Los Impredixibles

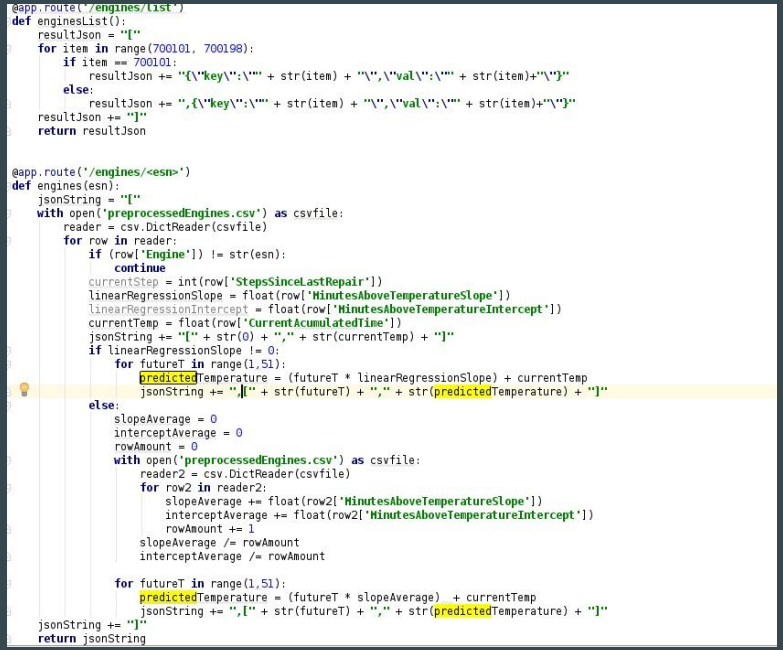
...

Summary

- We used linear regression to predict in which cycle will the engine probably fail, using the above temperatures minutes vs cycle as the variables used in the regression.
- We created two views for the user one contains routes and cost analysis and the second view contains the predictive analysis.

MicroServices

- Two MicroService for the creation of the data in JSON format.
- All data was preprocessed in a Python Script that generated a CSV file for the MicroService to process.



```
@app.route('/engines/list')
def enginesList():
    resultJson = "["
    for item in range(700101, 700198):
        if item == 700101:
            resultJson += "{\"key\":\"" + str(item) + "\",\"val\":\"" + str(item)+"\"}"
        else:
            resultJson += ",{\"key\":\"" + str(item) + "\",\"val\":\"" + str(item)+"\"}"
    resultJson += "]"
    return resultJson


@app.route('/engines/<esn>')
def engines(esn):
    jsonString = "["
    with open('preprocessedEngines.csv') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            if (row['Engine']) != str(esn):
                continue
            currentStep = int(row['StepsSinceLastRepair'])
            linearRegressionSlope = float(row['MinutesAboveTemperatureSlope'])
            linearRegressionIntercept = float(row['MinutesAboveTemperatureIntercept'])
            currentTemp = float(row['CurrentAcumulatedTime'])
            jsonString += "[" + str(0) + "," + str(currentTemp) + "]"
            if linearRegressionSlope != 0:
                for futureT in range(1,51):
                    predictedTemperature = (futureT * linearRegressionSlope) + currentTemp
                    jsonString += ",[" + str(futureT) + "," + str(predictedTemperature) + "]"
            else:
                slopeAverage = 0
                interceptAverage = 0
                rowAmount = 0
                with open('preprocessedEngines.csv') as csvfile:
                    reader2 = csv.DictReader(csvfile)
                    for row2 in reader2:
                        slopeAverage += float(row2['MinutesAboveTemperatureSlope'])
                        interceptAverage += float(row2['MinutesAboveTemperatureIntercept'])
                        rowAmount += 1
                    slopeAverage /= rowAmount
                    interceptAverage /= rowAmount

                for futureT in range(1,51):
                    predictedTemperature = (futureT * slopeAverage)  + currentTemp
                    jsonString += ",[" + str(futureT) + "," + str(predictedTemperature) + "]"
    jsonString += "]"
    return jsonString
```

Visualizations

- Two visualizations:
  - One for predictive data (using the linear regression).
  - One for descriptive data (route and cost analysis).

```
<px-card header-text='Menu 2'>
    <h3>Second Card</h3>

    <px-dropdown id="enginesDropdown" display-value="700101" selected-key="700101"><px-dropdown-content extend-dropdown='true' extend-dropdown-by='25' max-cont-character-width='10' items="
    <px-chart data="{{engineData}}" zoom-controls="hidecontrols" tooltip-type="normal"><px-chart-series-line id='hpt-acc-position-cruise' data="{{engineData}}"></px-chart-series-line></px-c

</px-card>
```

```
define(['angular', './sample-module'], function(angular, sampleModule) {
    'use strict';
    return sampleModule.controller('SampleCtrl2', ['$scope', '$http', function($scope, $http) {
        function fillEnginesDropdown($scope,$http) {
            $http({
                method: 'GET',
                url: 'https://inpredixibles-microservice.run.aws-usw02-pr.ice.predix.io/engines/list',
                headers: {

                }
            }).
            success(function(data){
                $scope.engines = data;
                console.log($scope.data);
                console.log('Successful call');
            });
        };

        function getDataForEngine($scope,$http,engineNumber) {
            $http({
                method: 'GET',
                url: 'https://inpredixibles-microservice.run.aws-usw02-pr.ice.predix.io/engines/' + engineNumber,
                headers: {

                }
            }).
            success(function(data){
                $scope.engineData = data;
                console.log($scope.data);
                console.log('Successful call');
            });
        };
        fillEnginesDropdown($scope,$http);
        getDataForEngine($scope,$http,700101);
        document.getElementById('enginesDropdown').addEventListener('dropdown-content-value_changed', function(e) {
            var selectedEngine = e.detail;
            $scope.selectedEngine = selectedEngine.textValue;
            console.log($scope.currentESN);
            getDataForEngine($scope, $http, $scope.selectedEngine);
        });


    }]);
});
```