decision tree model is it automatically provides a measure of predictor importance.

FuelFlow is the most important predictor in this case, followed by WindDirection, WindSpeed, and FanSpeed.

