

**Discovery Park**  
**Virtual Management Platform - Phase 2**  
**Project Manual**

Raymond Hall, Dwight Lewis, Austin Lin  
University of Washington MGIS 2019  
GEOG 569 Workshop  
August 24, 2019  
Manual provided for Friends of Discovery Park

# Table of Contents

<b>Executive Summary</b>	3
<b>1. Background and Project Objectives</b>	4
1.1 Background	4
1.2 Project Objectives	4
<b>2. Data Collector Web Applications</b>	6
2.1 Hosted Feature Layers	6
2.1.1 Setting Up Hosted Features for Data Collection	6
2.1.2 Visualization of Hosted Feature Layers	9
2.1.3 Privacy Settings for Hosted Features	10
2.2 Visualizing Layers in Web Maps	12
2.2.1 Setting Up Layers for Web maps	12
2.2.2 Adjusting Visualization	13
2.3.2 User-experience Widgets	16
2.3.3 Data Collection Widgets	17
2.3.4 Complementary Summary Widgets	19
<b>4. Recreational and Incident Reporting Platform</b>	25
4.1 Recreational Activities	26
4.1.1 Adding Recreational Activities	26
4.2 Incident Reporting	28
4.2.1 Adding Incidents	29
<b>5. “What-If” Analysis Application</b>	30
5.1 Data Management and Input Data Development	30
5.2.2 The geostatistical model	34
5.1.2 The simple web application model	37
5.2 Statistical Model Development and Deployment	39
5.3 Model Integration in ArcGIS AppBuilder	41
<b>6. Landing Page</b>	43
6.1 ArcGIS Experience Builder	43
6.2 How to Embed Image, Other Websites, and External Applications to Existing Website	44
<b>7. Recommendations</b>	45
7.1 Discovery Park Virtual Management Platform	45
7.1.1 Data Collection System	45
7.2 Professional Consultation	46
7.2.1 Strava Integration	46
7.3 Human-Nature Interaction	46
<b>APPENDICES</b>	48

Appendix 1: System Resource Requirements	49
Appendix 2: Business Case Evaluation	50
Appendix 3: Python Source Code for Model in “What-If” Analysis Application	59
Appendix 4: Data Dictionaries of the What-If Analysis	72

## Executive Summary

### Suggested key points to cover (Austin)

#### General:

- Data collector is meant to feed data into What if.
- Data collectors are made with public audience in mind.

#### Recreational & Incident Reporting Platform (RIRP):

- Data collection platform
- Provides two-way interaction between the park and the user
- Examines the basis of human-nature interaction.

#### Recommendations:

- Advertise data collector to get a dataset going for What if model
- Professionals in ecology, biology, and climatology should be consulted for major decisions with predicted change in ecosystem services
- Strava Metro provided potential partnership opportunities for access of robust recreational datasets for evaluating Discovery Park's recreational value
- Future human-nature interaction studies can use VMP's data collector as a framework for surveying.

# 1. Background and Project Objectives

## 1.1 Background

The largest city park in Seattle, Washington, Discovery Park offers the community a display of rich and diverse vegetation and wildlife, which has large areas of acreage preserved in its natural state. Recently, Discovery Park has made news for issues such as sewage spills, and proposed plans to build affordable housing within and near the park. These previously mentioned events has potential implications for a park that Earth Economics projects to have a financial value somewhere between \$557,000 and \$1.08 million dollars.

Given factors such as these, the Friends of Discovery Park (FoDP) reached out to the University of Washington's Master of Geographical Information Sciences (UW-MGIS) program to help them to develop a tool that can enable Discovery Park's stakeholders in making informed decisions about the recovery efforts and purposeful alterations to the park's natural landscape. Established in 1974, the FoDP is a non-profit organization powered by volunteer efforts "to defend the integrity of Discovery Park and to create and protect there an open space of quiet and tranquility, a sanctuary where the works of man are minimized." The collaborative efforts between FoDP and UW-MGIS began with a project executed by Rich, Peterson, and Marshall (2018). Their initial efforts laid the foundation of the work efforts for this project. Rich, Peterson, and Marshall's work was heavily focused on data collection and researching influences and the implications of landscaping alterations within Discovery Park. Our guided efforts for this project were focused on developing a decision support tool with stakeholders of FoDP using data resources created by Rich, Peterson, and Marshall. As with many projects in the GIS profession, we encountered challenges that were needed to be addressed to achieve our initial goal of developing a tool. The first hurdle was the input data used for the proposed geo-statistical modeling. One affordance in particular, recreational use of Discovery Park's land involved data considered proprietary, which had data use agreements constraining the utility of physical activity data for the proposed GIS application. Also, the data collected by last years MGIS cohort, though critical to the tool's functioning, does not appear sufficient to build a statistically accurate tool. The geo-statistical model would ideally have wildlife data points reflecting the diverse population of animals and insects within Discovery Park. Moreover, the ideal data source would have longitudinally robust data on wildlife and vegetation. Given these considerations, the authors of this report in collaboration with FoDP decided to make a necessary pivot after the sixth week of the academic quarter, to put FoDP in the best position to have a useful decision support tool as a long-term goal for this endeavour. The decision to make changes in the project's goals and objectives were reached after several failed attempts to build an effective analytic model accurate enough to make actionable decisions related to the preservation and improvements to Discovery Park.

## 1.2 Project Objectives

As mentioned in previous assignment submissions in GEOG 569, our initial primary objective was to develop a functional GIS web application for Seattle's Discovery Park so that FoDP can increase their odds of acquiring more financial resources for a decision support tool that The University of Washington will prototype. As such, the core mission of our GEOG 569 capstone project was to:

1. Propose Discovery Park Virtual Management Platform (VMP) as a framework for ecosystem-based green space management software across the country.
2. Implement the Virtual Management Platform (VMP) as an in-house educational interface for the general public on ecosystem services and benefits.

Given the project's pivot, our current objective now is to develop a dynamic and live data collection and analytic framework for FoDP. Though this pivot means that the VMP return on investment would be delayed, the long-term goals of having actionable data would increase multi-fold, as the data architecture could facilitate more accurate longitudinal and time-comparison analyses. The adjusted core mission of our GEOG 569 capstone project is to:

1. Develop and implement a wildlife, vegetation and recreational data collection application for Discovery Park.
2. Develop an automated geostatistical framework that will correlate FoDP's outcomes of interest using existing data sources and data collected from the developed data collection application.
3. Develop a web-focused wireframe that will synergize these web applications through a centralized browser interface.
4. Prepare a lay-language report that will inform FoDP how to replicate steps taken to develop tools so that they can enhance tool features and run statistical analyses once more data come available.

Though challenging to implement in three weeks, this approach was preferred over the original project scope at the beginning of the Summer 2019 quarter because: 1) current and accurate data appears to be the most vital component that will influence the success of the decision support tool, 2) the analytic needs of FoDP exceed the capacity of a single web application, and 3) clearly communicating the project's tasks in lay terms is critical in advancing the long-term success of FoDP's plans for this tool. Given the project's fourth objective, this report will follow a format unconventional to previous MGIS capstone reports. The following sections of this report will detail the steps needed to develop the proposed tool so that individuals with moderate ArcGIS knowledge can replicate and enhance the proposed tool. Due to this report's instructional format, typical GIS reporting sections such as "data development," "workflow implementation," are "results" will be interwoven within the instruction of each tool. The conclusions and recommendations section will be at the end of this report. The project's system resource requirements and business case evaluation sections are located in the project appendices.

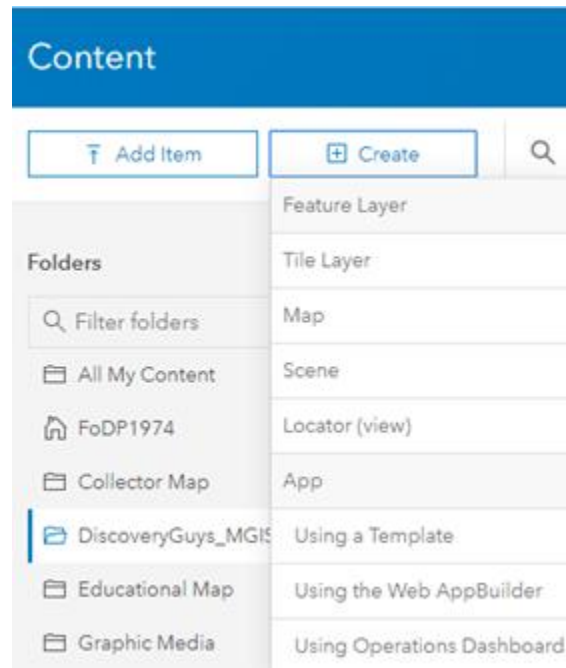
## 2. Data Collector Web Applications

### 2.1 Hosted Feature Layers

ArcGIS Online offers hosted layers to be published by accounts within an organization. FoDP's creator ArcGIS online account status satisfies the prerequisites for publishing hosted feature layers for other to query, visualize and edit. Hosted features functions like feature classes in ArcGIS Desktop software, these layers can house specific vector data types with a wide range of fields to supplement individual spatial elements. Users within the organization can be broken down to **Viewer**, **Data Editor**, **User**, **Publisher** and **Administrator** access to facilitate access and maintenance of the hosted feature layers. More details can be found on AGOL FAQ<sup>1</sup>.

#### 2.1.1 Setting Up Hosted Features for Data Collection

In order to provide hosted feature layers for other AGOL applications, simply navigate to the account's content on the AGOL platform and hit "**Create**". For the purposes of the VMP data collector platform, "**Feature Layer**" should be the primary hosted feature for FoDP (**Fig 1**).

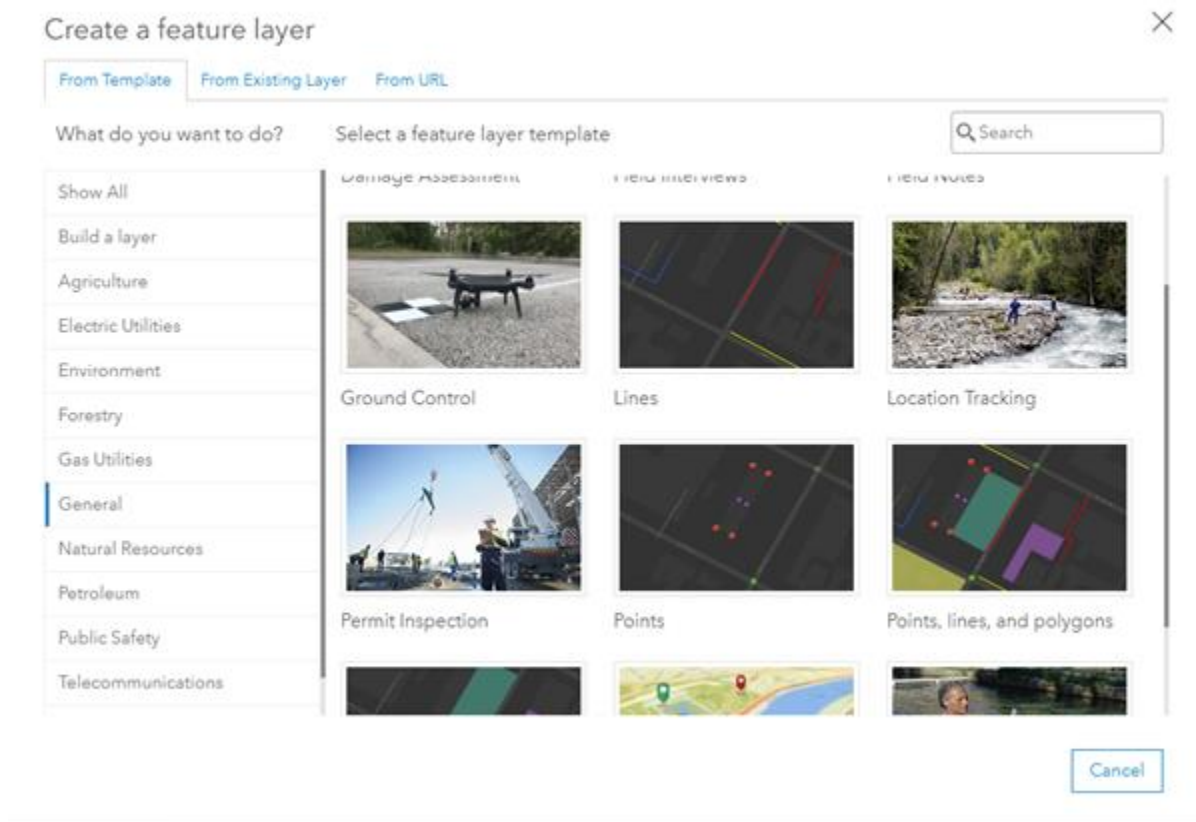


**Fig 1.** Creating hosted feature layer.

After "**Feature Layer**" is selected, the user will be prompted with options to create from default layer templates based on the user layer requirements. Depending on the specific real-life phenomenon FoDP wishes to capture, either **point**, **line** or **polygon** can be selected from the default template to provide the hosted feature layer with default fields

<sup>1</sup> [AGOL FAQ - User Privileges](#)

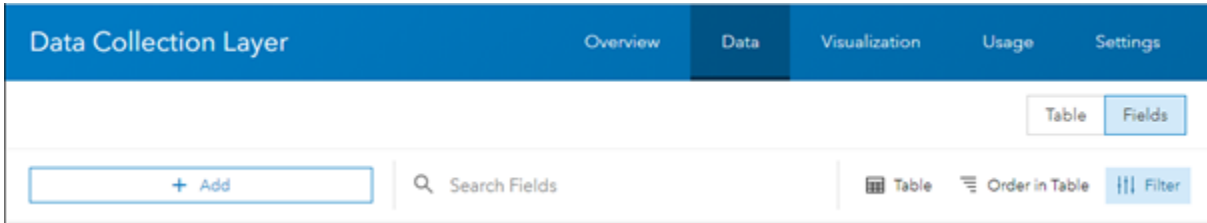
for the type of vector geometry (**Fig 2**). Proceed with the selected geometry type and a sequence of options will be presented for the user to fill out metadata for the feature layer. All the contents provided for the hosted feature layer's metadata can be updated from the layer details in FoDP's **"Content"** menu.



**Fig 2.** Default feature layer templates from ArcGIS Online.

Once the hosted feature layer is successfully created, an **"Overview"** will be shown up for the feature layer. The **"Data"** tab allows the user to create, modify and update existing elements and fields housed by the feature layer. To add additional fields for data collection, click on **"Add"** and fill out the **"Add Field"** form presented by the interface (Fig). The most important aspect of the field creation wizard is the **"Allow Null Values"** option. If the created field is meant to be a required field for the data collector, then the option should be left **"unchecked"** to disallow null values. Once a field is created, each field can be modified and updated by clicking on the **"Display Name"** for the field and hit the **"Edit"** option afterward (**Fig 3**).





**Fig X.** Sample hosted feature layer data menu.

### New Field

#### Description

Additional field for collecting data.

#### Field Value Type

Type or Category

#### Settings




Allows Null Values	Yes
Editable	No
Default Value	None
Length	128
Unique	No

**Fig 3.** Field overview from the Data tab under “Fields”.

A host of desired field domains (attribute domains) can be created for the added fields or existed fields by using the “**Create List**” option. The “**Label**” are user-defined values that can be selected by data collector, and the “**Code**” is the backend alias for the specific “**Label**” (**Fig 4**). Once completed, save and return the data view and you should be able to see a “**List of Values (Domain)**” listed for your newly created field. More details on customizing a hosted feature layer can be access from AGOL FAQ<sup>2</sup>.

<sup>2</sup> [AGOL FAQ - Publishing Hosted Feature Layers](#)

List of Values: New Field ✕

Label	Code
<input type="text" value="Value 1"/>	<input type="text" value="V_1"/> 
<input type="text" value="Value 2"/>	<input type="text" value="V_2"/> 
<input type="text" value="etc"/>	<input type="text" value="V_etc"/> 

+ Add

Create the list of values for this field by entering attribute values one at a time or by generating the list from the current attribute values in the layer.

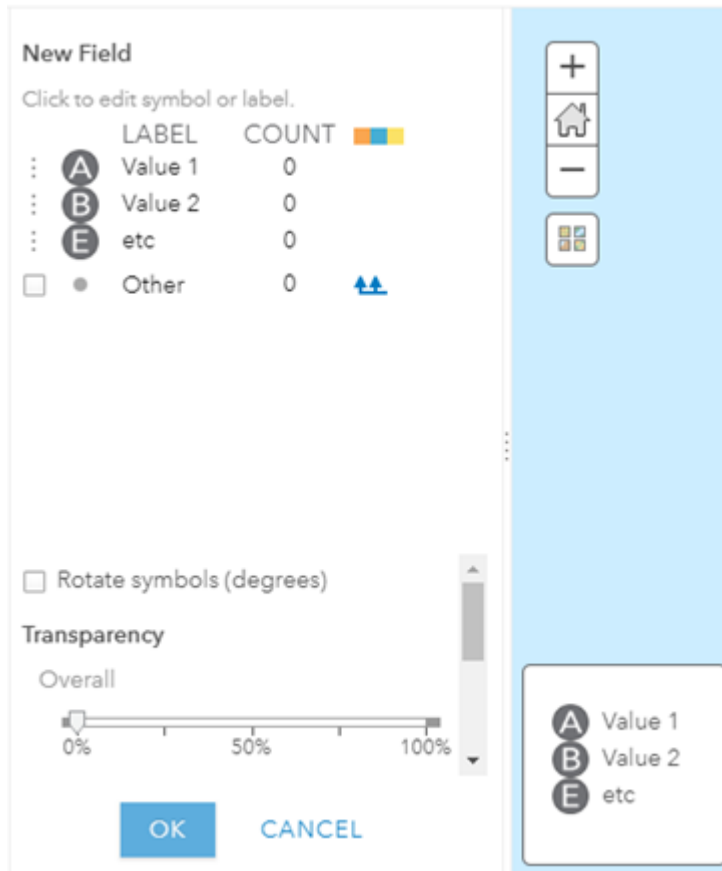
Add, edit, reorder, and delete items in the list. The Label is the displayed value and can be any text. The Code is the value stored in the database and must match the field type.

**Fig 4.** Sample attribute domains for a field in ArcGIS online hosted feature layer.

### 2.1.2 Visualization of Hosted Feature Layers

Hosted feature layers can contain default symbology for the layer to take on for visualizing on a web map. Within the feature layer, navigate to the “**Visualization**” tab and click on the “**Change Style**” icon (👤). Choose the field to default the data visualization, and then select “**OPTIONS**” to customize the desired symbology by clicking on the current symbology (**Fig 5**).

After the symbology has been customized for default visualizations, select “**Save Layer**” to register the visualization. By customizing visualization through the hosted feature layer “**Visualization**” menu all other AGOL applications and products that utilizes the feature layer will have universal visualization of the data layer.



**Fig 5..** Customizing symbology for data visualization of fields from feature layer.

### 2.1.3 Privacy Settings for Hosted Features

ArcGIS online hosted feature layer allows customized layer settings centered around data management and accessibility. For long-term usage, enabling “**Delete Protection**” is highly recommended to prevent data loss from misuse. Privacy settings can be access from the “**Settings**” tab under “**Feature Layer (hosted)**” section (**Fig 6**).

Feature Layer (hosted) Save Cancel

---

Editing

- Enable editing.
- Keep track of created and updated features.
- Keep track of who created and last updated features.
- Enable Sync (disconnected editing with synchronization).

- Who can edit features?  
Share the layer to specific groups of people, the organization or publicly via the Share button on the Overview tab. This layer is not shared.

- What kind of editing is allowed?
  - Add, update, and delete features
  - Add and update features
  - Add features
  - Update features
  - Update attributes only

- What features can editors see?
  - Editors can see all features
  - Editors can only see their own features (requires tracking)
  - Editors can't see any features, even those they add

- What features can editors edit?
  - Editors can edit all features
  - Editors can only edit their own features (requires tracking)

- What access do anonymous editors (not signed in) have?
  - The same as signed in editors
  - Only add new features, if allowed above (requires tracking)

- Who can manage edits?
  - You
  - Administrators
  - Data curators with the appropriate privileges

**Fig 6.** Hosted feature layer privacy setting for editing, viewing, and data management.

Depends on FoDP's comfort in providing access for the public to use the data collector to populate hosted feature layers, several important settings should be enabled for the intended audience. Suggested setting for hosted feature layer privacy can be found below.

### ***Public Audiences***

Hosted feature layers published to allow public usage often face difficulty in maintaining data quality and integrity. To assist in facilitating a productive process of data collection with the hosted layers below is a list of important settings to consider for public usage:

#### Editing

- Enable editing
- Keep track of created and updated features
- Keep track of who created and last updated features
- Enable Sync (disconnected editing with synchronization)

#### Editing Type (*only one*)

- Add features
- Add and update features

#### Editing Viewing Access (*only one*)

- Editors can see all features
- Editors can only see their own features (require tracking)

#### Editing Features (*only one*)

- Editors can only edit their own features (require tracking)

#### Anonymous Editors (*only one*)

- Only add new features, if allowed above (requires tracking)

### **Trusted Audience**

Users that are entrusted with collecting spatial elements for the feature layer should be allowed higher access to streamline data editing and in-field maintenance of data integrity. FoDP should consider modifying user privileges described in [Hosted Feature Layers](#) to manage trusted user access to hosted features. Within the feature layer privacy settings, a list of important settings to consider are listed below:

#### Editing

- Enable editing
- Keep track of created and updated features
- Keep track of who created and last updated features
- Enable Sync (disconnected editing with synchronization)

#### Editing Type (*only one*)

- Add and update features
- Add, update, and delete features

#### Editing Viewing Access (*only one*)

- Editors can see all features

#### Editing Features (*only one*)

- Editors can edit all features
- Editors can only edit their own features (require tracking)

#### Anonymous Editors (*only one*)

- Only add new features, if allowed above (requires tracking)

## 2.2 Visualizing Layers in Web Maps

ArcGIS Online Web Maps serve as the basis and foundation for building web applications. The online web map functions similarly to data frames in ArcGIS desktop counterparts. Despite having data analysis capabilities in the web map interface, it is recommended to only use ArcGIS online web maps for only data visualization and client-facing customizations. More information about the web map can be found on AGOL documentation<sup>3</sup>.

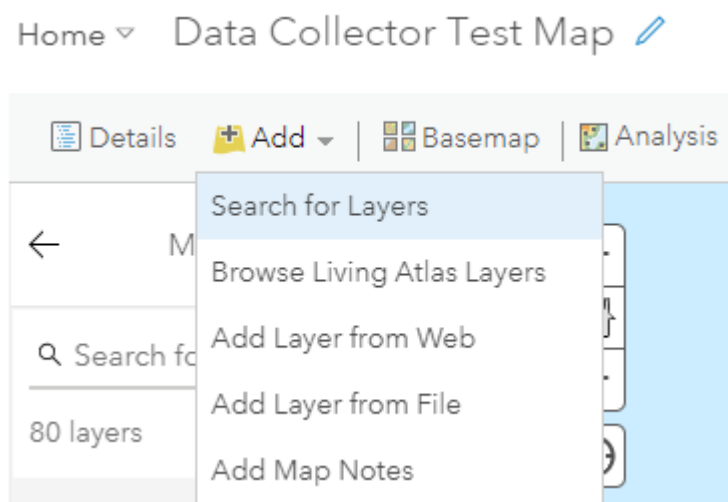
### 2.2.1 Setting Up Layers for Web maps

ArcGIS Online web map interface allows addition of feature layers from a wide range of sources including the user's content repository, public feature layers, local files, or other import options. To construct the web map for a data collector, hosted feature layers will be the main import for

---

<sup>3</sup> [AGOL - Web Map Documentation](#)

the web map. To add a layer into the web map, “**Search for Layers**” and locate the hosted feature layer created within FoDP’s content repository and “**Add to Map**” (Fig 7).



**Fig 7.** Adding layers into ArcGIS Online web map.

### 2.2.2 Adjusting Visualization

The hosted feature layer should have defaulted symbology if layer creation follows *section X.1.2 Visualization of Hosted Feature Layers*. If the layer symbology of the added hosted feature layer requires further customization, access data symbology with the “**Change Style**” icon (🎨) and follow the instruction to specified desired field to symbology and applied the desired symbology. More details about available options within the “**Change Style**” property can be found on the web map documentation on style customization<sup>4</sup>.

## 2.3 Using ESRI WebApplication for Data Collection

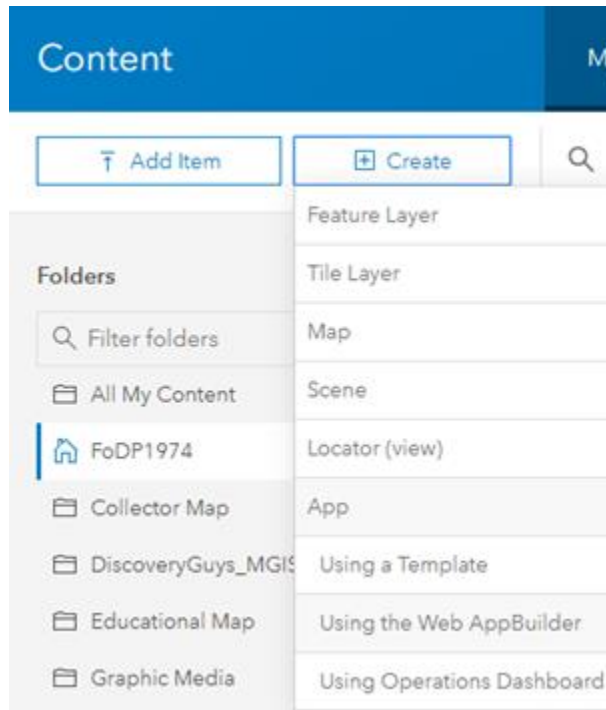
ESRI Web Appbuilder provides an online builder platform for developers to create web applications that are accessible across stationary and mobile platforms. These applications can perform a wide range of functions ranging from information visualization, management, collection, and other customizable functionality. The web application developed by University of Washington MGIS 2019 FoDP group will center around utilizing the existing ESRI builder framework to construct an open-source data collector for FoDP.

### 2.3.1 Building ESRI Web Application

Similar to constructing hosted feature layers and web maps, within the “**Content**” page of the ArcGIS Online platform, users can create a web application from the “**Create**” menu (Fig 8). Follow the wizard prompt after selecting “**Using the Web AppBuilder**” to provide the web

<sup>4</sup> [AGOL - Change Style Documentation](#)

application with metadata, these metadata can be updated on the web application details page from accessing the application from the content list.



**Fig 8.** Creating ESRI Web Application from Content page.

Once the web application is successful created, the user will be brought to the **Web Appbuilder** interface (**Fig 9**). The web application can be customized from the AppBuilder via four distinct aspects – **Theme, Map, Widget, Attribute**.

**Theme** – control of the overall application theme and color palette.

**Map** – settings for referenced web map and associated customization of map viewing.

**Widget** – controls for adding, editing, deleting, and customizing individual functions within the web applications. For instance, the **Edit** widget allows users to add, update, and manage data from editable layers. The **Edit** widget can be customized to restrict fields from user access.

**Attribute** – settings for the overall web application in regards to branding, backend data source and content access.

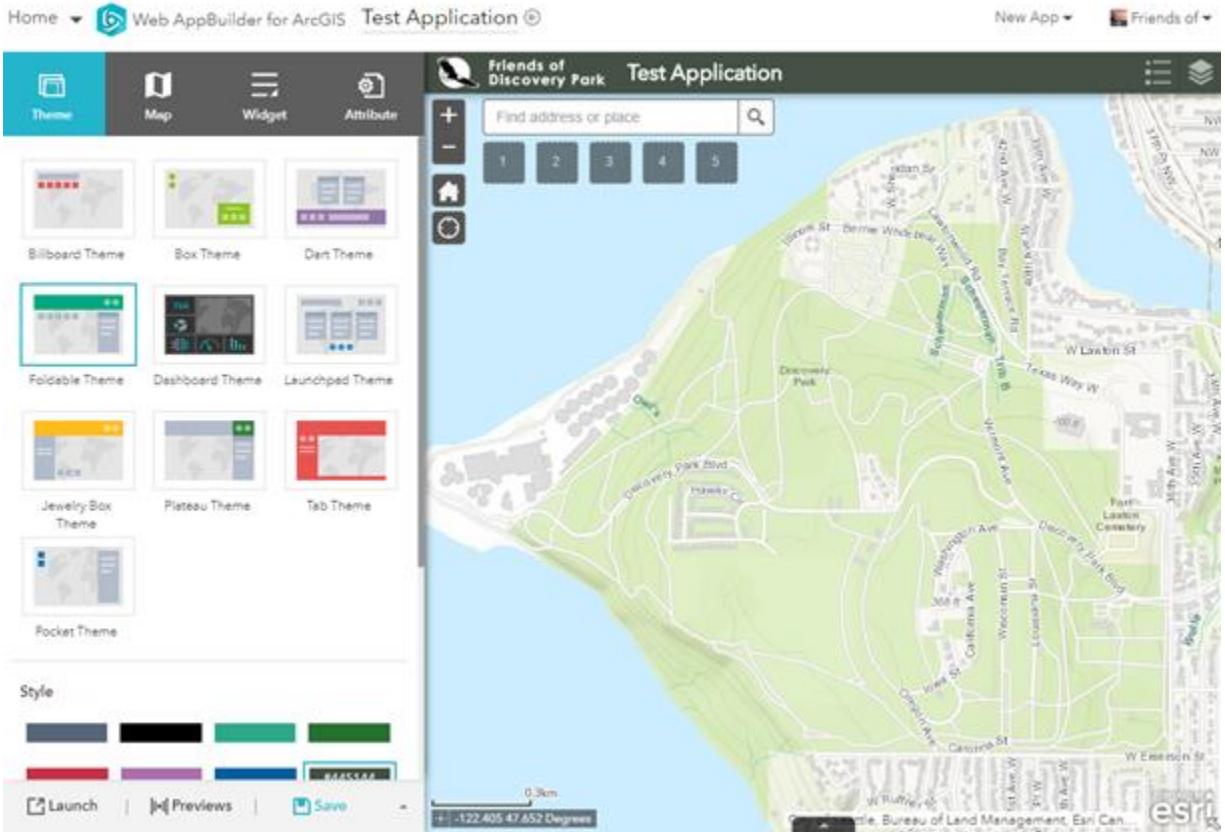
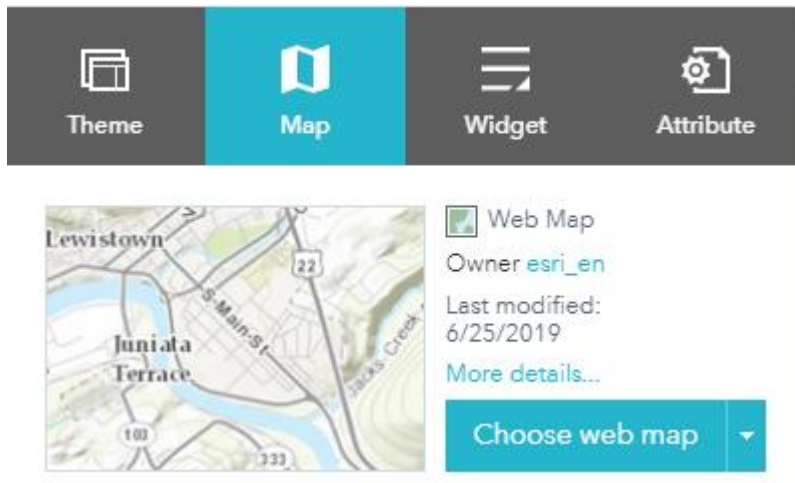


Fig 9. Default ESRI Web AppBuilder interface.

The first order of business in making a functional web application is to reference an existing web map from the user content repository. Within the “Map” tab select “Choose web map” to select the web map set up from [Visualizing Layers in Web Maps](#) of this document (Fig 10).





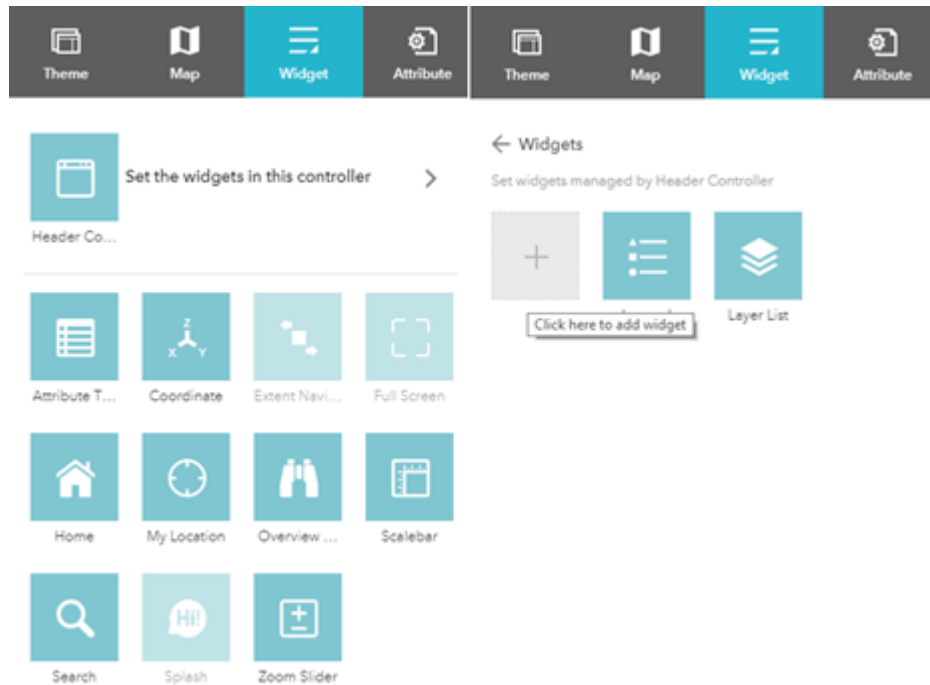
**Fig 10.** Selecting map for web application.

### 2.3.2 User-experience Widgets

ESRI web applications come with a suite of default widgets that enhances basic user functionality. These widgets can be customized and adjusted based on the needs of the application from the “**Widget**” menu in the AppBuilder (**Fig 11**). For the purpose of the data collector, legends, layer list are the two widgets that requires further customization from the default setting. The “**Legend**” widget governs the visibility of the symbology for each layer within the web application, while the “**Layer List**” controls the visibility and accessibility of each individual layer and the attribute table. Widget customization is application wide and will take effect once the changes are saved within the AppBuilder interface.

Several widgets that can be further customized to enhance user experience includes:

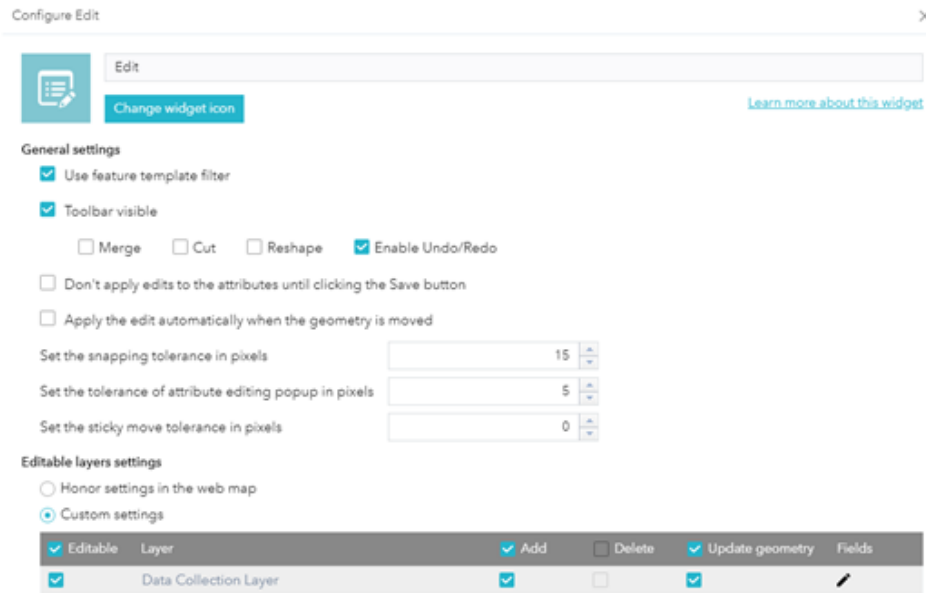
- Coordinate** – controls default projection of the web application.
- My Location** – adjust settings for using of GPS location systems by user.
- Overview** – adjust placement of in-set map for locating current extend on world extend.
- Attribute Table** – customizable viewing and access of layer attribute tables.
- Scalebar** – adjust visualization of map scale bar and units.



**Fig 11.** Default widget view for web application created from FoDP template.

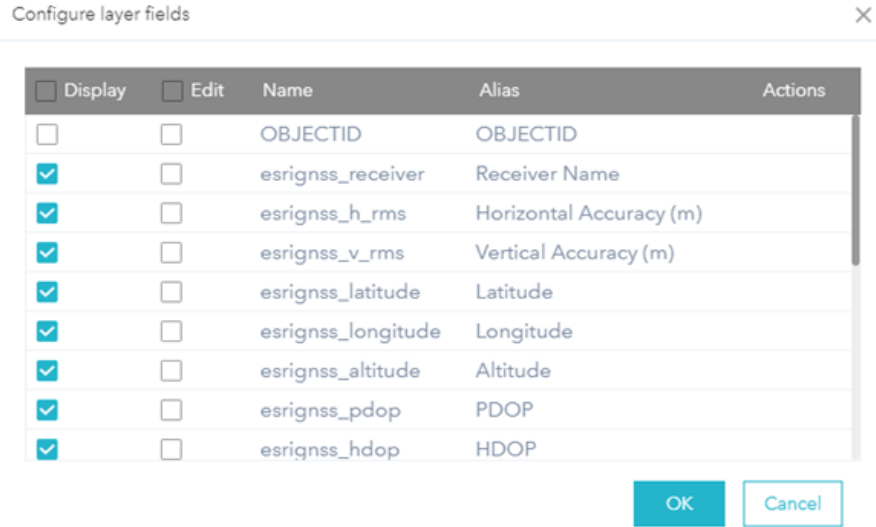
### 2.3.3 Data Collection Widgets

ESRI Web Appbuilder offers two options for on-the-fly data editing – 1) **Edit** Widget and 2) **Smart Editor**. For the purpose of the FoDP data collection framework, the edit widget was focused on for this project and primarily used in the construction of the data collector application. The “**Edit**” widget offers a whole suite of options to customize user-side viewing and interaction with the hosted feature layer fields (**Fig 12**).



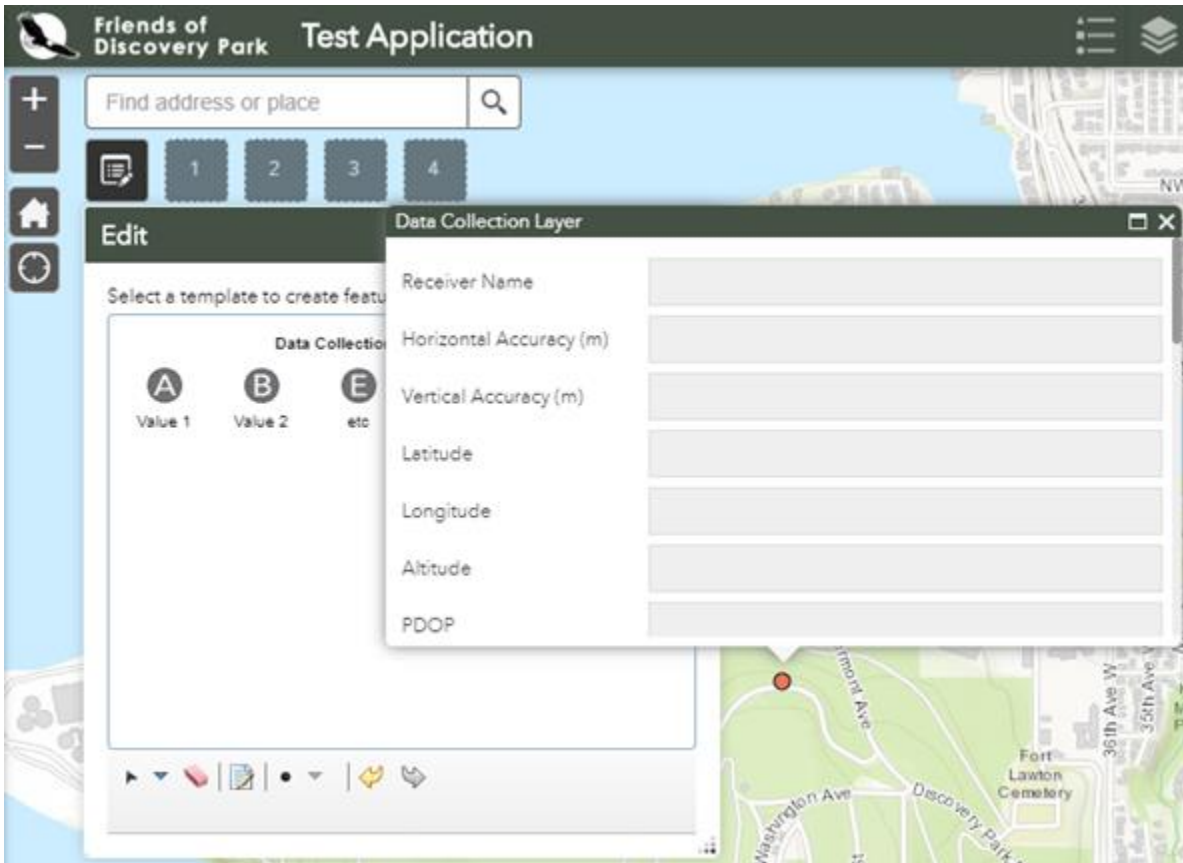
**Fig 12.** Edit widget customization menu.

Each editable field can be toggled for “**Display**” and “**Edit**” on the user side and these customized field settings allows the AGOL host to manage user access to recorded elements (**Fig 13**).



**Fig 13.** Hosted layer field access customization within Edit widget.

These settings are reflected on the user-end when elements are added to the hosted feature layer (**Fig 14**). The same settings are applied whenever the user decides to **add**, **update**, or **delete** existing features. End users do not have access to manipulate which fields are available or unavailable; however, as mentioned in [Setting Up Hosted Features for Data Collection](#), fields that do not accept “**null values**” will become required fields of the hosted layer and users will be required to provide a value for that field in order for an element to be added.



**Fig 14.** Edit widget user-end view.

The symbology of the available templates provided by the hosted layer is customizable within the “Data” tab of the hosted layer as discussed in [Visualization of Hosted Feature Layers](#). More information on the “**Edit**” widget can be found on ESRI AGOL documentation<sup>5</sup>.

### 2.3.4 Complementary Summary Widgets

Akin to the “**Edit**” widget’s option to customize for end user viewing and access. ESRI Web AppBuilder offers numerous widgets that caters to enriching user experience. Several default widgets provided by the AppBuilder serve as helpful complements for the data collector experience. Through the development of data collector, a list of four widgets were found to enhance user experience and provide information for FoDP’s data collectors.

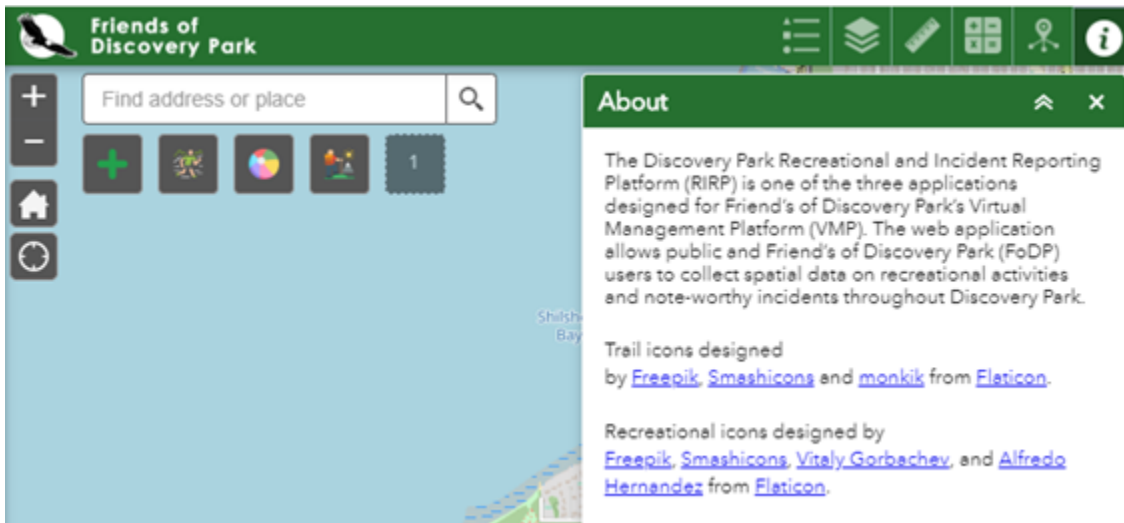
List of Complementary Widgets for Data Collector Application:

#### 1) **About**<sup>6</sup>

<sup>5</sup> [ESRI WebApp Builder - Edit widget](#)

<sup>6</sup> [ESRI Web AppBuilder - About widget](#)

The “**About**” widget is an all-purpose widget for providing a pop-up window within the web application for user to read any additional information, description, summary of the application. This pop-up window can also be used to create custom HTML contents for the user to view and interact (**Fig 15**).

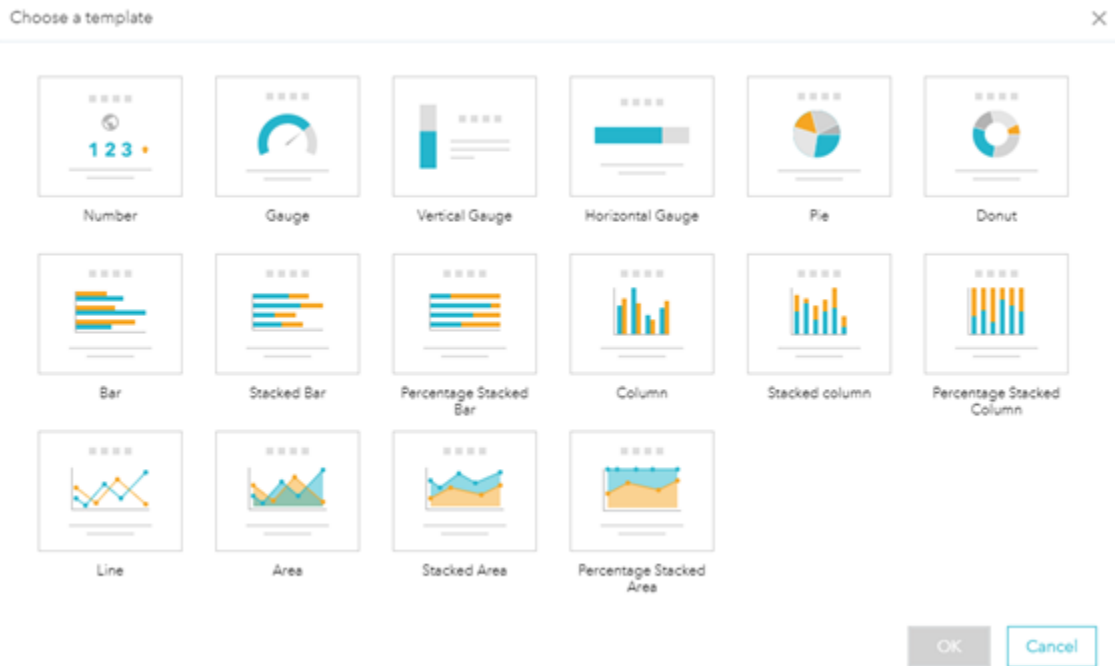


**Fig 15.** About widget from FoDP Recreational & Incident Reporting Platform.

## 2) Infographic<sup>7</sup>

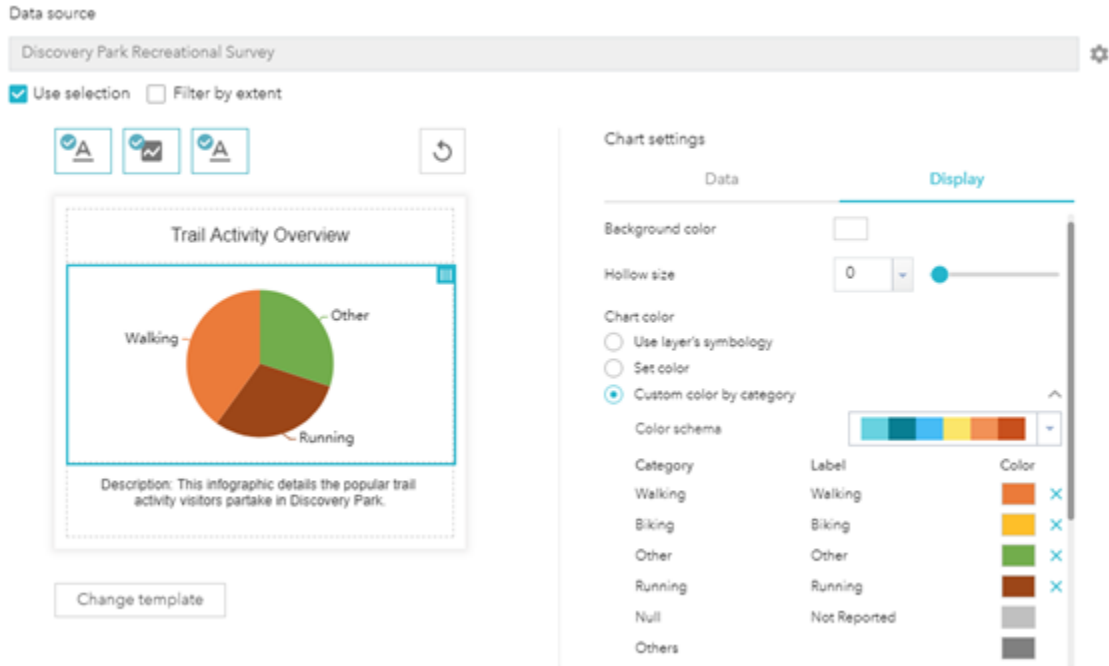
The “Infographic” widget provides templates for developer to use in illustrating field summaries or charted statistical data from the map (**Fig 16**).

<sup>7</sup> [ESRI Web AppBuilder - Infographic widget](#)



**Fig 16.** Options of available infographic templates.

The widget customization offers a limited selection of options for the user to customize their chart (**Fig 17**).



**Fig 17.** Widget customization for fine-tuning infographic display.

The result is a user-friendly display of map information that facilitate highlighting of individual field categories and features (**Fig 18**). This widget provides user with feedback on their contributions to the hosted feature layer and reinforce positive interaction with FoDP's data collector application.



**Fig 18.** Infographic from FoDP Recreational & Incident Reporting Platform. Map elements from a chart category is highlighted when user hovers over the infographic.

### 3) Summary<sup>8</sup>

The “**Summary**” widget allows web application to showcase a summary of numeric attributes from a feature layer. The summarized features are provided based on user viewing extent, and individual category for the summarized features can be filtered to isolated specific elements for viewing. To set up a “**Summary**” widget, four calculations are available for field summary – 1) sum (**SUM**), 2) average (**AVG**), 3) maximum (**MAX**) and 4) minimum (**MIN**). These calculations can be performed and displayed for multiple fields within a single feature layer (**Fig 19**).

<sup>8</sup> [ESRI Web AppBuilder - Summary widget](#)



Select Summary Layer: Discovery Park Recreation Survey 1

Summary Layer Filter Field: Activity

[Add Summary Field](#)

Type	Field	Label	Actions
SUM	Length	Total Length (m)	
AVG	Length	Average Length (m)	

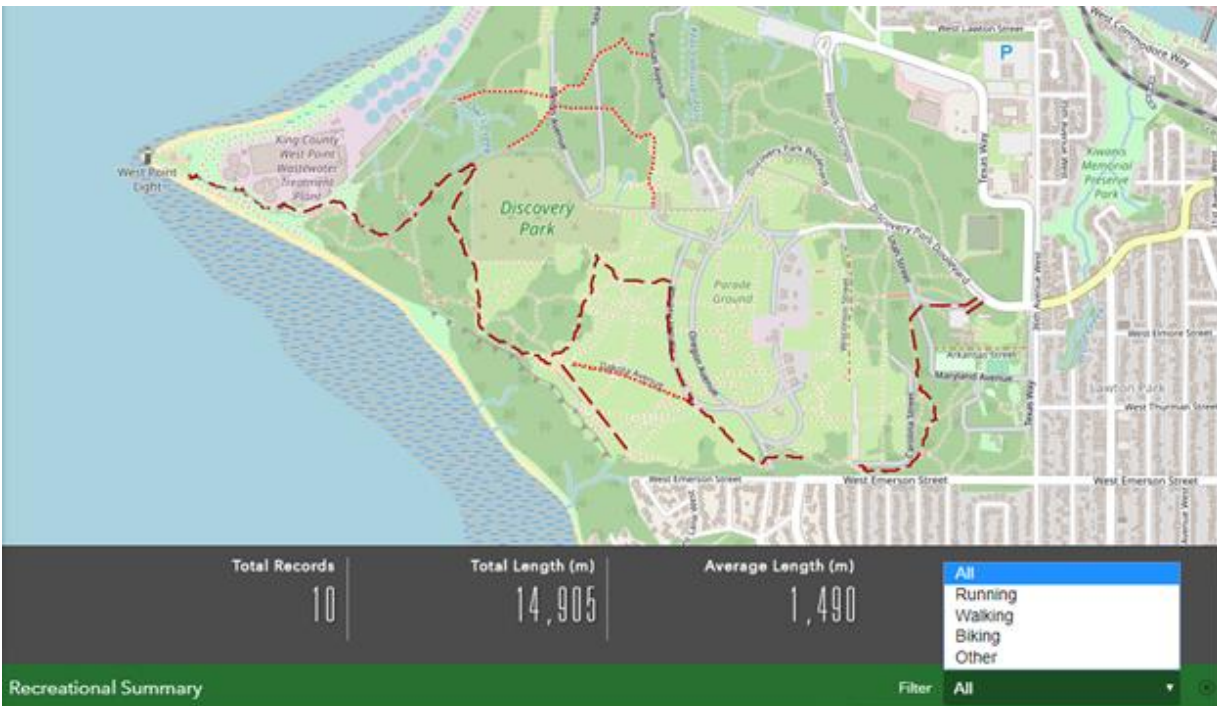
Feature Count Label    Feature Count Label: Total Records

Display as Summary Clusters

Refresh Interval (Minutes):

**Fig 19.** Summary widget set up for recreational activity survey 1.

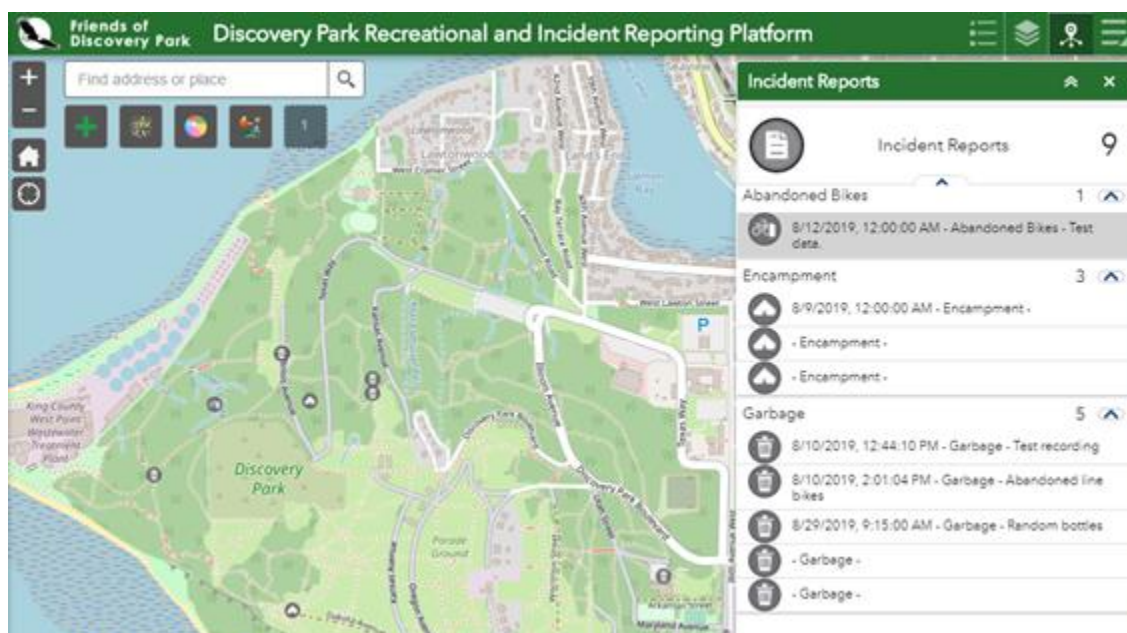
This widget functions similarly to the “**Infographic**” widget but provides a more concise view of numeric values for spatial elements (i.e. shape length) as seen in figure 20. Continuous value types are also well suited for this widget (i.e. elevation).



**Fig 20.** Trail activity summary from user contribution, individual activity can be filtered and summarized.

#### 4) Info Summary<sup>9</sup>

The “**Info Summary**” widget provides the user with a collection of features from the map extent for each tallied feature layer. The collected elements can be organized based on preset templates from the hosted feature layer and individual records are displayed when the collection is expanded for viewing (**Fig 21**). This widget is exceptional for monitoring incidents in Discovery Park reported by data collectors. Individual elements within the collection can be interacted directly and associated feature information are displayed. Another advantage of the “**Info Summary**” widget is the widgets ability to summarize multiple layers at the same time. This reduces cluttering of widgets on the web application without losing the summary function.



**Fig 21.** Info Summary from FoDP Recreational & Incident Reporting Platform.

#### 4. Recreational and Incident Reporting Platform

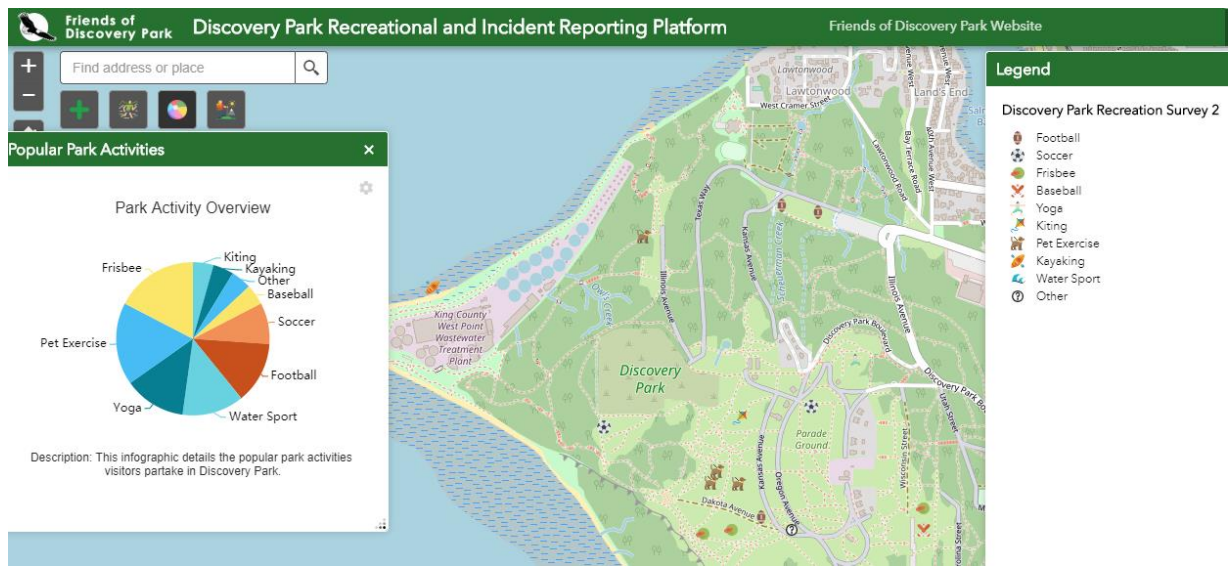
A primary part of Virtual Management Platform project’s objectives is to facilitate interaction and education of visitors at Discovery Park. The platform aims to provide users with an understanding of effects from potential changes to the park’s natural elements. Along with the educational mission, FoDP also seeks to portrait human-nature interaction through the lens’ of the park visitors. The Discovery Park Recreational and Incident Reporting Platform (RIRP) was

<sup>9</sup> [ESRI Web AppBuilder - Info Summary widget](#)

constructed as a means of establishing a foundation for surveying human-nature interactions while informing the public about the park's suggested recreational activities and safety precautions. The platform facilitates a two-way interaction where public users can review personal data contributions and FoDP users can manage park usage and respond to reported incidents.

#### 4.1 Recreational Activities

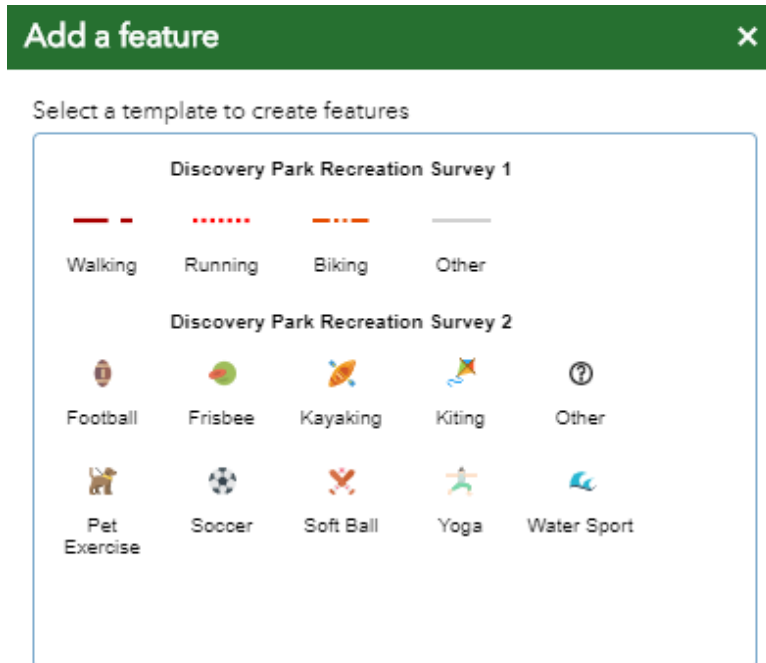
Discovery Park provides the natural environment and service needed for conducting recreational activities. Physical recreational activities such as hiking, running, sports, and other outdoor hobbies requires the necessary environment to perform. These physical recreational activities are also often associated with mental and physical health benefits. The RIRP data collector allows users to input recreational activity information onto the platform and provide a spatial display of the user's activity pattern (**Fig 22**).



**Fig 22.** User recreational activity overview provided by Discovery Park Recreational and Incident Reporting Platform.

##### 4.1.1 Adding Recreational Activities

The RIRP allows user to simply utilize the “**Edit**” widget to add in feature templates to denote the user's recreational activity (**Fig 23**).



**Fig 23.** Available templates for recreational activities with Discovery Park Recreational and Incident Reporting Platform.

A simple form will be provided when user finish drawing a feature from the available templates. Required fields are denoted by \* and any other fields within the form will be treated as optional (**Fig 24**). Closing the form will save the feature and update the hosted feature layer with the user edit.

The image shows a web-based form titled "Discovery Park Recreation Survey 2" overlaid on a map of Discovery Park. The form contains the following fields and options:

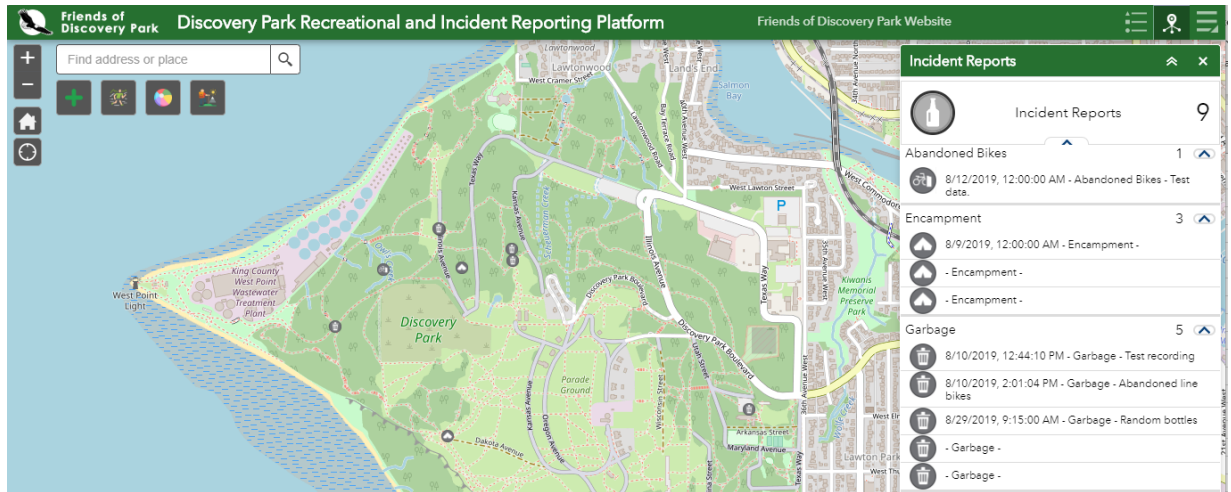
- Visit Date \***: A dropdown menu showing "8/15/2019".
- Activity \***: A dropdown menu showing "Kiting".
- Other (Please Specify)**: An empty text input field.
- Attachments:** A section with a horizontal line, the text "None", and an "Add:" label followed by a "Choose File" button and the text "No file chosen".

The background map shows various park features, including "Parade Ground", "West Emerson Street", "Perkins Lane West", and "West McLaren Street". A red dashed line on the map indicates a specific path or boundary.

**Fig 24.** Edit form for recreational activities with Discovery Park Recreational and Incident Reporting Platform.

## 4.2 Incident Reporting

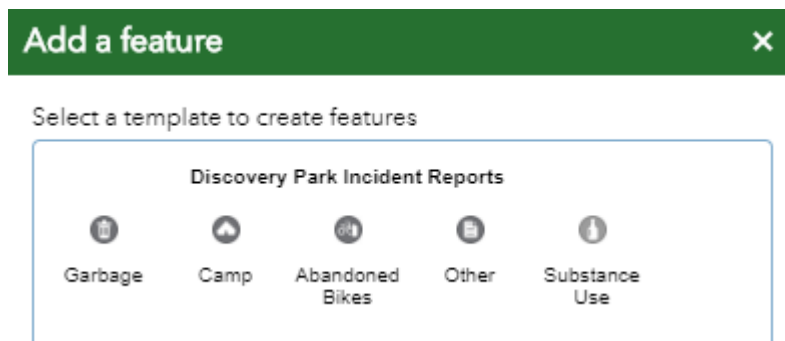
Another aspect of human-nature interaction focused by FoDP pertains to public safety within Discovery Park. Social activities are not limited to recreation, public safety concerns can oftentimes be attributed as another form of human-nature interaction. The RIRP allows users to add, update, and view identified public incidents within the park area. These incidents are live-updated and keeps users informed of areas to avoid and exercise caution (**Fig 25**).



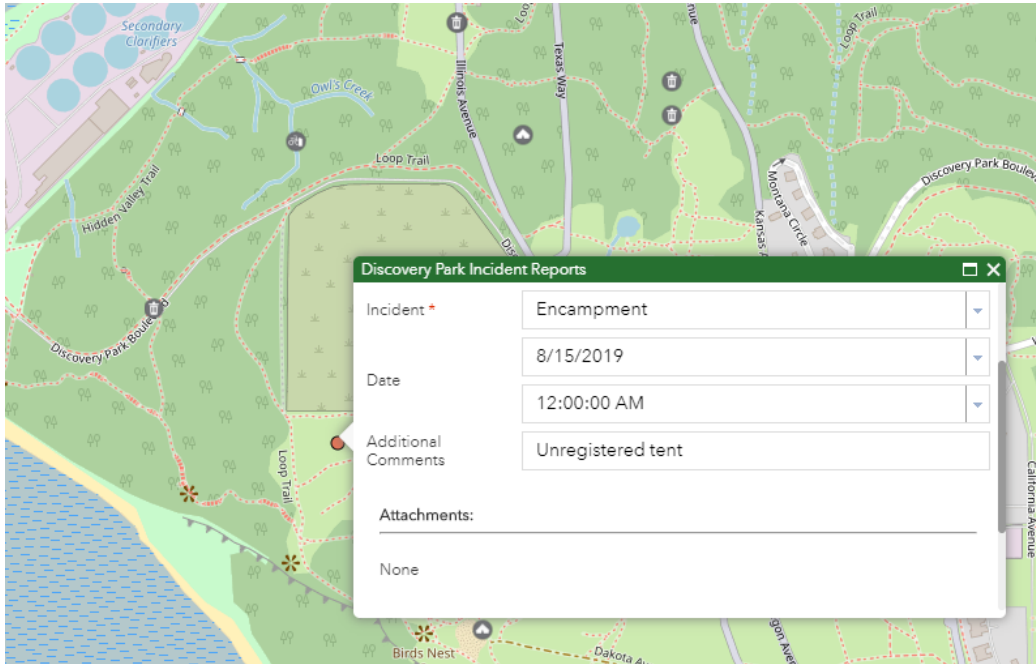
**Fig 25.** User incident report overview provided by Discovery Park Recreational and Incident Reporting Platform.

#### 4.2.1 Adding Incidents

Similar to recreational activities, public incidents can be reported on the RIRP using pre-coded templates (**Fig 26**). Once a template is added, additional details can be filled out by the form provided for the feature (**Fig 27**). Each feature allows the user to update on the fly and provide changes to reported incidents.



**Fig 26.** Available templates for incident reporting with Discovery Park Recreational and Incident Reporting Platform.



**Fig 27.** Edit form for incident reporting options with Discovery Park Recreational and Incident Reporting Platform.

## 5. “What-If” Analysis Application

The “What-If” Analysis Application is comprised of two major deliverables. The first deliverable being a statistical framework for FoDP, which can be used once more data becomes available for reasonably accurate analyses<sup>10</sup>. The second deliverable is a more simplified web application assessing implications in land cover modifications with respect to the affordance outcomes of interest (i.e., aesthetics, air flow quality, carbon sequestration, disaster mitigation, health, shelter, waste and water). The second deliverable with respect to the “What-If” analysis involves Python source code<sup>11</sup> and an ESRI Web AppBuilder Application<sup>12</sup>.

### 5.1 Data Management and Input Data Development

At the request of FoDP we built an analytic framework using ArcGIS the Python API and *arcpy* libraries. The intent is for this source code to serve as a foundational workflow than can automate analyses once more comprehensive data becomes available. Relative to the code

<sup>10</sup> GitHub location for Jupyter Notebook of the geostatistical model: <https://shorturl.at/hwFLS>

<sup>11</sup> GitHub location for Jupyter Notebook of the simple application source code model: <https://shorturl.at/bioJ3>

<sup>12</sup> Location for simple “What-If” application: <https://fodp.maps.arcgis.com/apps/webappbuilder/index.html?id=cc7f57f3f4c04736b40a578b34c12835>

used in the simple model, it is imperative that users of this analytic workflow is aware of all the Python libraries use to conduct analyses. The core libraries used for the geostatistical model are: [arcpy](#), [ArcGIS Python API](#), [pandas](#), [numpy](#), and [scikit-learn](#). There were other libraries used, but the six previous libraries are critical to the functionality of the provided code. To this extent, the arcpy and ArcGIS Python API libraries are most functional for analysts with access to an active ArcGIS Desktop and ArcGIS online (AGOL) license. Therefore, it is recommended that coded models are processed using Integrated Development Environments (IDE) installed with the ArcGIS Desktop download. Among the IDEs preinstalled, we used the Jupyter Notebook IDE for this model. Also, these methods were implemented on a Microsoft Windows operating system. We accessed Jupyter Notebook through the Python Command Prompt within the ArcGIS program folder (as displayed in **figure 28**). Using this interface allows us to change the directory of our program on the front end by using the 'cd' command proceeded with the folder path where we plan to store the code (e.g., C://user/name/documents/FoDP). After one changes their directory, the user needs to type 'Jupyter Notebook' to start the software. If one is using the ArcGIS installed Jupyter Notebook IDE, most libraries used come preinstalled. scikit-Learn may need to be [pip installed](#) by opening the ArcGIS Python Command Prompt **as an administrator**, and typing 'pip install -U scikit-learn ' prior to changing the directory.

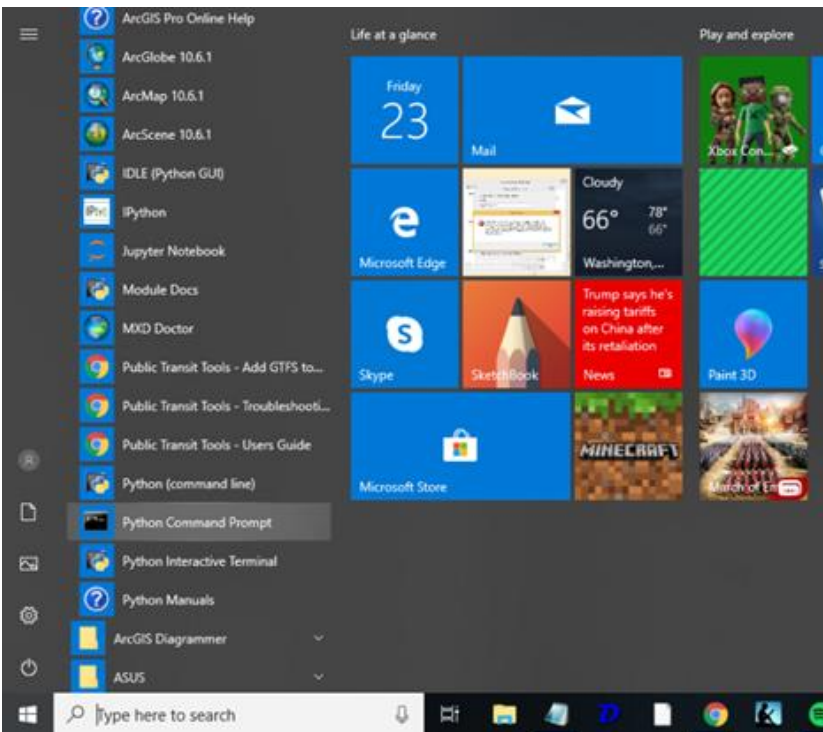


Fig 28. How to access Python Command Prompt from Microsoft Windows Start Menu.



It is not the intent of this report to teach readers how to use Jupyter Notebook, but the directions are made available on the Internet<sup>13</sup>. The libraries used for the geostatistical model can be imported using the following code:

```
import arcpy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matplotlib
import sklearn as sklearn
from matplotlib.ticker import FuncFormatter
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
import os
import getpass
from arcgis.gis import GIS
import sys
from arcgis.mapping import WebMap
from arcgis.features import SpatialDataFrame
```

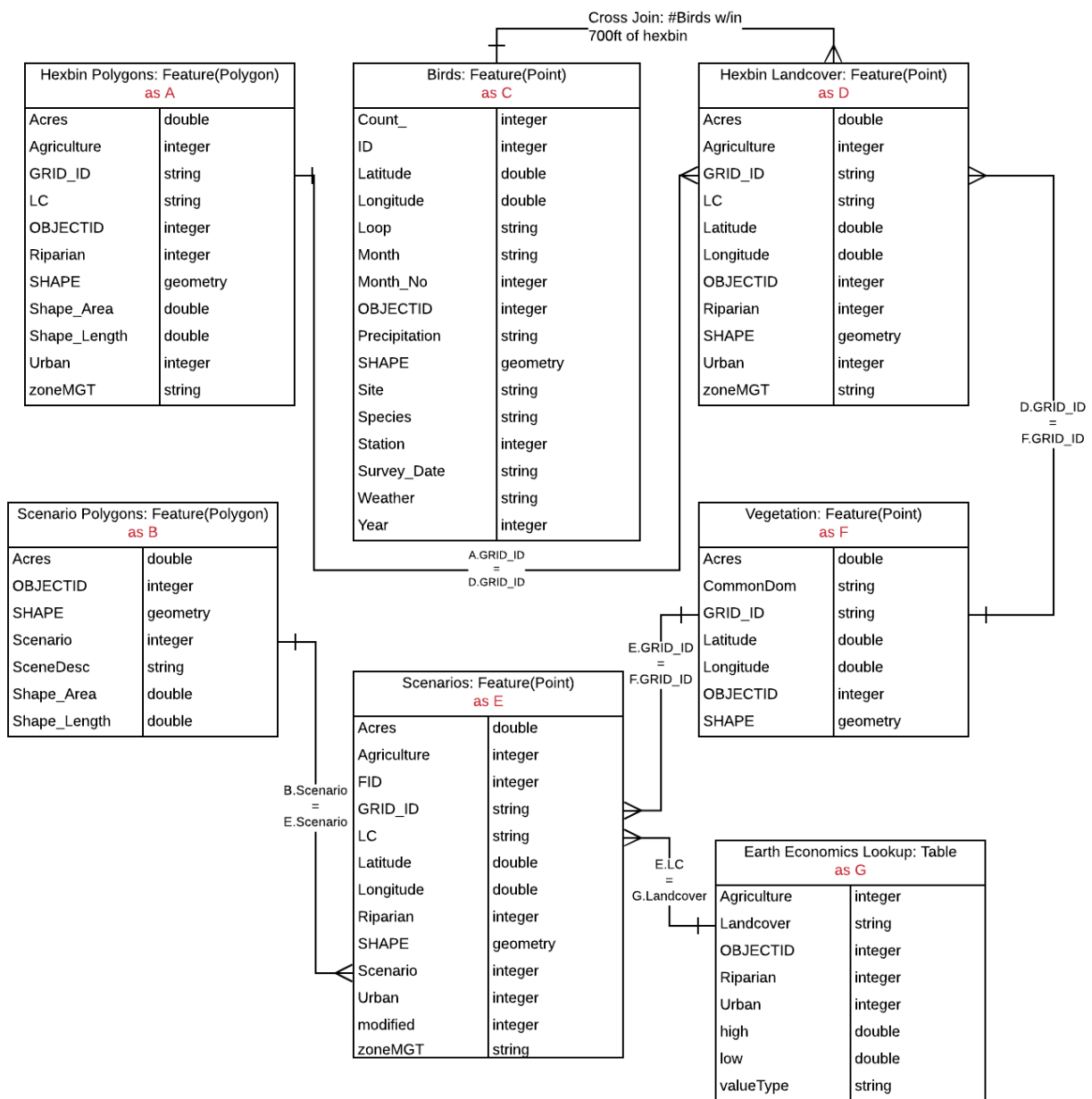
Next, we use the following code snippet to log into the clients AGOL portal. For security purposes, we use the Jupyter Notebook pre-installed *getpass* library to enter our stakeholder's username and password so that external users will not see the secure information. It is important to store the username and password as variables for increased security.

```
u = getpass.getpass(prompt='Enter username')
p = getpass.getpass(prompt='Enter password')
gis = GIS('https://fodp.maps.arcgis.com/', u, p)
```

An entity relationship diagram of the primary data sources used for statistical modeling and the “what-if” application is displayed in **Figure 29**. These feature classes and tables are located in the FoDP contents section of its AGOL account.<sup>14</sup> The full dictionaries for the data sources used are located in **Appendix 4**. All feature classes are projected using the following projection: *NAD 1983 StatePlane Washington North FIPS 4601 Feet*.

<sup>13</sup> <https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

<sup>14</sup> <https://fodp.maps.arcgis.com/home/item.html?id=945ae605ae594fa38bdb84e4d8e8a55d>  
and <https://fodp.maps.arcgis.com/home/item.html?id=a7e8c38fa59743cf8d3241abbbbeb372a>  
and <https://fodp.maps.arcgis.com/home/item.html?id=bf90bf466a75417c942802e55c1276a0>



**Fig 29.** Entity-relationship diagram of data sources used in “what-if” analyses

Given that the source code underlying the What-If application analyses are provided, this written report will not cover every detail of code used in the analyses. To promote transparency and reproducibility for modeling in the future, this report will cover important concepts so that GIS analysts will understand the rationale behind the coding design. A key component in the data massaging in this project is the importation of data. To do this we used the following code in the ArcGIS Python API:

```

#This code searches for feature Layer services in AGOL
search_result = gis.content.search(query="Input Data Used for Geospatial
Analysis v2")
Search_result #This line enables us to see the results in Notebook

#This code assumes what we are looking for is in the first feature Layer
#object from the code above
dataSource = search_result[0].layers
for lyr in dataSource:
    print(lyr.properties.name)

#After confirming the assigned title of layer from the code above, we
#decide to import the attribute table in the feature layer collection's
#first layer as a pandas dataframe
birds = dataSource[0].query().sdf
birds.head() #This line enables us to preview dataframe in Notebook

```

The code used to import tables from AGOL is slightly different when compared to importing feature classes. The primary difference from the above code is in the second block of code, which looks like this when importing a table:

```

#This code searches for data in AGOL
search_result = gis.content.search(query="Earth Economics Lookup Table")
search_result

#This code assumes what we are looking for is in the first feature Layer
#object from the code above
dataSource = search_result[0].tables #<<<This is what's different>>>
Look = dataSource[0].query().sdf
Look.head()

```

### 5.2.2 The geostatistical model

Early in this project, the FoDP stakeholder shared a keen interest in understanding the influence that vegetation has on wildlife. At the time of the project's initiation, a dataset representing bird sightings in years 2015 and 2016 appeared to be the most robust data source representing wildlife. There was also interest in understanding the influence that wildlife has on vegetation. Early in the project, the MGIS students came to the conclusion that understanding the influence that wildlife has on vegetation would require more historical data and data reflecting a more diverse sampling of wildlife. Though we caution FoDP to accept this model's findings as representative, we do believe that there are higher odds in getting data predicting wildlife presence given vegetation in the nearer future.

The key dependent variable in our geostatistical model is bird counts, which is present in our *birds* feature class. As suggested by the first phase of the virtual management platform project, we decided to use the 49,000 hexbin point feature class in our modeling so that we can leverage the increased power with large sampled data. That being said, the bird sightings are sampled from snapshot instances, which may not accurately reflect all areas within Discovery Park that serve as birds' habitat. To better reflect vegetative areas where various species of birds within Discovery Park likely spend time at, we aggregated the count of birds within 700 feet of each indexed hexbin. We considered using a buffer analysis where we looped through indexed hexbins and counted varying bird species within 700 feet of the centroid of each hexbin. Unfortunately, though simpler to code this approach was very computationally intensive on our computers. Therefore, we decided to do a cross-join between between our birds feature class and hexbin landcover point feature class using the pandas library. Therefore, to accomplish this task we first aggregated each species' bird count over the 1.5 year period. Next, we pivoted the bird species column so that each column counted the species for each bird. Afterwards, we crossed joined the pivoted bird dataframe with the hexbin landcover dataframe, which allowed us to aggregate the number of birds within 700 feet of each hexbin without having to use any computationally intense geoprocessing tools. Finally, we left joined our vegetation dataframe to the cross-joined bird and landcover dataframe. In this model, the vegetation columns (1= yes this vegetation species is dominant in hexbin; and 0 = this vegetation species does not have a major presence in the hexbin) are our independent variables. The final dataframe used to process the statistical analyses are indexed at the hexbin-level, which means that this analysis comprises of roughly 49,000 observations. A summary of the source code to accomplish the previously described data management tasks are below:

```
#pandas will not allow cross joins on columns
#from different dataframes with same name
key2 = key[['GRID_ID', 'Latitude', 'Longitude']]
birds2 = birds[['Species', 'Count_', 'Latitude', 'Longitude']]
birds2['Latitude2']=birds2['Latitude']
birds2['Longitude2']=birds2['Longitude']
del birds2['Latitude']
del birds2['Longitude']

#Lets do a cross join between the birds and hexbins,
#but only keep the rows where bird sightings
#are within 700 feet of a hexbin
test=key2.assign(foo=1).merge(birds2.assign(foo=1)).drop('foo', 1)
test['dist'] = np.sqrt(((test['Latitude']-
test['Latitude2']).pow(2))+((test['Longitude']-test['Longitude2']).pow(2)))
test = test[test['dist']<=700]
test.reset_index(inplace=True)
del test['index']
```

```

#Sum bird counts within each species at each GRID_ID
birdSum = test.groupby(['GRID_ID', 'Species'])['Count_'].sum().reset_index()

#now lets pivot the bird Species into separate columns
birdPivot =
pd.pivot_table(birdSum, index=['GRID_ID'], columns='Species', values='Count_')
birdPivot.reset_index(inplace=True)

#lets fill the nulls with zeros
birdPivot.fillna(0, inplace=True)
birdPivot.head()

#Now lets pivot the vegetation across columns
hexbins_pt_veg['val']=1
veggiePivot =
pd.pivot_table(hexbins_pt_veg, index=['GRID_ID'], columns='CommonDom', values=
'val')
veggiePivot.fillna(0, inplace=True)

#now let's left join the birds pivot on our key table
almost = pd.merge(key[['GRID_ID', 'LC', 'Latitude', 'Longitude']], birdPivot,
how='left',
                left_on='GRID_ID', right_on='GRID_ID')

#now left left join our vegetation to finalize our dataframe that we will
model
data = pd.merge(almost, veggiePivot, how='left',
                left_on='GRID_ID', right_on='GRID_ID')
data.rename(columns={'Null': 'Barren'}, inplace=True)

```

At this point, we are ready to build our statistical model. It is important to note that important covariates are likely missing from this model. Also, it is important to note that some data management steps that will take a fairly long time if an analyst is only working with a conventional home use desktop (e.g., 12 GB RAM and i5 Intel Processor). Prior to the draft of this report, we simultaneously ran models on more powerful servers that may not be available to FoDP. Given these considerations, it is important to reiterate that findings from our preliminary models are likely biased and not reflective on the true relationship between wildlife and vegetation at Discovery Park. Hopefully, once more data is collected, this will not always be the case. Moreover, it is paramount to mention that to optimize users' experience for "real-time analyses," it would be ideal that the back-end model in this framework is processed on a powerful server. Now that we covered data management for the geostatistical model, let's highlight the data management activities in the simple web application model.

### 5.1.2 The simple web application model

As a consolation deliverable for FoDP in the project, we created a simple web application using ESRI appBuilder © that will help users understand the implications that landcover modifications have on the environmental value of discovery park. Originally, it was requested by the FoDP stakeholder for the application to have the ability to show instantaneous results based on quick “on-the-fly” edits by users. Our background research suggested that odds of meeting this request would be higher if a custom widget (i.e., function) was created using JavaScript. Given that the MGIS analysts did not feel that they had adequate expertise in web development to create the originally requested application, they decided to use existing templates in ESRI’s appBuilder, which could serve as an interactive wireframe.

The feature class backending the simple application is essentially the point scenarios feature class merged with the Earth Economics lookup table so that users would have an understanding of the affordance valuation given five preprocessed situations that can serve as a potential site for residence at Discovery Park. This was done by importing the scenario feature class and lookup table from FoDP’s AGOL account. Next, a pivot table was created from the Earth Economics lookup table by affordance type. Next, the pivot table was joined on the scenario feature class. After that, the Earth Economic affordance values were multiplied by the acreage of each hexbin given the hexbin’s landcover. Finally, the generated dataframe was converted back to a feature class on a MGIS students local computer. To conserve FoDP’s ArcGIS web credits, the MGIS group preferred to do most geoprocessing on their local computers. The code snippet to generate the underlying data for the simple application go as follows:

```
#code snippet doing a pivot on the Earth Economics Lookup table
test=pd.pivot_table(Look,index=['Landcover','Agriculture','Riparian','Urban'],columns='valueType',values='low')
test.reset_index(inplace=True)
test['index1']=test.index
values1=test[['Aesthetic','Air','Carbon Sequestration',
              'Disaster Mitigation','Health','Shelter','Waste','Water']].add_suffix('_low')
values1['index1']=values1.index
test2=pd.pivot_table(Look,index=['Landcover','Agriculture','Riparian','Urban'],
                    columns='valueType',
                    values='high')
test2.reset_index(inplace=True)
values2=test2[['Aesthetic','Air','Carbon Sequestration',
               'Disaster
Mitigation','Health','Shelter','Waste','Water']].add_suffix('_high')
values2['index1']=values2.index
keyTableEE2 =
pd.merge(test[['index1','Landcover','Agriculture','Riparian','Urban']],
        values1)
keyTableEE = pd.merge(keyTableEE2,values2)
del keyTableEE['index1']
keyTableEE.head()
```

```

newOriginal = scene[['FID','GRID_ID','Acres','Latitude','Longitude',
'zoneMGT','LC','Agricultur','Riparian','Urban','Scenario','modified']]
newOriginal.rename(columns={'Agricultur':'Agriculture'}, inplace=True)

preOrig = pd.merge(newOriginal,keyTableEE, how='left',
left_on=['LC','Agriculture','Riparian','Urban'],
right_on=['Landcover','Agriculture','Riparian','Urban'])

preOrig['aestheticLowTotal'] = preOrig['Acres']*preOrig['Aesthetic_low']
preOrig['aestheticHighTotal'] = preOrig['Acres']*preOrig['Aesthetic_high']
preOrig['airLowTotal'] = preOrig['Acres']*preOrig['Air_low']
preOrig['airHighTotal'] = preOrig['Acres']*preOrig['Air_high']
preOrig['carbonLowTotal'] = preOrig['Acres']*preOrig['Carbon Sequestration_low']
preOrig['carbonHighTotal'] = preOrig['Acres']*preOrig['Carbon Sequestration_high']
preOrig['mitigationLowTotal'] = preOrig['Acres']*preOrig['Disaster Mitigation_low']
preOrig['mitigationHighTotal'] = preOrig['Acres']*preOrig['Disaster
Mitigation_high']
preOrig['healthLowTotal'] = preOrig['Acres']*preOrig['Health_low']
preOrig['healthHighTotal'] = preOrig['Acres']*preOrig['Health_high']
preOrig['shelterLowTotal'] = preOrig['Acres']*preOrig['Shelter_low']
preOrig['shelterHighTotal'] = preOrig['Acres']*preOrig['Shelter_high']
preOrig['wasteLowTotal'] = preOrig['Acres']*preOrig['Waste_low']
preOrig['wasteHighTotal'] = preOrig['Acres']*preOrig['Waste_high']
preOrig['waterLowTotal'] = preOrig['Acres']*preOrig['Water_low']
preOrig['waterHighTotal'] = preOrig['Acres']*preOrig['Water_high']
preOrig['lowTotal'] =
preOrig['aestheticLowTotal']+preOrig['airLowTotal']+preOrig['carbonLowTotal']+preOr
ig['mitigationLowTotal']+preOrig['healthLowTotal']+preOrig['shelterLowTotal']+preOr
ig['wasteLowTotal']+preOrig['waterLowTotal']
preOrig['highTotal'] =
preOrig['aestheticHighTotal']+preOrig['airHighTotal']+preOrig['carbonHighTotal']+pr
eOrig['mitigationHighTotal']+preOrig['healthHighTotal']+preOrig['shelterHighTotal']
+preOrig['wasteHighTotal']+preOrig['waterHighTotal']
preOrig=preOrig.drop(['Aesthetic_low','Aesthetic_high','Air_low','Air_high',
'Carbon Sequestration_low','Carbon Sequestration_high',
'Disaster Mitigation_low','Disaster Mitigation_high',
'Health_low','Health_high','Shelter_low','Shelter_high',
'Waste_low','Waste_high','Water_low','Water_high'], axis = 1)
preOrig.fillna(0, inplace=True)

x = np.array(np.rec.fromrecords(preOrig.values))
names = preOrig.dtypes.index.tolist()
x.dtype.names = tuple(names)

arcpy.da.NumPyArrayToTable(x,

```

```

"C:/Users/lewis/Documents/ArcGIS/Projects/FoDP/FoDP.gdb/Scenarios_eg_table")

# convert pop geodatabase table to point file for network analysis
in_table = "Scenarios_eg_table"
out_feature_class = "Scenarios_eg"
x_coords = "Longitude"
y_coords = "Latitude"

# Make the XY event layer for providers.
arcpy.management.XYTableToPoint(in_table, out_feature_class,x_coords,
y_coords, None, arcpy.SpatialReference(2285))

```

## 5.2 Statistical Model Development and Deployment

Though the MGIS students in this capstone project feels as though there is insufficient data to replicate the wildlife and vegetation ecosystem present at Discovery Park, it will be to our stakeholder's benefit if some foundational source code was provided that is capable of handling a large number of interactions. Therefore, the MGIS students decided to implement a random forest regression model to model the correlation between the population density of birds and vegetation. In this dataset, there are over 100 species of birds and almost 20 species of vegetation. Despite the preference of using a robust statistical model, we decided to keep the modeling simple. With respect to statistical modeling, interpreting complex models rivals the challenge of building a complex model. As such, we did not see the need in building a complex statistical model without proof that the findings would be useful. We also decided to use the sciKit-Learn library as it is one of the more established machine learning library used in Python. It is important to note that though ArcGIS has several geoprocessing tools that leverages the sciKit-Learn library, that users must activate (i.e., pip install) the library prior to use. There are several ways to do this, but the pip installation approach in the ArcGIS Python Command Prompt appears to be the easiest approach.

The first step in generating a prediction model that can predict the presence of wildlife given an area's vegetation status is to partition our data. We partitioned the data to prevent overfitting. Overfitting can drastically reduce our odds of reasonably predicting our outcome of interest. In this project we suggest that FoDP implement 10-fold cross validation where the estimates are based on an ensemble of ten models where 10% of our data are removed and compared against predictions to help develop our weights. Accuracy in modeling will be based upon the pooled negative mean absolute error in our ten models. Despite the challenges in interpreting beta weights in a random forest model, the model can output a variable importance factor to give us a hint as to predictors with the largest influence on our wildlife outcome. Code and preliminary findings are described below:

```

X_train, X_test, Y_train, Y_test = train_test_split(good_data[['Big-leaf
Maple', 'Bitter Cherry', 'California Blacberry',

```



```

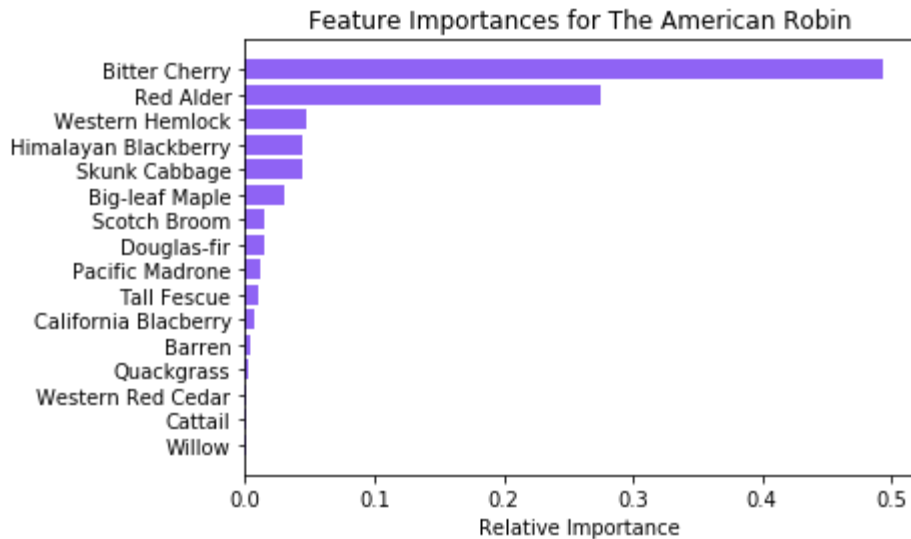
        'Cattail', 'Douglas-
fir', 'Himalayan Blackberry', 'Barren', 'Pacific Madrone',
        'Quackgrass', 'Red
Alder', 'Scotch Broom', 'Skunk Cabbage', 'Tall Fescue',
        'Western
Hemlock', 'Western Red Cedar', 'Willow']],
        good_data[['American Robin']],
        test_size=0,
        random_state=226)

gsc = GridSearchCV(estimator=RandomForestRegressor(),
                   param_grid={'max_depth': range(3,30),
                                'n_estimators': (10, 50, 100, 1000)},
                   cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=1)
grid_result = gsc.fit(X_train, Y_train.values.ravel())
best_params = grid_result.best_params_
rfr = RandomForestRegressor(max_depth=best_params["max_depth"],
                            n_estimators=best_params["n_estimators"],
                            random_state=False, verbose=False)
# Perform K-Fold CV
scores = cross_val_score(rfr, X_train, Y_train.values.ravel(), cv=10,
                          scoring='neg_mean_absolute_error')
rfr.fit(X_train, Y_train.values.ravel())
scores

features = X_train.columns.values
importances = rfr.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances for The American Robin')
plt.barh(range(len(indices)), importances[indices], color='#8f63f4',
         align='center')
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
plt.show()

```



**Fig 30.** Feature importance results for the American Robin

Now that we have our base model built, now it is time to predict out values. To do this we will essentially feed in the vegetation status of each of our five preprocessed scenarios to generate predictions of American Robin density per each hexbin. An example, of the python syntax to generate predictions is below:

```
scenario1['pred_AmericanRobin'] = rfr.predict(scenario1[['Big-leaf Maple', 'Bitter Cherry', 'California Blackberry', 'Cattail', 'Douglas-fir', 'Himalayan Blackberry', 'Barren', 'Pacific Madrone', 'Quackgrass', 'Red Alder', 'Scotch Broom', 'Skunk Cabbage', 'Tall Fescue', 'Western Hemlock', 'Western Red Cedar', 'Willow']])
```

### 5.3 Model Integration in ArcGIS AppBuilder

Since many details about appBuilder were discussed previously in this report, we will just highlight details about widgets used in the simple “What-If” Analysis application. As mentioned in section 5.1.2., the simple application is based on five preprocessed scenarios where we simulate a case where city government officials are planning to have low-income housing at Discovery Park. The feature class that was previously created using Earth Economics affordance values were published as a Web Mapping Application in FoDP’s AGOL account. From there the web map was imported into ESRI’s appBuilder. In this application, users can use a filter that alters the affordance values based upon the low-income housing location of choice (displayed in **Figure 31**). The *Dashboard Theme* template was used because it has a preset template that allows for a large number of chart and numeric display options.

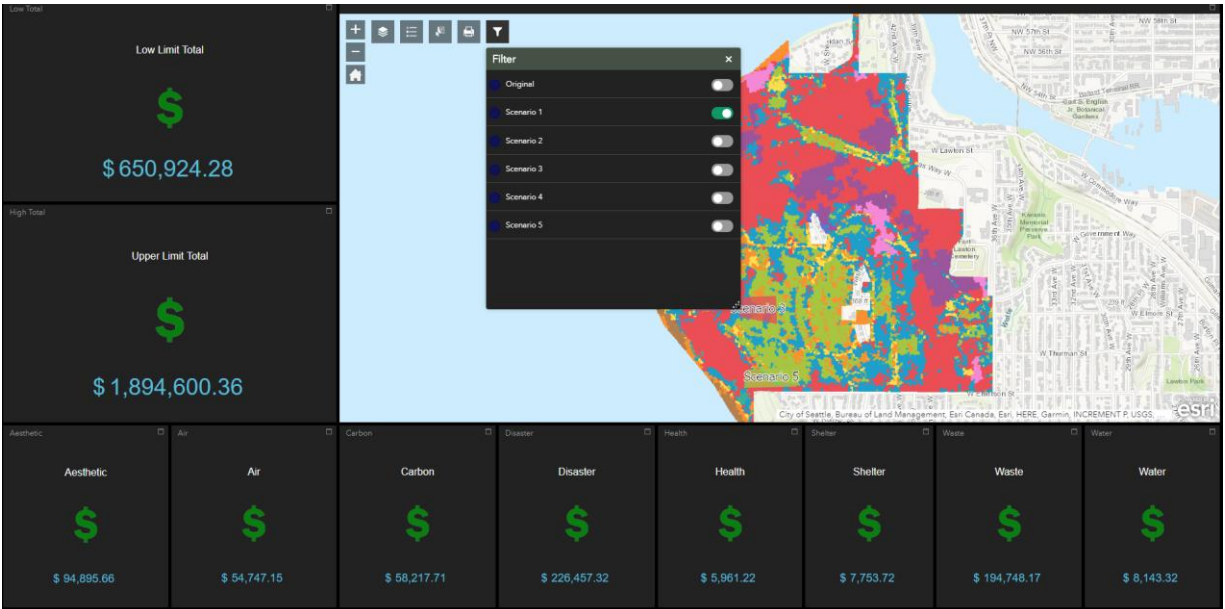


Fig 31. "What-IF" Analysis Application

The widgets of choice with respect to this application were 10 infographic widgets, which aggregates affordance values based upon the scenario chosen. A filter widget that allows users to select a particular scenario (as displayed in **Figure 31**). Moreover, there is a *select* widget that allows users to estimate affordances within the discovery park area (as displayed in **Figure 32**). Aside from some of the default widgets, the application also has a print widget that will allow users to print and share their analyses to facilitate data-driven dialogue.

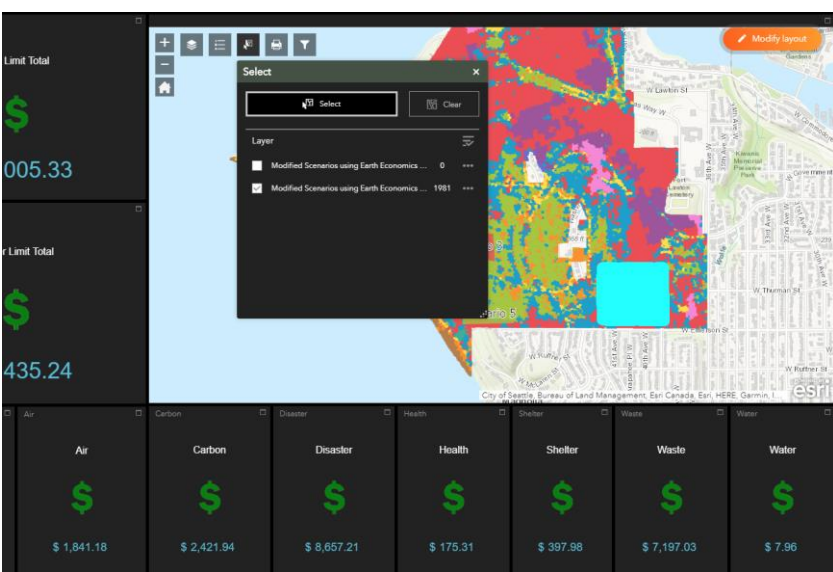
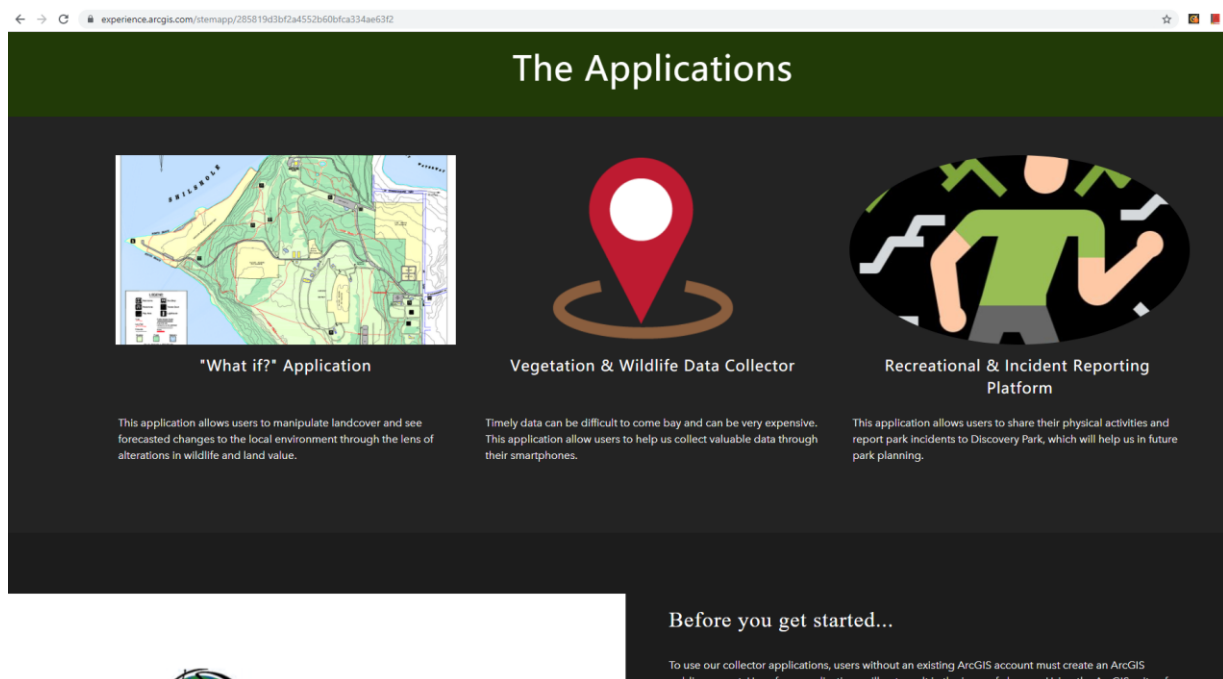


Fig 32. "What-IF" Analysis Application - Select Widget

## 6. Landing Page

### 6.1 ArcGIS Experience Builder

One of the key challenges we faced when piecing together applications was that last year's cohort had a dense amount of information within a single application. Moreover, as we continued to work on the project the application got even more complex and crowded with information. To increase the odds of information literacy among our prospective users, we decided to have our applications hosted on a website. Since we do not have extensive website development skills, we decided to use one of ESRI's newly released software applications, Experience Builder.<sup>15</sup> At the moment, Experience Builder is in Beta Phase and has not officially been released with full documentation and support. That being said, the functionality of the application is very similar to appBuilder. One of the primary differences between Web appBuilder and Experience Builder is that Experience Builder has more extensive web building technologies. Though knowledge of JavaScript is still recommended to develop powerful web application, Experience Builder provides more flexibility and functionality without having to know great detail about JavaScript. A visual of our website is displayed in **Figure 33**.

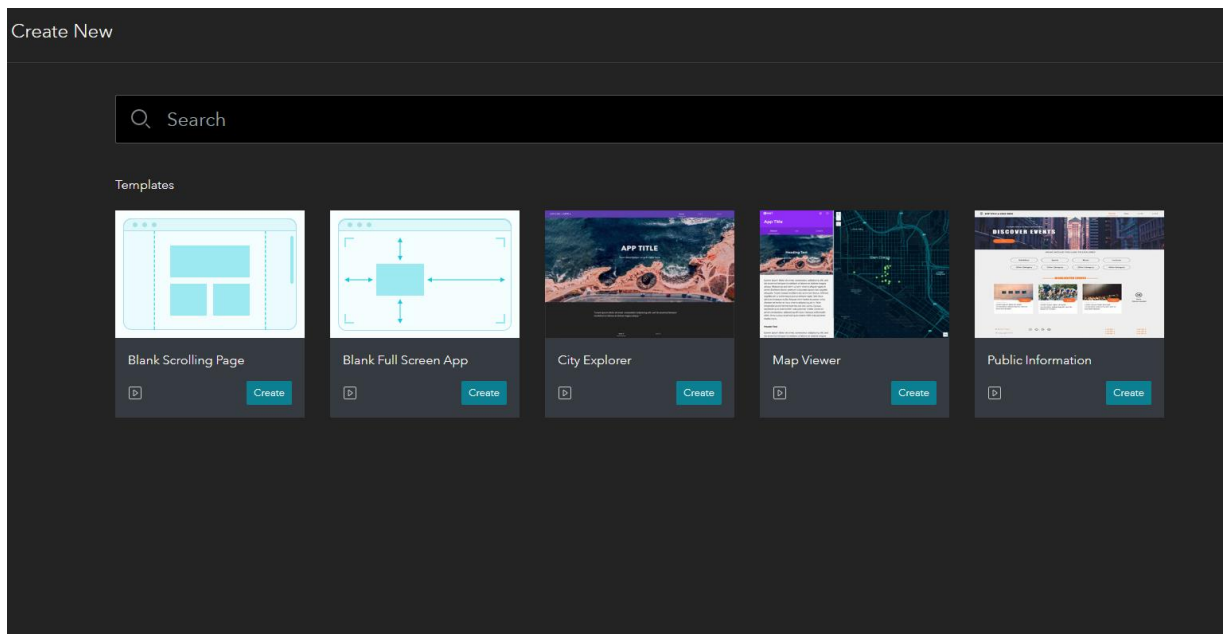


**Fig 33.** FoDP Web Application Landing Website

<sup>15</sup> <https://www.esri.com/en-us/arcgis/products/arcgis-experience-builder/overview>

## 6.2 How to Embed Image, Other Websites, and External Applications to Existing Website

Due to the limited documentation and low user-group presence on the Internet, we used a pre-existing template, *City Explorer*, that had the most primary functions such as hyperlinks and tiles to more easily build a website (**Figure 34**). From there we were able to right click and change hyperlinks to existing FoDP websites and Story Maps. Moreover, we were able to integrate our Story Map presentation for our capstone presentation. One key challenge with Experience Builder being in Beta Phase is that (at the moment), users are unable to share websites to the public. During an instructor JavaScript API training session hosted by ESRI, one of the MGIS group members were told that the application should officially launch before the end of 2019, and that at that point, users will be able to share their website applications to the public.



**Fig 34.** Experience Builder Opening Screen

**Figure 35** shows a screenshot of a user hyperlinking another application through an image. Within the right side of the screen you will see the blue bar labeled “set link”, which allowed us to type in the url of the “What-If” tool. By selecting the set button on the right side of the screen, we are able to select an image as an icon. We are also able to create a live window where users can use applications and web maps without leaving the screen.

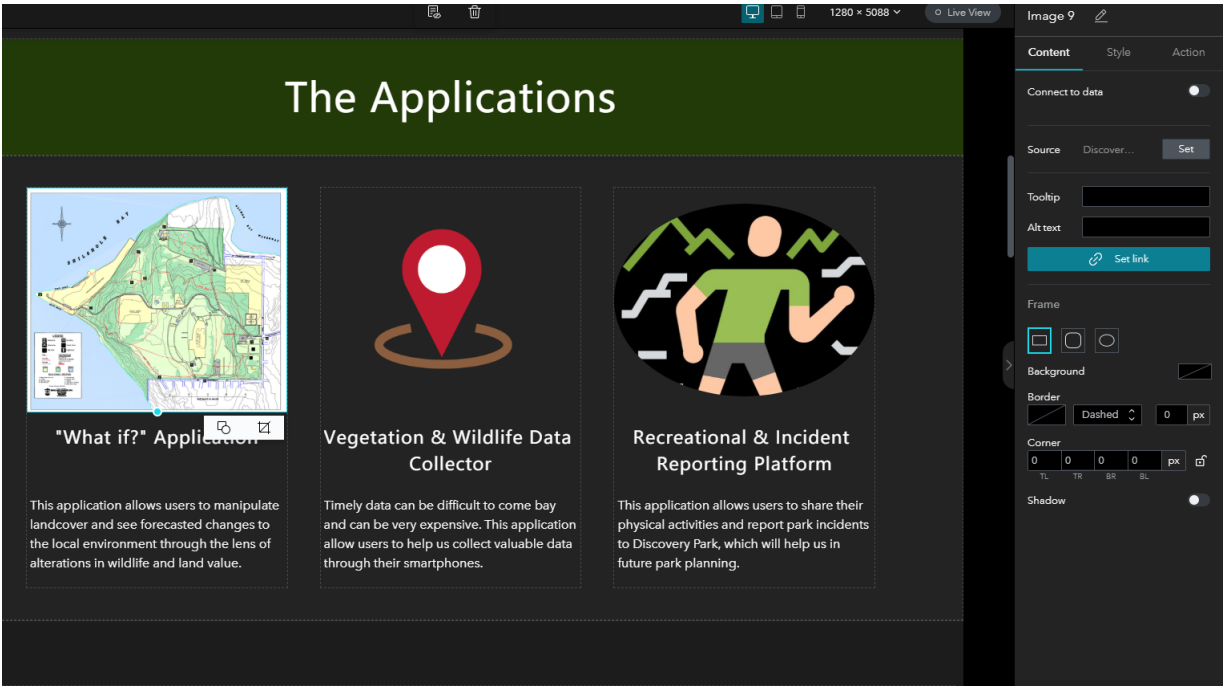


Fig 35. Hyperlinking and embedding applications

## 7. Recommendations

### 7.1 Discovery Park Virtual Management Platform

The current Discovery Park VMP is constructed primarily with ESRI's ArcGIS Online systems and applications. The landing page provides a centralized interface with ESRI's experience builder and directs user to three applications for visualizing, collecting, and analyzing park data. The current design concept of the VMP focuses on leveraging available public sources to provide updated datasets for the analysis model to execute predictive analysis of Discovery Park's vegetation and wildlife. The data collectors were developed with public audiences in mind to provide an interface for public user to establish long-term relationships with the collector.

#### 7.1.1 Data Collection System

Provided that data collection hinges on public sources, the data collector uses ESRI's Web AppBuilder to provide a user-friendly interface that facilitates two-way interaction between Discovery Park management and public users. In order to fuel the analytical model, up-to-date vegetation and wildlife data is essential for model accuracy. Hence, the MGIS 2019 cohort recommends FoDP to focus future effort in advertising and marketing VMP data collector to the public. An informational or educational session would be crucial to accurate data reporting and application usage. Provided that publicly sourced dataset tends to introduce varied data

integrity, data maintenance should be facilitated simultaneously to ensure reliable performance of the “What if” analysis model.

## 7.2 Professional Consultation

The “What if” analysis model provides predictive scenarios of change in Discovery Park’s ecosystem value. To fully understand the relationships between individual ecosystem services and the covariance of collective service, professional consultation should be considered by FoDP in the decision-making process. The current model does not provide a post-analysis report detailing relationships of calculated output.

### 7.2.1 Strava Integration

One of the resources examined during the project cycle was Strava, provided Strava Metro offers partnership for organization that plans with recreational data from local Strava users. FoDP should consider the option of leveraging on non-profit status to apply for access to Strava’s enterprise datasets. The process for user to report trail activities is more cumbersome for the average voluntary audience without the streaming support from ArcGIS Collector Classic<sup>16</sup>. With the added volatility in data integrity for crowd-sourced datasets, recreational analysis can be proven to be inaccurate and less efficient. Leveraging on an existing database for recreational analysis of Discovery Park will provide more effective results for future funding opportunities.

## 7.3 Human-Nature Interaction

One of the requested relationships to examine within FoDP VMP project was human-nature interaction. This interaction is characterized by the preferred physical and mental exposure to nature by individuals that visits the park. Without having a robust dataset, no analytical framework was able to be developed within the project cycle. The RIRP section of the current VMP serve as a foundational framework for FoDP to establish future studies of human-nature interaction within Discovery Park. The current surveying questionnaire does not offer an extensive coverage of documented variability for human-nature interaction from Kahn et al. (2019)<sup>17</sup>. As VMP data collector becomes more available to the public, FoDP should revise existing questionnaire for data collection to enhance data quality for human-nature interaction studies.

---

<sup>16</sup> [ArcGIS Collector Classic](#)

<sup>17</sup> Manuscript provided by Garrett Esperum from Friends of Discovery Park





APPENDICES

## Appendix 1: System Resource Requirements

For this project, the minimum resource technological requirements were a suite of ESRI products, a database management system, and Python. The ESRI products used in this project was ArcGIS Pro, Web appBuilder, AGOL, and Experience Builder. Though we were capable of using AGOL's cloud as a database, it was very labor intensive for use to run powerful analyses on our PC which may or may not display errors in a timely manner. As such, we also leveraged use of a Microsoft SQL Server, to make some of the "big data" processing more tolerable. Also, given FoDP preference of coding and archiving as much information as possible on their AGOL account, having strong internet connections for our working sessions and weekly meetings outside of class was very important.

## Appendix 2: Business Case Evaluation

### 1.3 Overarching Approach in the Quantification of Costs and Benefits

The assessment of costs and benefits is a critical aspect of any GIS project given that it provides a rationale for engaging in the endeavour. While the qualitative aspect of our project’s benefits was assessed by the University of Washington’s Geography Department prior to the start of the capstone project, this section of our report will highlight the business case for this project. It is important to note that given our adjusted objectives, the empirical benefits of our efforts may not fully materialize until years of data collection. As such, some assumptions in our business case may heavily influence the outcomes of our case analyses. The business case focuses on work associated with developing the statistical framework and data collection applications. Based upon a previous study conducted by Earth Economics, we will anchor our analysis on promoting the preservation of Discovery Park, which is an asset valued anywhere between \$556,978 and \$1,079,587 (breakdown displayed in **Table 1**). Therefore, though the return of investment in this project may not be immediate, the legitimacy of costs associated with this project is based upon the risks of reducing an asset worth at least half a million dollars every year. It was also recommended by Earth Economics to evaluate Discovery Park’s valuation to use the “High” estimates presented in **Table 1**. The ecosystem services were evaluated across the Washington state using various research literature and field survey observations<sup>18</sup>. The condition and services provided by Discovery Park was quantified using the full spectrum of presented within Washington state.

**Table 1.** Earth Economics’ Assessment of Discover Park’s Value per Year

Service	Low	High
Air	\$ 54,824	\$ 54,824
Climate Stability	\$ 226,771	\$ 374,750
Biological Control	\$ 5,964	\$ 5,964
Shelter	\$ 7,793	\$ 20,709
Waste	\$ 195,109	\$ 379,620
Water	\$8,149	\$ 89,804
Carbon	\$ 58,368	\$ 153,916
<b>Total</b>	<b>\$ 556,978</b>	<b>\$ 1,079,587</b>

<sup>18</sup> [Earth Economics Central Puget Sound Open Space Evaluation \(2015\)](#)

## 1. Benefit

2.1 Type 1 Benefits: Quantifiable efficiencies in current practices, or benefits that reflect improvements to existing practices

The 2018 MGIS cohort in their capstone project reported that FoDP “spent an average of 8 hours per week across the past 18 months on this particular project” prior to the start of their project.<sup>19</sup> Rich, Peterson, and Marshall (2018) also mentions that the governing body for Discovery Park, Seattle Parks and Recreation, values volunteers at an estimated benefit of \$25 an hour. After a skim of the internet to value the costs of GIS services in the free market,<sup>20</sup> we came across a comparable estimate of \$25.65 an hour. For the geostatistical component of our project, we assume that at the develop framework would not require major modifications should the server FoDP is using have the minimum technological specifications of our groups computer resources (i.e., laptops with 16 GB of RAM on Intel i5 processor). The random forest model is a data mining model that has a decent and somewhat self-sustaining algorithm for variable selection. As such, as long as the independent variables are compiled as a list, then we would not have to make any major coding changes to the geostatistical model. Should Scikit-Learn depreciate some components of its library, then adjustments will need to be made. Based upon previous conversations with FoDP, we are confident that they are capable of adjusting code based upon recommended modifications in the Python © log should any components in SciKit Learn approach depreciation in the near future. We estimate that one hour per week will be saved based upon our contribution of the geostatistical model. We also integrated a data collection infrastructure, which significantly reduce the need to collect various data sources external of FoDP’s repository. We estimate that this would provide a time savings of seven hours of FoDP and data will be managed and auto-populated through the web application. That being said, there needs to be some management of the platform’s data repository as issues may arise. Though we do not forecast any major issues, we do think that it would be beneficial for FoDP to invest at minimum one hour a week in managing the data collection system.

**Table 2.** Type 1 Benefits Calculated as Reduced Work Hours

FoDP’s original weekly efforts * (hours)	8
Anticipated effort after VMP development (hours)	1
Time savings due to VMP	7
Value of work *	\$25/hour
Savings	\$175/week

\* Also quoted by Rich, Peterson, and Marshall (2018)

<sup>19</sup> [https://depts.washington.edu/mgis/capstone/files/GEOG569\\_2018\\_Rich\\_Peterson\\_Marshall.pdf](https://depts.washington.edu/mgis/capstone/files/GEOG569_2018_Rich_Peterson_Marshall.pdf)

<sup>20</sup> <https://www.indeed.com/salaries/GIS-Analyst-Salaries>

## 2.2 Type 2 Benefits: Quantifiable expanded capabilities, or benefits that offer added capabilities

There were challenges in the valuation of the expanded capabilities of the VMP due to our groups contribution. Discovery park currently does not have an active infrastructure to have real-time data. Services such as Strava © costs \$0.80 per unique Strava user with activity reported in area of interest (first 10,000 Strava user with minimum of \$1,000)<sup>21</sup>. The license fee lowers to \$0.70 per user when the first 10,000 user count is reached. This pricing guideline is provided version 2.0 of the Strava © user guideline. The most current guideline does not detail specific licensing fees for purchasing the data<sup>22</sup>. Despite the enticing offer from Strava Metro ©, Discovery Park, with an visitorship of 250,000 per year<sup>23</sup>, if only 10% of the visitor uses Strava, then an annual cost of \$18,500 is estimated for FoDP to maintain recreational data for analysis. Through using Strava Metro © service, FoDP would not have full ownership of the data and data cost would vary from location to location based on the current floating rate licensing fee. If Discovery Park were to have in-house data collectors for vegetation and wildlife, we could assume that this wouldn't be a \$40/week position and would most likely pay minimum wage (\$15/h in Seattle). To be safe the park would need at least two part-time data collectors to make any kind of progress, which would put the cost of data collectors at ~\$600 a week. So if we assume that FoDP would be gathering this data regardless of if we created the collection app, our open source collection app will save the park on average \$31,200. The cost of a GIS analyst to complete a project for a single analysis is projected to be \$3,000 should it take an analyst 120 hours to structure data, check for errors, run model, and provide intelligence at a rate of \$25 an hour. Though the geostatistical framework would not be an actual artificial intelligence framework that would provide some form of business intelligence in an automated fashion, the proposed framework should provide the ability to provide both archived and “current” statistical relationships without an analyst having to spend too much time generating analyses. Assuming that this would be done once a year, this would provide a benefit of an additional \$58 a week (i.e., \$3,000/52 weeks in a year). Calculations are displayed in **Table 3**.

---

<sup>21</sup> [Strava Metro User Guide Version 2.0](#)

<sup>22</sup> [Strava Metro Comprehensive User Guide Version 8.0](#)

<sup>23</sup> [Discovery Park's One-Millionth Visitor](#)

**Table 3.** Type 2 Benefits

	Geostatistical Model
Original	\$175/week
Geostatistical Model Added Benefit	\$58/week
Recreational Activity Collector Added Benefit	\$355.77/week
Wildlife and Veg. Collector Added Benefit	\$600/week
Net Gain Per Week	\$413.77/week
Total Efficiency Gained	\$646.77/week

### 2.3 Type 3: Quantifiable unpredictable events or benefits that result from unpredictable events

Rich, Peterson, and Marshall (2018) in their MGIS capstone project highlighted the West Point Treatment Disaster (page 9<sup>24</sup>) as an example of the use of the VMP for an unpredictable event. Since we do not want to be redundant and highlight that case as an example. Interestingly, it was brought to our attention by our FoDP contact that the city of Seattle once explored the alternative uses of Discovery Park, which may alter the use and landcover of Discovery Park. More specifically, it was thought that the city might explore use of Discovery Park to serve as temporary shelter for individuals who are homeless in the Seattle metropolitan area. Though this event is not unpredictable (per se), it is an event that is beyond the control of FoDP. This means that this tool may aid in the decision making process of Discovery Parks use by the public. We believe that the VMP could be used to assist the city in locating these shelters should the city decided to implement this project. The VMP could assist stakeholders in understanding the financial implications of locating shelter camps at particular locations. Though there would be residual value lost in a project such as homeless camps at Discovery Park, we will assume that the platform would suggest locating these camps at landcovers that are habitable and Earth Economics value at \$0. To value this event, we will assume that all areas of the park except for water bodies and forest could be used for a homeless camp project. Given that all possible landcovers not excluded will be eligible to be selected as an area for a homeless camp, we project the impacted landcover that could decrease in value to have a worth at approximately \$213,076 a year (or roughly \$5,000 a week). This estimate is the median value of landcover eligible to be selected as appropriate for a homeless camp. We'll assume that an endeavour like this to be viable for only nine months out of the year due to weather (38 weeks). We also project this event to have a probability of once every 50 years (i.e., 2,600 weeks) given our FoDP confidence that the event will likely not happen due to political restructuring. Similar to Rich, Peterson, and Marshall's (2018) projections, we will assume that the VMP will improve the odds of benefits by 33%. As a result of our assumptions, we believe that the VMP will result in a cost savings of **\$24** a week for an unpredictable event related to long term use of Discovery Park land for shelter related purposes. Calculations are displayed in **Table 4**.

<sup>24</sup> [https://depts.washington.edu/mgis/capstone/files/GEOG569\\_2018\\_Rich\\_Peterson\\_Marshall.pdf](https://depts.washington.edu/mgis/capstone/files/GEOG569_2018_Rich_Peterson_Marshall.pdf)

Table 4. Type 3 Benefits

Worth of Land Capable for Homeless Shelter	\$5,000/week
Duration of Shelter Project if Implemented	38 weeks
Probability of Shelter Project Occuring	1:50 years (or 1:2,600 weeks)
Projected Mitigation of Landcover Devaluation Due to VMP	33%
Total Weekly Improvement in Impact	\$24/week $(\{ \$5,000 * 38 \} / 2,600) * 0.33$

#### 2.4 Type 4: Intangible benefits

Given that the motivation for the VMP is for FoDP to get other stakeholders to understand the value of Discovery Park’s green space, we will use the intangible benefits purported by Rich, Peterson, and Marshall (2018) in their MGIS capstone project. They identified the park’s green space as an area of interest for the intangible benefits. This sentiment was also shared by our FoDP administrative contact. In their analyses, they projected the benefit to be **\$523 a week**, which was based from a 0.1% improvement rate of the conservative \$541,917.40 value of parcels at Discovery Park.

## 2. Cost

### 3.1 Capital Costs

#### *Research and Tool Development*

With respect to the research and development costs associated with this project, we define research as effort spent learning Python ©, ArcGIS ©, and scholarly content needed to successfully implement this project. The tool development is defined as effort spent managing input data, coding to develop analytic model, effort spent to develop data collection applications, and time spent debugging our code and applications. The research and tool development costs are displayed in **Table 5**.

Table 5. Hours worked performing research and developing the tool.

Week	DL Research (hours)	RH Research (hours)	AL Research (hours)	DL Develop (hours)	RH Develop (hours)	AL Develop (hours)	Total Hours
Week 1	8	8	8	0	0	0	24
Week 2	8	8	8	0	0	0	24
Week 3	4	4	4	4	4	4	24
Week 4	4	4	4	4	4	4	24
Week 5	4	4	4	4	4	4	24
Week 6	2	2	2	8	8	8	30
Week 7	2	2	2	8	8	8	30
Week 8	1	1	1	8	8	8	27
Week 9	0	0	0	8	8	8	24
Total hours							231
Hourly rate							\$25.00
Total							\$5,775.00

DL=Dwight Lewis; RH=Ray Hall; AL=Austin Lin

### *Hardware and Software*

Fixed costs associated with hardware and software are constrained due to the non-profit status of FoDP. Due to their non-profit status, they are able to use ESRI's software for essentially free. To the best of our knowledge, they have computer resources that are already depreciated. So, similar to logic used by Rich, Peterson, and Marshall (2018), we display the hardware and software costs in **Table 6**.



Table 6. Hardware and software costs

Estimated Hours Worked per Week Using Computer Across Members	22
Computer Value	\$3,000*
Estimated Mean Life of Computer in Hours	5,480 hours**
Weekly Cost of Computer Usage	\$12.04
Computer Use During Project	\$108.36

\*Collective cost of Dwight Lewis and Austin Lin’s personal laptop

\*\*rate used by Rich, Peterson, and Marshall (2018)

### 3.2 Operating Costs

#### *Personnel*

As mentioned, the collected data resource and data collection system will require a minimum of one hour per week of maintenance by on-sight technician to maintain the estimated benefits. With the current job market rate of \$17.27/hr, an estimated annual cost of **\$898.04** can be expected<sup>25</sup>.

#### *Consultation*

To fully maximize user collected data it is recommended to work with experters in the fields or ornithology, botany, biology, etc. to act as yearly consultants. Having experts in the fields of study related to the data collected will verify that the attributes collected by the data are useful in identification and analysis. With a 2014 survey stating that the annual income for a PhD Biologist consultant is \$115,000 or roughly \$55/h<sup>12</sup>. We believe that any consultant working on this application would only need to be used for a week every year evaluating the data that has been collected and updating any attributes that may need be added, costing FoDP \$2,200.

#### *Maintenance Fees*

FoDP’s non-profit status provides an Creator-level ESRI ArcGIS Online license with many online resources available to the account. Previous cohort communicated with FoDP and documented a budgeted credit count of 200 credits annually for FoDP’s AGOL account<sup>26</sup>. Associated storage cost of currently available feature and hosted services are documented by Rich, Peterson, and Marshall (2018). Due to the nature of our effort in forwarding the VMP project, no further data storage is expected from phase 2 of the VMP project. However, it is worth noting the final deliverable will likely result in an increased storage cost of vector feature services alongside the data collection application we will be delivering to FoDP to achieve our adjusted project objective.

#### *Utilities, Supplies, and Other*

At the current state of the VMP project, there are no foreseeable increase in infrastructure cost of the platform. Hence, we would extend our estimation of infrastructure and equipment upkeep through the

<sup>25</sup> [Entry-Level GIS Technician Salary from PayScale](#)

<sup>26</sup> [https://depts.washington.edu/mgis/capstone/files/GEOG569\\_2018\\_Rich\\_Peterson\\_Marshall.pdf](https://depts.washington.edu/mgis/capstone/files/GEOG569_2018_Rich_Peterson_Marshall.pdf)

estimation performed by the previous cohort. As estimated electricity cost performed by Rich, Peterson, and Marshall (2018) with the publically available electricity data browser for the United States. With the updated information on May 2019, the estimated cost for retail/commercial electricity usage totals to \$0.1053/kWh<sup>27</sup>. Despite the “negligible” cost of utility estimated by the electricity cost for FoDP, it was discussed in our sponsor meeting that an eventual goal for a locally hosted park management interface can be expected towards the end of the FoDP VMP project. With the expansion in information infrastructure, FoDP can expect an increase in utility and equipment cost annually.

### 3. Benefit-Cost Analysis

		Weekly Value (8-week period)
Type 1 Benefits (Efficiency)	MGIS Voluntary Effort	\$ 175
Type 2 Benefits (Progression)	Geostatistical Analysis Model	\$ 58
	Reactional Data	\$ 355.77
	Wildlife and Vegetation Data	\$ 600
Type 3 Benefits (Unpredictable)	Social Impact	\$ 24
Type 4 Benefits (Foundational)	Green-space Value	\$ 523
Capital Cost		-\$ 721.88
Hardware/Software Cost		-\$ 12.04
Personnel Cost		-\$ 17.27
Consultation Cost*		-\$2,200/Year or \$42/Week
Maintenance Cost		TBA <sup>28</sup>
Utility/Infrastructure Cost		Annually 15kWh per sq ft <sup>29</sup>
	Sum	\$ 942.58**

\* Annual cost of a one week process.

\*\* Calculated from weekly adjusted Consultation cost

#### Important Assumptions

- Associated cost for utility and infrastructure is using national averages

<sup>27</sup> [Electricity Data Browser - Average Retail/Commercial Electricity Price](#)

<sup>12</sup> <https://www.sciencemag.org/careers/2014/09/science-careers-guide-consulting-careers-phd-scientists>

<sup>28</sup> [ArcGIS Online Credit Pricing](#)

<sup>29</sup> [Electricity Cost for Small to Mid-size Office](#)

- Maintenance cost can be counted as 0 if credit usage falls within 200
- Capital cost can be front loaded
- Benefits will be lasting so long system is maintained

## Appendix 3: Python Source Code for Model in “What-If” Analysis Application

### Full Code--“What-IF” Analysis

(note: full code with comments can be seen

here:[http://htmlpreview.github.io/?https://github.com/lewis060-UAT/FreindsOfDiscoverParkAnalysis/blob/master/What%20If%20Input%20Generation%20-%20FoDP%20\(Final\).html](http://htmlpreview.github.io/?https://github.com/lewis060-UAT/FreindsOfDiscoverParkAnalysis/blob/master/What%20If%20Input%20Generation%20-%20FoDP%20(Final).html) )

```
import arcpy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matplotlib
import getpass
from arcgis.gis import GIS
from arcgis.mapping import WebMap
from arcgis.features import SpatialDataFrame
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', -1)
direct="C:/Users/lewis/Documents/GEOG 569/"

# Set environment settings for database so that there isn't a need to type out full
# pathnames in arcpy library
arcpy.env.workspace = "C:/Users/lewis/Documents/ArcGIS/Projects/FoDP/FoDP.gdb"

u = getpass.getpass(prompt='Enter username')
p = getpass.getpass(prompt='Enter password')
gis = GIS('https://fodp.maps.arcgis.com/',u, p)

search_result = gis.content.search(query='Earth Economics Lookup Table')
#This is a single table, so no need to import layers like the geostatistical
#example
dataSource = search_result[0].tables
#Lets import the Lookup table fom Earth Economics
Look = dataSource[0].query().sdf

#data dictionary for the Look-up tables
lookDictionary = Look.dtypes.to_frame('dtypes').reset_index()
lookDictionary.columns = ['Column', 'Data Type']
column = [
lookDictionary['Column'] == 'Agriculture',
lookDictionary['Column'] == 'Landcover',
lookDictionary['Column'] == 'ObjectId',
lookDictionary['Column'] == 'Riparian',
lookDictionary['Column'] == 'Urban',
```

```

lookDictionary['Column'] == 'high',
lookDictionary['Column'] == 'low',
lookDictionary['Column'] == 'valueType'
]
descriptions = [
'Indicator (0-1) whether the landcover is agrarian',
'Landcover type',
'Default index for feature classes created by ArcGIS',
'Indicator (0-1) whether the landcover is riparian',
'Indicator (0-1) whether the landcover is urban',
'Upper limit dollar value for affordance',
'Lower limit dollar value for affordance',
'Affordance type (Aesthetics, Air, Carbon Sequestration,...)'
]
lookDictionary['Description'] = np.select(column,descriptions, default='error')

test=pd.pivot_table(Look,index=['Landcover','Agriculture','Riparian','Urban'],
columns='valueType',
values='low')
test.reset_index(inplace=True)
test['index1']=test.index
values1=test[['Aesthetic','Air','Carbon Sequestration',
'Disaster Mitigation','Health','Shelter','Waste','Water']].add_suffix('_low')
values1['index1']=values1.index
test2=pd.pivot_table(Look,index=['Landcover','Agriculture','Riparian','Urban'],
columns='valueType',
values='high')
test2.reset_index(inplace=True)
values2=test2[['Aesthetic','Air','Carbon Sequestration',
'Disaster
Mitigation','Health','Shelter','Waste','Water']].add_suffix('_high')
values2['index1']=values2.index
keyTableEE2 =
pd.merge(test[['index1','Landcover','Agriculture','Riparian','Urban']],
values1)
keyTableEE = pd.merge(keyTableEE2,values2)
del keyTableEE['index1']

search_result = gis.content.search(query="ScenariosFeature_pts")
#The required data is a "Feature Layer Collection" so it's the only option (i.e., indexed at 0)
dataSource = search_result[0].layers
for lyr in dataSource:
print(lyr.properties.name)

#Importing the point data representing Landcover at
#Discovery Park aggregated to the centroids of hexbins

```

```

scene = dataSource[0].query().sdf

#data dictionary for the polygon hexbins
sceneDictionary = scene.dtypes.to_frame('dtypes').reset_index()
sceneDictionary.columns = ['Column','Data Type']
column = [
sceneDictionary['Column'] == 'Acres',
sceneDictionary['Column'] == 'Agricultur',
sceneDictionary['Column'] == 'FID',
sceneDictionary['Column'] == 'GRID_ID',
sceneDictionary['Column'] == 'LC',
sceneDictionary['Column'] == 'Latitude',
sceneDictionary['Column'] == 'Longitude',
sceneDictionary['Column'] == 'Riparian',
sceneDictionary['Column'] == 'SHAPE',
sceneDictionary['Column'] == 'Scenario',
sceneDictionary['Column'] == 'Urban',
sceneDictionary['Column'] == 'modified',
sceneDictionary['Column'] == 'zoneMGT'
]
descriptions = [
'The acreage of the hexbin',
'Indicator (0-1) whether the landcover in hexbin is agrarian',
'Feature class index created by ArcGIS',
'Indexed unique identifier for hexbin',
'Landcover type',
'Latitude coordinate (feet) for hexbin centroid',
'Longitude coordinate (feet) for hexbin centroid',
'Indicator (0-1) whether the landcover in hexbin is riparian',
'Geometry object of feature class',
'Identifier (0-5) of the scenario for landcover manipulations',
'Indicator (0-1) whether the landcover in hexbin is urban',
'Indicator (0-1) whether the landcover in hexbin is different from original',
'Description of zone area within Discover Park'
]
sceneDictionary['Description'] = np.select(column,descriptions, default='error')
sceneDictionary

newOriginal = scene[['FID','GRID_ID','Acres','Latitude','Longitude',
'zoneMGT','LC','Agricultur','Riparian','Urban','Scenario','modified']]
newOriginal.rename(columns={'Agricultur':'Agriculture'}, inplace=True)

preOrig = pd.merge(newOriginal,keyTableEE, how='left',
                    left_on=['LC','Agriculture','Riparian','Urban'],
                    right_on=['Landcover','Agriculture','Riparian','Urban'])
preOrig['aestheticLowTotal'] = preOrig['Acres']*preOrig['Aesthetic_low']

```

```

preOrig['aestheticHighTotal'] = preOrig['Acres']*preOrig['Aesthetic_high']
preOrig['airLowTotal'] = preOrig['Acres']*preOrig['Air_low']
preOrig['airHighTotal'] = preOrig['Acres']*preOrig['Air_high']
preOrig['carbonLowTotal'] = preOrig['Acres']*preOrig['Carbon Sequestration_low']
preOrig['carbonHighTotal'] = preOrig['Acres']*preOrig['Carbon Sequestration_high']
preOrig['mitigationLowTotal'] = preOrig['Acres']*preOrig['Disaster Mitigation_low']
preOrig['mitigationHighTotal'] = preOrig['Acres']*preOrig['Disaster
Mitigation_high']
preOrig['healthLowTotal'] = preOrig['Acres']*preOrig['Health_low']
preOrig['healthHighTotal'] = preOrig['Acres']*preOrig['Health_high']
preOrig['shelterLowTotal'] = preOrig['Acres']*preOrig['Shelter_low']
preOrig['shelterHighTotal'] = preOrig['Acres']*preOrig['Shelter_high']
preOrig['wasteLowTotal'] = preOrig['Acres']*preOrig['Waste_low']
preOrig['wasteHighTotal'] = preOrig['Acres']*preOrig['Waste_high']
preOrig['waterLowTotal'] = preOrig['Acres']*preOrig['Water_low']
preOrig['waterHighTotal'] = preOrig['Acres']*preOrig['Water_high']
preOrig['lowTotal'] =
preOrig['aestheticLowTotal']+preOrig['airLowTotal']+preOrig['carbonLowTotal']+preOr
ig['mitigationLowTotal']+preOrig['healthLowTotal']+preOrig['shelterLowTotal']+preOr
ig['wasteLowTotal']+preOrig['waterLowTotal']
preOrig['highTotal'] =
preOrig['aestheticHighTotal']+preOrig['airHighTotal']+preOrig['carbonHighTotal']+pr
eOrig['mitigationHighTotal']+preOrig['healthHighTotal']+preOrig['shelterHighTotal']
+preOrig['wasteHighTotal']+preOrig['waterHighTotal']
preOrig=preOrig.drop(['Aesthetic_low','Aesthetic_high','Air_low','Air_high',
'Carbon Sequestration_low','Carbon Sequestration_high',
'Disaster Mitigation_low','Disaster Mitigation_high',
'Health_low','Health_high','Shelter_low','Shelter_high',
'Waste_low','Waste_high','Water_low','Water_high'], axis = 1)
preOrig.fillna(0, inplace=True)
x = np.array(np.rec.fromrecords(preOrig.values))
names = preOrig.dtypes.index.tolist()
x.dtype.names = tuple(names)
arcpy.da.NumPyArrayToTable(x,
"C:/Users/lewis/Documents/ArcGIS/Projects/FoDP/FoDP.gdb/Scenarios_eg_table")
# convert pop geodatabase table to point file for network analysis
in_table = "Scenarios_eg_table"
out_feature_class = "Scenarios_eg"
x_coords = "Longitude"
y_coords = "Latitude"
# Make the XY event Layer for providers.
arcpy.management.XYTableToPoint(in_table, out_feature_class,x_coords,
y_coords, None, arcpy.SpatialReference(2285))

```

## Full Code--Statistical Model

(note: full code with comments can be seen here:)

```
import arcpy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matplotlib
import sklearn as sklearn
from matplotlib.ticker import FuncFormatter
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
import os
import getpass
from arcgis.gis import GIS
import sys
from arcgis.mapping import WebMap
from arcgis.features import SpatialDataFrame
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', -1)
direct="C:/Users/lewis/Documents/GEOG 569/"

u = getpass.getpass(prompt='Enter username')
p = getpass.getpass(prompt='Enter password')
gis = GIS('https://fodp.maps.arcgis.com/',u, p)

search_result = gis.content.search(query="Input Data Used for Geospatial Analysis
v2")

#The required data is a "Feature Layer Collection" so it's the second option (i.e.,
indexed at 0)
dataSource = search_result[0].layers
for lyr in dataSource:
    print(lyr.properties.name)

#Lets import the bird data
birds = dataSource[0].query().sdf

#now Lets create and display data dictionary for the bird data
birdsDictionary = birds.dtypes.to_frame('dtypes').reset_index()
birdsDictionary.columns = ['Column', 'Data Type']
column = [
birdsDictionary['Column'] == 'Count_',
birdsDictionary['Column'] == 'ID',
```



```

birdsDictionary['Column'] == 'Latitude',
birdsDictionary['Column'] == 'Longitude',
birdsDictionary['Column'] == 'Loop',
birdsDictionary['Column'] == 'Month',
birdsDictionary['Column'] == 'Month_No',
birdsDictionary['Column'] == 'OBJECTID',
birdsDictionary['Column'] == 'Precipitation',
birdsDictionary['Column'] == 'SHAPE',
birdsDictionary['Column'] == 'Site',
birdsDictionary['Column'] == 'Species',
birdsDictionary['Column'] == 'Station',
birdsDictionary['Column'] == 'Survey_Date',
birdsDictionary['Column'] == 'Weather',
birdsDictionary['Column'] == 'Year'
]
descriptions = [
'The count of birds witnessed',
'Indexed unique identifier for bird data collection event',
'Latitude coordinate (feet) where collection occurred',
'Longitude coordinate (feet) where collection occurred',
'Park specific descriptive location',
'Descriptive month',
'Month of event in integer format (1-12)',
'Default index for feature classes created by ArcGIS',
'Description of whether precipitation took place',
'Geometry object of feature class',
'Park where bird sighting took place',
'Descriptive bird species',
'Station number where bird sighting took place',
'Date when bird sighting took place (YYYY-MM-DD)',
'Description of weather on the day of bird sighting',
'Year when bird sighting took place (YYYY)'
]
birdsDictionary['Description'] = np.select(column,descriptions, default='error')

#now Lets import the polygons with our simulated scenarios to output bird counts
when we modify Land type
Scenarios = dataSource[1].query().sdf

#data dictionary for the polygon scenarios
ScenariosDictionary = Scenarios.dtypes.to_frame('dtypes').reset_index()
ScenariosDictionary.columns = ['Column','Data Type']
column = [
ScenariosDictionary['Column'] == 'Acres',
ScenariosDictionary['Column'] == 'OBJECTID',
ScenariosDictionary['Column'] == 'SHAPE',
ScenariosDictionary['Column'] == 'Scenario',

```

```

ScenariosDictionary['Column'] == 'SceneDesc',
ScenariosDictionary['Column'] == 'Shape__Area',
ScenariosDictionary['Column'] == 'Shape__Length'
]
descriptions = [
'The acreage of the area simulated for change in landcover/vegetation status',
'Default index for feature classes created by ArcGIS',
'Geometry object of feature class',
'Nominal indicator (1-5) identifying simulated change in landcover/vegetation
status',
'Descriptive summary where change in landcover/vegetation status will take place',
'Area (in meters) of the polygon simulated for change in landcover/vegetation
status',
'Length of perimeter of polygon simulated for change in landcover/vegetation
status'
]
ScenariosDictionary['Description'] = np.select(column,descriptions,
default='error')

#importing the polygon hexbins
hexbins_polygons = dataSource[2].query().sdf

#data dictionary for the polygon hexbins
hexPolyDictionary = hexbins_polygons.dtypes.to_frame('dtypes').reset_index()
hexPolyDictionary.columns = ['Column','Data Type']
column = [
hexPolyDictionary['Column'] == 'Acres',
hexPolyDictionary['Column'] == 'Agriculture',
hexPolyDictionary['Column'] == 'GRID_ID',
hexPolyDictionary['Column'] == 'LC',
hexPolyDictionary['Column'] == 'OBJECTID',
hexPolyDictionary['Column'] == 'Riparian',
hexPolyDictionary['Column'] == 'SHAPE',
hexPolyDictionary['Column'] == 'Shape__Area',
hexPolyDictionary['Column'] == 'Shape__Length',
hexPolyDictionary['Column'] == 'Urban',
hexPolyDictionary['Column'] == 'zoneMGT'
]
descriptions = [
'The acreage of the hexbin',
'Indicator (0-1) whether the landcover in hexbin is agrarian',
'Indexed unique identifier for hexbin',
'Landcover type',
'Default index for feature classes created by ArcGIS',
'Indicator (0-1) whether the landcover in hexbin is riparian',
'Geometry object of feature class',
'Area (in meters) of the hexbin',

```

```

'Length of perimeter of hexbin',
'Indicator (0-1) whether the landcover in hexbin is urban',
'Description of zone area within Discover Park'
]
hexPolyDictionary['Description'] = np.select(column,descriptions, default='error')

#Importing the point data representing vegetation at
#Discovery Park aggregated to the centroids of hexbins
hexbins_pt_veg = dataSource[3].query().sdf

#data dictionary for the vegetation data
hexVegDictionary = hexbins_pt_veg.dtypes.to_frame('dtypes').reset_index()
hexVegDictionary.columns = ['Column','Data Type']
column = [
hexVegDictionary['Column'] == 'Acres',
hexVegDictionary['Column'] == 'CommonDom',
hexVegDictionary['Column'] == 'GRID_ID',
hexVegDictionary['Column'] == 'Latitude',
hexVegDictionary['Column'] == 'Longitude',
hexVegDictionary['Column'] == 'OBJECTID',
hexVegDictionary['Column'] == 'SHAPE'
]
descriptions = [
'The acreage of the hexbin',
'Common name of the dominant vegetation species within hexbin',
'Indexed unique identifier for hexbin',
'Latitude coordinate (feet) for hexbin centroid',
'Longitude coordinate (feet) for hexbin centroid',
'Default index for feature classes created by ArcGIS',
'Geometry object of feature class'
]
hexVegDictionary['Description'] = np.select(column,descriptions, default='error')

#Importing the point data representing Landcover at
#Discovery Park aggregated to the centroids of hexbins
key = dataSource[4].query().sdf

#data dictionary for the polygon hexbins
keyDictionary = key.dtypes.to_frame('dtypes').reset_index()
keyDictionary.columns = ['Column','Data Type']
column = [
keyDictionary['Column'] == 'Acres',
keyDictionary['Column'] == 'Agriculture',
keyDictionary['Column'] == 'GRID_ID',
keyDictionary['Column'] == 'LC',
keyDictionary['Column'] == 'Latitude',

```

```

keyDictionary['Column'] == 'Longitude',
keyDictionary['Column'] == 'OBJECTID',
keyDictionary['Column'] == 'Riparian',
keyDictionary['Column'] == 'SHAPE',
keyDictionary['Column'] == 'Urban',
keyDictionary['Column'] == 'zoneMGT'
]
descriptions = [
'The acreage of the hexbin',
'Indicator (0-1) whether the landcover in hexbin is agrarian',
'Indexed unique identifier for hexbin',
'Landcover type',
'Latitude coordinate (feet) for hexbin centroid',
'Longitude coordinate (feet) for hexbin centroid',
'Default index for feature classes created by ArcGIS',
'Indicator (0-1) whether the landcover in hexbin is riparian',
'Geometry object of feature class',
'Indicator (0-1) whether the landcover in hexbin is urban',
'Description of zone area within Discover Park'
]
keyDictionary['Description'] = np.select(column,descriptions, default='error')

search_result = gis.content.search(query="ScenariosFeature_pts")
search_result
#The required data is a "Feature Layer Collection" so it's the only option (i.e., indexed at 0)
dataSource = search_result[0].layers
for lyr in dataSource:
    print(lyr.properties.name)

#Importing the point data representing landcover at Discovery Park aggregated to the centroids of hexbins
scene = dataSource[0].query().sdf

#data dictionary for the polygon hexbins
sceneDictionary = scene.dtypes.to_frame('dtypes').reset_index()
sceneDictionary.columns = ['Column','Data Type']
column = [
sceneDictionary['Column'] == 'Acres',
sceneDictionary['Column'] == 'Agricultur',
sceneDictionary['Column'] == 'FID',
sceneDictionary['Column'] == 'GRID_ID',
sceneDictionary['Column'] == 'LC',
sceneDictionary['Column'] == 'Latitude',
sceneDictionary['Column'] == 'Longitude',
sceneDictionary['Column'] == 'Riparian',
sceneDictionary['Column'] == 'SHAPE',

```

```

sceneDictionary['Column'] == 'Scenario',
sceneDictionary['Column'] == 'Urban',
sceneDictionary['Column'] == 'modified',
sceneDictionary['Column'] == 'zoneMGT'
]
descriptions = [
'The acreage of the hexbin',
'Indicator (0-1) whether the landcover in hexbin is agrarian',
'Feature class index created by ArcGIS',
'Indexed unique identifier for hexbin',
'Landcover type',
'Latitude coordinate (feet) for hexbin centroid',
'Longitude coordinate (feet) for hexbin centroid',
'Indicator (0-1) whether the landcover in hexbin is riparian',
'Geometry object of feature class',
'Identifier (0-5) of the scenario for landcover manipulations',
'Indicator (0-1) whether the landcover in hexbin is urban',
'Indicator (0-1) whether the landcover in hexbin is different from original',
'Description of zone area within Discover Park'
]
sceneDictionary['Description'] = np.select(column,descriptions, default='error')

#pandas will not allow cross joins on columns
#from different dataframes with same name
key2 = key[['GRID_ID','Latitude','Longitude']]
birds2 = birds[['Species','Count_','Latitude','Longitude']]
birds2['Latitude2']=birds2['Latitude']
birds2['Longitude2']=birds2['Longitude']
del birds2['Latitude']
del birds2['Longitude']

#Lets do a cross join between the birds and hexbins,
#but only keep the rows where bird sightings
#are within 700 feet of a hexbin
test=key2.assign(foo=1).merge(birds2.assign(foo=1)).drop('foo', 1)
test['dist'] = np.sqrt(((test['Latitude']-
test['Latitude2']).pow(2))+((test['Longitude']-test['Longitude2']).pow(2)))
test = test[test['dist']<=700]
test.reset_index(inplace=True)
del test['index']

#Sum bird counts within each species at each GRID_ID
birdSum = test.groupby(['GRID_ID','Species'])['Count_'].sum().reset_index()

#now Lets pivot the bird Species into separate columns
birdPivot =
pd.pivot_table(birdSum,index=['GRID_ID'],columns='Species',values='Count_')

```

```

birdPivot.reset_index(inplace=True)

#Lets fill the nulls with zeros
birdPivot.fillna(0, inplace=True)

#Now Lets pivot the vegetation across columns
hexbins_pt_veg['val']=1
veggiePivot =
pd.pivot_table(hexbins_pt_veg,index=['GRID_ID'],columns='CommonDom',values='val')
veggiePivot.fillna(0, inplace=True)
#now Let's left join the birds pivot on our key table
almost = pd.merge(key[['GRID_ID','LC','Latitude','Longitude']],birdPivot,
how='left',
                    left_on='GRID_ID', right_on='GRID_ID')
#now Let's left join our vegetation to finalize our dataframe that we will model
data = pd.merge(almost,veggiePivot, how='left',
                left_on='GRID_ID', right_on='GRID_ID')
data.rename(columns={'Null':'Barren'}, inplace=True)

data2=data[['American Robin','Big-leaf Maple','Bitter Cherry','California
Blackberry',
            'Cattail','Douglas-fir','Himalayan Blackberry','Barren','Pacific
Madrone',
            'Quackgrass','Red Alder','Scotch Broom','Skunk Cabbage','Tall Fescue',
            'Western Hemlock','Western Red Cedar','Willow']]
null_data = data2[data2.isnull().any(axis=1)]
null_data
good_data = data2[~data2.isnull().any(axis=1)]
good_data = good_data.reset_index()

X_train, X_test, Y_train, Y_test = train_test_split(good_data[['Big-leaf
Maple','Bitter Cherry','California Blackberry',
                                                             'Cattail','Douglas-
fir','Himalayan Blackberry','Barren','Pacific Madrone',
                                                             'Quackgrass','Red
Alder','Scotch Broom','Skunk Cabbage','Tall Fescue',
                                                             'Western
Hemlock','Western Red Cedar','Willow']],
                                                    good_data[['American Robin']],
                                                    test_size=0,
                                                    random_state=226)

gsc = GridSearchCV(estimator=RandomForestRegressor(),
                    param_grid={'max_depth': range(3,30),
                                'n_estimators': (10, 50, 100, 1000)},
                    cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=1)
grid_result = gsc.fit(X_train, Y_train.values.ravel())

```

```

best_params = grid_result.best_params_
rfr = RandomForestRegressor(max_depth=best_params["max_depth"],
n_estimators=best_params["n_estimators"],
                           random_state=False, verbose=False)
# Perform K-Fold CV
scores = cross_val_score(rfr, X_train, Y_train.values.ravel(), cv=10,
scoring='neg_mean_absolute_error')
rfr.fit(X_train, Y_train.values.ravel())
scores

features = X_train.columns.values
importances = rfr.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances for The American Robin')
plt.barh(range(len(indices)), importances[indices], color='#8f63f4',
align='center')
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
plt.show()

predict = pd.merge(scene[['GRID_ID', 'Scenario', 'modified']],
                   data[['GRID_ID', 'Big-leaf Maple', 'Bitter Cherry', 'California
Blackberry',
                           'Cattail', 'Douglas-fir', 'Himalayan
Blackberry', 'Barren', 'Pacific Madrone',
                           'Quackgrass', 'Red Alder', 'Scotch Broom', 'Skunk
Cabbage', 'Tall Fescue',
                           'Western Hemlock', 'Western Red Cedar', 'Willow']],
                   how='left', left_on='GRID_ID', right_on='GRID_ID')

predict.loc[predict['modified'] == 1, 'Big-leaf Maple'] = 0
predict.loc[predict['modified'] == 1, 'Bitter Cherry'] = 0
predict.loc[predict['modified'] == 1, 'California Blackberry'] = 0
predict.loc[predict['modified'] == 1, 'Cattail'] = 0
predict.loc[predict['modified'] == 1, 'Douglas-fir'] = 0
predict.loc[predict['modified'] == 1, 'Himalayan Blackberry'] = 0
predict.loc[predict['modified'] == 1, 'Barren'] = 1
predict.loc[predict['modified'] == 1, 'Pacific Madrone'] = 0
predict.loc[predict['modified'] == 1, 'Quackgrass'] = 0
predict.loc[predict['modified'] == 1, 'Red Alder'] = 0
predict.loc[predict['modified'] == 1, 'Scotch Broom'] = 0
predict.loc[predict['modified'] == 1, 'Skunk Cabbage'] = 0
predict.loc[predict['modified'] == 1, 'Tall Fescue'] = 0
predict.loc[predict['modified'] == 1, 'Western Hemlock'] = 0
predict.loc[predict['modified'] == 1, 'Western Red Cedar'] = 0
predict.loc[predict['modified'] == 1, 'Willow'] = 0

```

```

scenario1=predict.loc[(predict['Scenario'].isin([0,1]))&(predict['modified']==1)]
scenario2=predict.loc[(predict['Scenario'].isin([0,2]))&(predict['modified']==1)]
scenario3=predict.loc[(predict['Scenario'].isin([0,3]))&(predict['modified']==1)]
scenario4=predict.loc[(predict['Scenario'].isin([0,4]))&(predict['modified']==1)]
scenario5=predict.loc[(predict['Scenario'].isin([0,5]))&(predict['modified']==1)]

scenario1['pred_AmericanRobin'] = rfr.predict(scenario1[['Big-leaf Maple','Bitter
Cherry','California Blackberry',
                'Cattail','Douglas-fir','Himalayan
Blackberry','Barren','Pacific Madrone',
                'Quackgrass','Red Alder','Scotch Broom','Skunk
Cabbage','Tall Fescue',
                'Western Hemlock','Western Red Cedar','Willow']])
scenario2['pred_AmericanRobin'] = rfr.predict(scenario2[['Big-leaf Maple','Bitter
Cherry','California Blackberry',
                'Cattail','Douglas-fir','Himalayan
Blackberry','Barren','Pacific Madrone',
                'Quackgrass','Red Alder','Scotch Broom','Skunk
Cabbage','Tall Fescue',
                'Western Hemlock','Western Red Cedar','Willow']])
scenario3['pred_AmericanRobin'] = rfr.predict(scenario3[['Big-leaf Maple','Bitter
Cherry','California Blackberry',
                'Cattail','Douglas-fir','Himalayan
Blackberry','Barren','Pacific Madrone',
                'Quackgrass','Red Alder','Scotch Broom','Skunk
Cabbage','Tall Fescue',
                'Western Hemlock','Western Red Cedar','Willow']])
scenario4['pred_AmericanRobin'] = rfr.predict(scenario4[['Big-leaf Maple','Bitter
Cherry','California Blackberry',
                'Cattail','Douglas-fir','Himalayan
Blackberry','Barren','Pacific Madrone',
                'Quackgrass','Red Alder','Scotch Broom','Skunk
Cabbage','Tall Fescue',
                'Western Hemlock','Western Red Cedar','Willow']])
scenario5['pred_AmericanRobin'] = rfr.predict(scenario5[['Big-leaf Maple','Bitter
Cherry','California Blackberry',
                'Cattail','Douglas-fir','Himalayan
Blackberry','Barren','Pacific Madrone',
                'Quackgrass','Red Alder','Scotch Broom','Skunk
Cabbage','Tall Fescue',
                'Western Hemlock','Western Red Cedar','Willow']])

```



Appendix 4: Data Dictionaries of the What-If Analysis

**Polygon Feature Class: Hexbin Polygons**

**Short Description:** This is a feature class consisting of roughly 49,000 hexbins covering the Discovery Park area

Column	Data Type	Description
Acres	double	The acreage of the hexbin
Agriculture	integer	Indicator (0-1) whether the landcover in hexbin is agrarian
GRID_ID	string	Indexed unique identifier for hexbin
LC	string	Landcover type
OBJECTID	integer	Default index for feature classes created by ArcGIS
Riparian	integer	Indicator (0-1) whether the landcover in hexbin is riparian
SHAPE	geometry	Geometry object of feature class
Shape__Area	double	Area (in meters) of the hexbin
Shape__Length	double	Length of perimeter of hexbin
Urban	integer	Indicator (0-1) whether the landcover in hexbin is urban
zoneMGT	string	Description of zone area within Discover Park

**Polygon Feature Class: Scenario Polygons**

**Short Description:** This feature class has polygons representing a simulated case study where areas in Discovery Park are considered for commercial use.

Column	Data Type	Description
Acres	double	The acreage of the area simulated for change in landcover/vegetation status
OBJECTID	integer	Default index for feature classes created by ArcGIS
SHAPE	geometry	Geometry object of feature class
Scenario	integer	Nominal indicator (1-5) identifying simulated change in landcover/vegetation status
SceneDesc	string	Descriptive summary where change in landcover/vegetation status will take place
Shape__Area	double	Area (in meters) of the polygon simulated for change in landcover/vegetation status
Shape__Length	double	Length of perimeter of polygon simulated for change in landcover/vegetation status

**Point Feature Class: Birds**

**Short Description:** This is feature class consists of points throughout the park where birds has been sighted in 2015 and 2016

Column	Data Type	Description
Count_	integer	The count of birds witnessed
ID	integer	Indexed unique identifier for bird data collection event
Latitude	double	Latitude coordinate (feet) where collection occurred
Longitude	double	Longitude coordinate (feet) where collection occurred
Loop	string	Park specific descriptive location
Month	string	Descriptive month
Month_No	integer	Month of event in integer format (1-12)
OBJECTID	integer	Default index for feature classes created by ArcGIS
Precipitation	string	Description of whether precipitation took place
SHAPE	geometry	Geometry object of feature class
Site	string	Park where bird siting took place
Species	string	Descriptive bird species
Station	integer	Station number where bird sighting took place
Survey_Date	string	Date when bird sighting took place (YYYY-MM-DD)
Weather	string	Description of weather on the day of bird sighting

Year	integer	Year when bird sighting took place (YYYY)
------	---------	---

**Point Feature Class: Scenarios**

**Short Description:** This is a point feature class representing centroids of the hexbins. Furthermore, landcover is modified within some hexbins to enable modeling of landcover changes.

Column	Data Type	Description
Acres	double	The acreage of the hexbin
Agriculture	integer	Indicator (0-1) whether the landcover in hexbin is agrarian
FID	integer	Feature class index created by ArcGIS
GRID_ID	string	Indexed unique identifier for hexbin
LC	string	Landcover type
Latitude	double	Latitude coordinate (feet) for hexbin centroid
Longitude	double	Longitude coordinate (feet) for hexbin centroid
Riparian	integer	Indicator (0-1) whether the landcover in hexbin is riparian
SHAPE	geometry	Geometry object of feature class
Scenario	integer	Identifier (0-5) of the scenario for landcover manipulations
Urban	integer	Indicator (0-1) whether the landcover in hexbin is urban
modified	integer	Indicator (0-1) whether the landcover in hexbin is different from original
zoneMGT	string	Description of zone area within Discover Park

**Point Feature Class: Hexbin Landcover**

**Short Description:** This is a point feature class representing centroids of the hexbins highlighting Discovery Park's landcover in its current state.

Column	Data Type	Description
Acres	double	The acreage of the hexbin
Agriculture	integer	Indicator (0-1) whether the landcover in hexbin is agrarian
GRID_ID	string	Indexed unique identifier for hexbin
LC	string	Landcover type
Latitude	double	Latitude coordinate (feet) for hexbin centroid
Longitude	double	Longitude coordinate (feet) for hexbin centroid
OBJECTID	integer	Default index for feature classes created by ArcGIS
Riparian	integer	Indicator (0-1) whether the landcover in hexbin is riparian
SHAPE	geometry	Geometry object of feature class
Urban	integer	Indicator (0-1) whether the landcover in hexbin is urban
zoneMGT	string	Description of zone area within Discover Park

**Point Feature Class: Vegetation**

**Short Description:** This is a point feature class representing centroids of the hexbins highlighting specific vegetative species at Discovery Park.

Column	Data Type	Description
Acres	double	The acreage of the hexbin
CommonDom	string	Common name of the dominant vegetation species within hexbin
GRID_ID	string	Indexed unique identifier for hexbin
Latitude	double	Latitude coordinate (feet) for hexbin centroid
Longitude	double	Longitude coordinate (feet) for hexbin centroid
OBJECTID	integer	Default index for feature classes created by ArcGIS
SHAPE	geometry	Geometry object of feature class

**Table: Earth Economics Lookup**

**Short Description:** This is a table listing monetary affordances of Discovery Park's acreage given landcover.

Column	Data Type	Description
Agriculture	integer	Indicator (0-1) whether the landcover is agrarian
Landcover	string	Landcover type
ObjectID	integer	Default index for feature classes created by ArcGIS
Riparian	integer	Indicator (0-1) whether the landcover is riparian
Urban	integer	Indicator (0-1) whether the landcover is urban

high	double	Upper limit dollar value for affordance
low	double	Lower limit dollar value for affordance
valueType	string	Affordance type (Aesthetics, Air, Carbon Sequestration,...)