UI Element Categorization

A predictive model to identify hand drawn images of User Interface components

Graham M. Smith
Springboard Data Science 2022
graham-macisaac.com

Project Overview

Scope, goals, and context

What’s Going On?

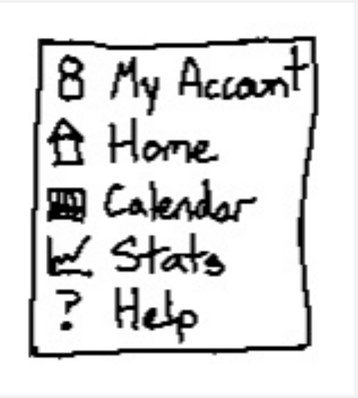
- The goal of this project is to create a model that can **predict what type of common computer interface element an image is from a hand-made drawing** (buttons, toggles, windows, etc.).

- I used the **UISketch dataset from Kaggle**. it can be found at:
  - https://www.kaggle.com/datasets/vinothpandian/uisketch

Who Is This For?

- The recent proliferation of no-code tools to allow designers to more directly contribute to software development shows there is a demand for streamlining the design → engineering pipeline

- This project's highest potential would be as part of a software tool that converted hand drawn images from **UI/UX designers** or **product managers**, and turned them directly into interactive prototypes in software like Adobe XD or Figma*



*Which I used for this presentation!

Formulation As A Data Science Problem

- The data contained 19,000 hand-drawn sketches of 21 UI element categories (such as buttons, checkboxes, menu's sliders, etc.) collected from 1,218 partiacipants and in the form of greyscale 64 × 64 pixel JPEG's

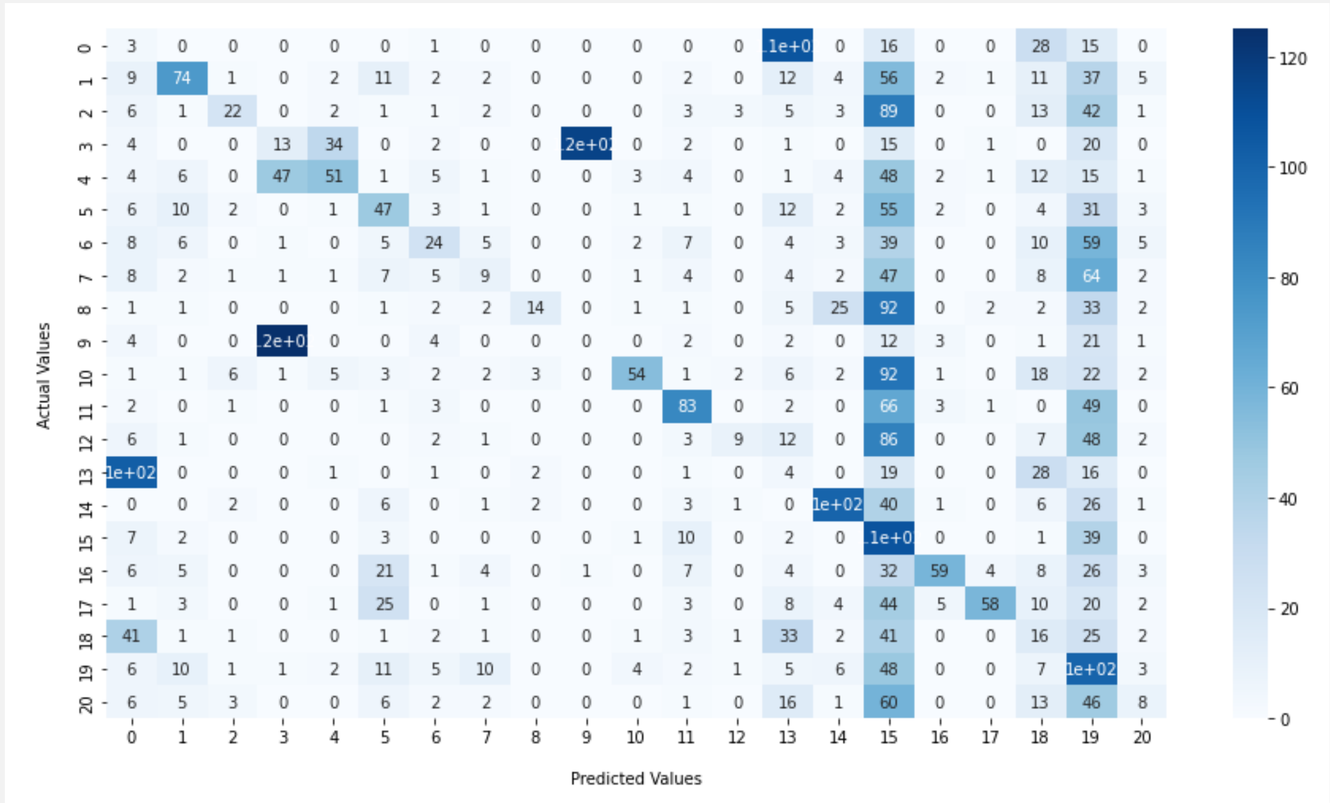
- I converted the images into a single matrix, with each row as a one dimensional 4096 length array representing each image.

	4071	4072	4073	4074	4075	4076	4077	4078	4079	4080	...	33543	33544	33545	33546	33547	33548	33549	33550	33551	label
0	255	255	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	dropdown_menu
1	255	255	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	dropdown_menu
2	255	255	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	dropdown_menu
3	255	255	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	dropdown_menu
4	255	255	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	dropdown_menu

- Sampling equally from each class, this was then broken up into a training and test matrix from which I could build models. The goal was to find the most generally accurate model at predicting what group an image belonged to.

Model 1: K-Nearest Neighbors

- Overall, 10 classes have a precision greater than 50%, but 9 have a precision less than 20%, and 3 are actually worse than random chance. A good representation of this is shown in the following confusion matrix:



Modeling Outcomes

Results

Model 1: K-Nearest Neighbors

- Simple model used as a baseline because it's fairly quick to run using a few arbitrary values of K

- This initial model has a pretty abysmal mean precision (rate of true positives) of 39.6%, however, given that there are 18 classes, this is not a good method of measuring the quality of the model. Not only would picking randomly only yield an accuracy of 5.5%, making this 4x better than chance, but the variance between classes is extremely high.

- As you can see in the figure on the right, it ranged from a reasonabl 85% accuracy for enabled switches to an abysmal 1% for alerts.

	precision	recall	f1-score	support
alert	0.01	0.02	0.01	170
button	0.58	0.32	0.41	231
card	0.55	0.11	0.19	194
checkbox_checked	0.07	0.06	0.07	208
checkbox_unchecked	0.51	0.25	0.33	206
chip	0.31	0.26	0.28	181
data_table	0.36	0.13	0.20	178
dropdown_menu	0.20	0.05	0.09	166
floating_action_button	0.67	0.08	0.14	184
grid_list	0.00	0.00	0.00	175
image	0.79	0.24	0.37	224
label	0.58	0.39	0.47	211
menu	0.53	0.05	0.09	177
radio_button_checked	0.02	0.02	0.02	175
radio_button_unchecked	0.63	0.53	0.58	189
slider	0.10	0.62	0.17	173
switch_disabled	0.76	0.33	0.46	181
switch_enabled	0.85	0.31	0.46	185
text_area	0.08	0.09	0.09	171
text_field	0.13	0.45	0.20	222
tooltip	0.19	0.05	0.08	169
accuracy			0.22	3970
macro avg	0.38	0.21	0.22	3970
weighted avg	0.39	0.22	0.23	3970

Model 2: XGBoost

- The second model I attempted was XGBoost, because it has ridge regression built in and I figured with so many parameters some would probably be reduced to 0. It is also faster than multiple logistic regression by itself.

- Even when removing all the whitespace and eliminating pixels that were identical for all images, there was still too much data for my computer to handle, so I removed 70% of the images (sampling proportionally from each class)

- This initial XGBoost model has an overall accuracy score of **42%.** This is only slightly better than the ~40% we were getting with KNN in the preprocessing step.

- This model still took 12 hours to run, so I decided against trying to tune the parameters and instead moved on to my 3rd model.

Model 3: Light GBM

- Due to the training time issues, I decided to use a model with lower memory usage
- It's also high accuracy, since It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy.
- It had a much better testing accuracy than either of the other models, with an overall testing accuracy of 63%.

Conclusion

What did we learn

Learnings

- While I’m going to leave the project here for the sake of expediancy, I will note that if I had more time a Convolutional Neural Network would almost certainly be more accurate than the models I've used here. Sadly 63% is still a far cry from good enough to replace a human at the task of identifying the images.