User Interface

CAPSTONE PROJECT | ASSIGNMENT 7

In this part of the capstone project we will build a User Interface to visualize the output of our raw data, our metrics and our Machine Learning predictions.

1. CREATING A RISK SCORE METRIC

Risk Score

Before we create our UI, there is one metric that we need to add so that we can populate the UI with our risk prediction scores. Create a new `RiskScore_SmartBulb.json` file in the `seed/SimpleMetric` folder:

```
{
  "id" : "RiskScore_SmartBulb",
  "name" : "RiskScore",
  "description": "Percentage value indicating risk of SmartBulb failure",
  "srcType" : "SmartBulb",
  "expression" : "identity(currentPrediction.prediction/365)"
}
```

In the expression you can see that we are dividing by 365. We do this in order to create a disaggregated value of 0 to 1 for our risk score since the metric is evaluated as a daily sum over a one year period.

2. FOLDER STRUCTURE

UI development starts with the creation of a new UI application, and to do this we will need to create the proper folder structure. Add the following folders and placeholder files to your lightbulb application:

```
lightbulbTraining/
        seed/
                UIConfig/
                      c3ui.json
        ui/
                lightbulb/
                        dataSources/
                               Bulbs.c3ui
                        pages/
                                Home.c3ui
                        templates/
                                DashboardTemplate.c3ml
                        Application.c3ui
```

3. CREATING THE APPLICATION

Note: The .c3ui files follow standard JSON notation. Make sure you don't have any unnecessary line breaks, commas, or misplaced brackets. These will throw errors when provisioning.

The first step of creating your UI is to define the application:

Application.c3ui

```
ui module lightbulb {
 application lightbulb {
 "name": "lightbulb",
 "humanName": "Smart Bulb Failure Prediction",
 "showDesignerComponents": true,
 "url": "lightbulb",
 "id": "lightbulb.lightbulb",
 "theme": "base",
 "package": "lightbulbTraining",
 "defaultPage": "Home",
 "thumbnail": "template-thumbs/120/ui.Dashboard2x2.gif",
 "noneditable": false,
 "moduleName": "lightbulb"
 }
}
```

4. CREATING THE PAGE TEMPLATE

Now that we have defined the application, we want to make a home page for the application. Before we can create the page, however, we must create a template specifying where all of our components will live on the page:

DashboardTemplate.c3ml

```
{{ div.container-fluid.c3-page.c3-page-dashboard({cls: cls}) do }}
   <div class="row">
       <div class="c3-component-socket col-xs-12 col-sm-3" id="LeftCol"></div>
       <div class="col-xs-12 col-sm-9" id="RightCol">
           <div class="row">
               <div class="c3-component-socket col-xs-12" id="RightColRow1"></div>
           </div>
           <div class="row">
               <div class="c3-component-socket col-xs-12" id="RightColRow2"></div>
           </div>
           <div class="row">
               <div class="c3-component-socket col-xs-12" id="RightColRow3"></div>
           </div>
       </div>
   </div>
{{ end }}
```

5. ADDING A NEW DATA SOURCE

After the new UI page template has been created, we will start developing components.

We want to visualize a number of things, such as:

- The risk scores in a histogram
- The location and their risk status of our light bulbs on a map
- A list of light bulbs and details about them on a scrollable grid
- A filter based on light bulb manufacturer and bulb type

First off, we need to create a data source that will be feeding all of our components in this UI application.

Bulbs.c3ui

```
ui module lightbulb {
  dataSource Bulbs   {
     "collection":true,
      "c3function":"fetch",
      "c3arguments":{
         "spec":{
            "limit":"-1",
            "include":"currentPrediction.prediction, manufacturer, bulbType, 
startDate, manufacturer, bulbStatus, latitude, longitude"
         }
      },
      "c3type":"SmartBulb",
      "name":"Bulbs",
      "responseSelector":"objs",
      "record":false,
      "responseTransform":{
         "combine":false,
         "rootProperty":false,
         "firstItem":false,
         "order":false,
         "tuples":false,
         "fields":false
      },
      "id":"lightbulb.Bulbs"
   }
}
```

7. CREATING A HOME PAGE

Now that we have a page template and data source, we need to make a home page for our components to live on. We will later configure our components in this file as well, but first we must make specifications about the page itself.

Home.c3ui

```
ui module lightbulb {
  page Home   {
      "id":"lightbulb.Home",
      "name":"Home",
      "template":"lightbulb.DashboardTemplate",
      "thumbnail":"template-thumbs/120/ui.Dashboard2x2.gif",
      "title":"Home",
      "url":"home",
      "components":[
      ]
    }
}
```

7. CREATING A HISTOGRAM COMPONENT

We will not be making any custom components in this project, but instead using some of C3's out of the box components. This makes the process a little easier, as we do not need to create any component files.

Instead, we can simply configure existing C3 components in the `Home.c3ui` file. The first component we will add is the “Histogram”.

Copy the following code into to your `Home.c3ui` file, inside the square brackets following "components".

Home.c3ui

```
{
   "id": "HistogramChart",
   "component": "chart.MetricsHistogram",
   "icon": "bar-chart",
   "name": "HistogramChart",
   "box": true,
   "title": "Failure Risk",
   "numberPoints": "-1",
   "numberBins": "10",
   "renderTo": "#RightColRow1",
   "height": 556,
   "titleUrl": null,
   "data": {
       "itemsDataSource": "lightbulb.Bulbs"
   },
   "metricNameTitle": true,
   "metricNameTooltip": true,
   "chartingType": "count",
   "binningType": "value",
   "labelDecimals": null,
   "worstPerformingLegendText": null,
   "period": null,
    "start": null,
    "pointPath": null,
    "pointWidth": "25",
    "nameField": null,
    "countText": null,
    "maxValueText": null,
    "nameText": null,
    "valueText": null,
    "percentileText": null,
    "drilldownBackText": null,
    "noHistogramDataMessageText": null,
    "noItemDataMessageText": null,
    "errorDataMessageText": null,
    "ignoreZeroValues": false,
    "displayXAxisLegend": false,
    "tooltipBeneathColumns": false,
    "currentMomentEval": false,
    "enableDrilldown": false,
    "histogramMetric": {
        "metricName": "RiskScore",
        "metricTitle": "RiskScore"
    },
    "colors": [
        "#4f6e04",
        "#31b82e",
        "#74f03a",
        "#f5df16",
        "#ecf029",
        "#f6f79c",
        "#ffa462",
        "#f27201",
        "#ff6d4d",
        "#e34400"
    ]
},
```

8. CREATING A STATUS MAP

After creating our histogram, let's create another component, the "Status Map". This map will show us the location of all the light bulbs and color light bulbs red that are above a certain risk threshold (50% in this case).

Copy the following code into the `Home.c3ui` file, directly after the code for the "Histogram" component.

Home.c3ui

```
{
   "id": "Map",
   "component": "Map",
   "icon": "map-marker",
   "name": "Map",
   "box": true,
   "title": "Status Map",
   "autoLoadData": true,
   "lat": "latitude",
   "lng": "longitude",
   "autocenter": true,
   "cluster": false,
   "mapConfig": {
       "mapTypeId": "roadmap"
   },
   "components": [
       {
           "component": "Tooltip",
           "id": "PhantomTooltip",
           "renderTo": ".phantom-tooltip"
       }
   ],
   "renderTo": "#RightColRow2",
   "height": 556,
   "markerColors": [
       {
           "id": "1",
           "color": "#fc3d00",
           "display": "if 'currentPrediction.prediction > 0.5' then color =
#fc3d00",
           "field": "currentPrediction.prediction",
           "value": "0.5",
           "comparator": ">"
       }
   ],
   "useInfoWindowTpl": false,
   "data": {
       "collection": "lightbulb.Bulbs"
   },
   "allowGeoFilter": false,
   "showNumberOfRecords": false
},
```

9. CREATING A LIGHT BULB DETAIL LIST

We now have two components, one that shows us our risk of failure distribution in a histogram and one that shows a map of light bulbs, indicating which light bulbs are above a certain risk threshold.

Our third component will be a list of details relating to individual light bulbs, displaying information on:

- bulb ID
- bulb type
- manufacturer
- start date
- risk score

Copy the following code into `Home.c3ui`, after the code for the Map component.

Home.c3ui

```
{
   "id": "Grid",
   "component": "KendoGrid",
   "icon": "align-justify",
   "name": "Grid",
   "box": true,
   "paginate": true,
   "title": "Lightbulb Details",
   "data": {
       "collection": "lightbulb.Bulbs"
   },
   "filterable": false,
   "renderTo": "#RightColRow3",
   "height": 556,
   "titleUrl": null,
   "aggregate": null,
   "small": false,
   "checkBoxes": false,
   "columns": [
       {
           "id": "id",
           "field": "id",
            "label": "Bulb ID"
        },
        {
             "id": "bulbType",
            "field": "bulbType",
            "label": "Bulb Type"
        },
        {
            "id": "manufacturer.id",
            "field": "manufacturer.id",
            "label": "Manufacturer"
        },
        {
            "id": "startDate",
            "field": "startDate",
            "label": "Start Date",
            "format": "time"
        },
        {
            "id": "currentPrediction.prediction",
            "field": "currentPrediction.prediction",
            "label": "Risk Score",
            "format": "humanize"
        }
    ]
},
```

10. CREATING A FILTER PANEL COMPONET

Having all these light bulbs in our network, we would like to be able to filter the dashboard on Manufacturer and Bulb Type. We will do this by creating a “Filter Panel” component.

Copy the following code into `Home.c3ui`, after the code for the “Light Bulb Detail List” component.

Home.c3ui

```
{
   "id": "FilterPanel",
   "component": "FilterPanel",
   "icon": "lightbulb",
   "name": "FilterPanel",
   "box": false,
    "title": "Filter",
    "filterForm": {
        "component": "FilterForm",
        "filters": [
            {
                "id": "manufacturer.id",
                "field": "manufacturer.id",
                "label": "Manufacturer",
                "dataType": "string",
                "comparator": false,
                "component": "field.Text"
            },
            {
                "id": "bulbType",
                "field": "bulbType",
                "label": "Bulb type",
                "dataType": "string",
                "comparator": false,
                "component": "field.Text"
            }
        ],
        "id": "FilterForm"
    },
    "renderTo": "#LeftCol",
    "height": 556,
    "data": {
        "collection": "lightbulb.Bulbs"
    },
    "geoFilterText": "Geographical Filtering",
    "allowGeoFilter": false
}
```

11. MAKING "HOME" THE DEFAULT PAGE

The final piece of creating a successful UI for your lightbulb project is to configure the c3ui.json file. The code may look complex, but for our purposes were simply using it to ensure that our home page is the default application page.

c3ui.json

```
{
    "id": "uiconfig",
    "applications": [{
        "id": "lightbulb"
    }],
    "environments": [{
        "id": "development",
        "session": {
            "cache": "LocalStorage"
        },
        "locales": [
            {
                "id": "en",
                "name": "English"
            }
        ],
        "site": {
            "renderLogo": false,
            "hideNavigation": true
        },
        "connection": {
            "cache": false
        },
        "designer": true,
        "c3Tools": true,
        "acl": false
    }]
}
```

Congratulations, you have completed the lightbulb UI! Admire your application by visiting your URL with `/lightbulb`, or the URL you gave, at the end of it.

Example: `https://<tagName>-<tenantName>.c3-e.com/lightbulb`