AUTOHOPTS
Hyperparameter Optimization

Developed by Rudresh Nitinku, Kristen Laird, Daniel Belmes, James Michaud, and Ira Ceka
DeepCure

Using AI to discover highly effective small-molecule drug candidates

Analyzing a space of one trillion compounds and multiple hyperparameters in parallel
DeepCure trains and evaluates hundreds of machine learning models every day.

Algorithms are run by command line, making it:

- Easy to make mistakes because there is minimal validation
- Hard to get new employees up to speed
- Inaccessible to employees without a strong machine-learning background
Project Scope

Start New Experiment
Configure search space and experiment setup. Send to scheduler to start training

Monitor Experiment
Has anything broken or failed?

Analyze Results
What models performed best? What should be considered next?

Investigate
Once there’s a good drug candidate, investigate further

© DeepCure, Inc. All Rights Reserved. Confidential & Proprietary.
Goal

Minimize the overhead in problem solving and increase the speed of DeepCure’s hypothesis testing cycle by making it easier to start an experiment
Target Users

- New employees with minimal ML experience
- Employees with some ML experience
- Advanced users with significant ML and DeepCure experience
Solution

Intuitive web app that lets DeepCure employees easily start a new experiment

- **Accessible** to users regardless of level of ML knowledge, with advanced options available for more advanced users
- **Code must adapt** to DeepCure’s changing needs
Begin New Experiment

Experiment Name: 3rd cephalosporins
Path to Data: C:/MoleculeSet
Answer Column: delta_g
Worker Class: Generic
Split Type: train-test
Test Size: 1
Problem: regression
Models: classical
Flag File: optional
Model Configurations: optional

Models
Platform Flags
Search Space Resources
Problem Space Parameters
<table>
<thead>
<tr>
<th>Experiment Name:</th>
<th>Worker Class:</th>
<th>Problem:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd cephalosporins</td>
<td>Generic</td>
<td>regression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path to Data:</th>
<th>Split Type:</th>
<th>Models:</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:/MoleculeSet</td>
<td>train-test</td>
<td>classical</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Answer Column:</th>
<th>Test Size:</th>
<th>Flag File:</th>
</tr>
</thead>
<tbody>
<tr>
<td>delta_g</td>
<td>1</td>
<td>optional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Configurations:</th>
<th></th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Options:
- Models
- Platform Flags
- Search Space Resources
- Problem Space Parameters
<table>
<thead>
<tr>
<th>Experiment Name:</th>
<th>Worker Class:</th>
<th>Problem:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd cephalosporins</td>
<td>Generic</td>
<td>regression</td>
</tr>
<tr>
<td>Path to Data:</td>
<td>Split Type:</td>
<td>Models:</td>
</tr>
<tr>
<td>C:/MoleculeSet</td>
<td>train-test</td>
<td>classical</td>
</tr>
<tr>
<td>Answer Column:</td>
<td>Test Size:</td>
<td>Flag File: optional</td>
</tr>
<tr>
<td>delta_g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models:</td>
<td></td>
<td>Model Configurations: optional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>Platform Flags</th>
<th>Search Space Resources</th>
<th>Problem Space Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiment Name:</td>
<td>3rd cephalosporins</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path to Data:</td>
<td>C:/MoleculeSet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Answer Column:</td>
<td>delta_g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worker Class:</td>
<td>Generic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Split Type:</td>
<td>train-test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Size:</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem:</td>
<td>regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models:</td>
<td>classical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flag File:</td>
<td>optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Configurations:</td>
<td>optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Models

Platform Flags

Search Space Resources

Problem Space Parameters
Product Highlights

- Adaptive Interface
- Reusable Components
- Dynamically Generated Forms
- Input Validation
Adaptability

Default flags and models are maintained by DeepCure. Changes to these defaults are parsed automatically, ensuring all users start from the same baseline.

```
"n_neighbors": {
    "display_name": "Number of Neighbors",
    "lower": 3,
    "upper": 20,
    "quantization": 1,
}
```

```
"n_neighbors": {
    "display_name": "Number of Nay-bohrs",
    "lower": 10,
    "upper": 200,
    "quantization": 10,
}
```
Reusable Components

Each Model contains multiple Parameter components, and each Model is a component.

Each Flag is a separate component.

Creating a component is like using an object constructor, allowing the same code to be reused in a for loop.
Input cards are created based on the object they represent. Card type is **dynamically** chosen based on input type.
Input cards are created based on the object they represent. Card type is dynamically chosen based on input type.

```json
"C": {
  "display_name": "C",
  "type": "Float",
  "lower": 1,
  "lower_min": 0,
  "lower_max": 1000,
  "upper": 3,
  "upper_min": 50,
  "upper_max": 1000,
  "quantization": null,
},

"penalty": {
  "display_name": "Penalty",
  "type": "Categorical",
  "selected": ["..."],
  "sequence": ["..."],
  "error_count": 0
}
```

C:
- Lower: 1
- Upper: 3

Distribution: uniform
Log: false
Type: Float

Penalty:
- I1
- I2

Type: Categorical
Input cards are created based on the object they represent. Card type is **dynamically** chosen based on input type.
Input cards are created based on the object they represent. Card type is dynamically chosen based on input type.

```
"n_neighbors": {
  "display_name": "Number of Neighbors",
  "type": "Int",
  "lower": 3,
  "lower_min": 0,
  "lower_max": 50,
  "upper": 50,
  "upper_min": 50,
  "upper_max": 1000,
  "quantization": 1,
}
```

**Number of Neighbors:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lower</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>Upper</strong></td>
<td>50</td>
</tr>
<tr>
<td><strong>Quantization</strong></td>
<td>1</td>
</tr>
</tbody>
</table>

Distribution: uniform Log: false Type: Int
Input Validation

Input validation reflects values contained in the default options file.

```
"n_neighbors": {
    "display_name": "Number of Neighbors",
    "type": "Int",
    "lower": 3,
    "lower_min": 0,
    "lower_max": 50,
    "upper": 50,
    "upper_min": 50,
    "upper_max": 1000,
}
```

**Number of Neighbors:**

- **Lower**
  - Value: 51
  - Requirement: Lower must be less than 50

- **Upper**
  - Value: 49
  - Requirement: Upper must be greater than 50
Software and Tools

Programming languages used in this repository

- Vue: 63.31%
- JavaScript: 20.99%
- Python: 15.43%
- HTML: 0.27%
System Architecture

Backend
- Flask
- Database

Frontend
- Axios
- Vue

Users
Application Workflow

- **Browser**
  - Request
  - Response

- **Frontend Server**
  - HTML pages (with CSS and Vue)

- **Backend Server**
  - Get/Post
  - Queries

- **Database**
  - Tables
    - Configurations
    - Experiments
    - Results
Next Steps

- **Documentation** for project handoff

- DeepCure *integration*
  - Use real data
  - Submission, their monitoring, analyzing experiments

- **Backlog** for features for future development
What We Learned

- **Development**
  - Confidence as software engineers, optimality: testable code vs. beautiful prototype (but deadlines!); branch structure

- **Organization**
  - Regular meetings, solid notes, timeline tags for feature implementations

- **Communication**
  - Team-client communication: clarification
  - Member-member: getting stuck & falling behind; open and collaborative on implementing issues

- **Teamwork**
  - Open and collaborative on implementing issues; all the difference passionate and serious about project

- **Have fun!**
Thank you Thras, Manu, Alex, & Professor Pomplun and the VDC!
Questions?