ML For Everyone

By Davis Wojnovich and Len Huang

Project Goal:
Create a user friendly web interface that allows users to generate ML models of uploaded CSV files.

Main Components:

- **Display Stuff:** *Front End (Javascript Focus)*
  - Charts.js to visualize graphs
  - React.js framework
- **Compute Stuff:** *Back End (API To Render Data)*
  - Matplotlib
  - Compute Models
    - Regression
    - Clustering (K-Means)
    - Neural Network
  - Flask/Django
  - Handle Requests for Data / Graphs
  - https://www.pythonanywhere.com/(hosting maybe)
  - Likely AWS/Google Cloud
  - Send Graph "component" {xaxis, yaxis, data} + stats to front end
- **Store Stuff:** *Firebase*
  - Store User Data
  - Accounts / OAuth

Side Ideas:

- File compatibility (auto-convert .xlsx -> .csv)

Ideal Timeline:
May 13th: Successfully generate data for a dummy CSV
May 14th: Host above computations on Flask/Django
May 15th: Host above computations on Flask/Django
May 16th: Make site that connects to above, renders graphs
May 17th: Integrate front-end with firebase / users

May 2020 ©
Progress log for Davis Wojnovich and Len Huang.

May 13th

Goals:

- https://www.kaggle.com/geomack/spotifyclassification
- https://www.kaggle.com/leonardopena/top50spotify2019
- https://www.kaggle.com/somesnm/partynyc
- Make a model to find correlation between Song features and some number of artists (ie this combination of loudness, pitch, key, etc, is most likely to be Lil Pump, Trippie Redd, Tekashi 6ix9ine, and Kanye West.

Things Completed:

- (Len) Created a python jupyter notebook with classification models for the Spotify Top 50 dataset with six different approaches. This will be the bulk of our code for processing the information.

Necessary Libraries:

- Scikit-learn
- Seaborn
- Pandas
- Matplotlib
- (PyLab?)

Resources Used:

- Fruit Example: https://towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lovers-edition-d20ab6b071d2
- Train/Test Splitting: https://www.bitdegree.org/learn/train-test-split
- Preprocessing (Scaling): https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9
- Making Single Predictions: https://machinelearningmastery.com/make-predictions-scikit-learn/

May 14th:

Goals:

- Convert above code to a regular python file
- Alter inputs and parameters to allow for increased generality (ie allowing users to pick columns)
- Develop an API to host these computations:
  https://www.youtube.com/watch?v=s_ht4AKnWZg
  https://towardsdatascience.com/flask-an-easy-access-door-to-api-development-2147ae694ceb
- **DAVIS TODO:**
  1. Write code to run all models and pickle them locally
     - Generality is king
     - Have inputs be like ints to specify columns
  2. Load the pickled models and run predictions
  3. Load the pickled models and get visualizations
  4. Load the pickled models and get accuracies
  5. https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/
  6. https://stackoverflow.com/questions/39173992/drop-all-data-in-a-pandas-dataframe
- **LEN TODO:**
  1. Move pythonanywhere to lendevelops@gmail.com
  2. Set up Firebase to allow for python uploads of CSV's and retrieval

Things Completed:

- (Len) Made a simple API hosted on the following links to store JSON data that can be used as python dictionaries.
  http://eliteorigin.pythonanywhere.com/visual
  http://eliteorigin.pythonanywhere.com/predict
  http://eliteorigin.pythonanywhere.com/
- (Len) Found a basic template for a multi-page react app, is put under "Frontend"
- (Davis) Wrote a function that can take an arbitrary csv and specific cols to build a model. I used strings, because to specify columns because it makes more sense, but this can be changed if need be. Wrote debugging code to test the pickling.

- (Davis) Wrote a function that can load an arbitrary locally stored pickled model, and return a prediction, given X variables.
- (Len) Followed tutorial series to make React/Redux/Firebase web app to store "projects". Will expands this hopefully to our application.
  https://www.youtube.com/watch?v=4uzEUATtNHQ&list=PL4cUxeGkcC9ij8CfkAY2RAGb-tmkNwQHG&index=29 (axios for API)
  https://www.youtube.com/watch?v=1w-oQ-i1XB8 (redux)
  https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/ (more redux)
  https://www.youtube.com/watch?v=hOQ7x_X2gIg&list=PL4cUxeGkcC9iWstfXntcj8f-dFZ4UtlN3&index=11 (TheNetNinja)
  https://www.youtube.com/watch?v=PWadEeOuv5o (Material UI)
  https://www.youtube.com/watch?v=zT62eVxShsY (multstageform)
- (Davis) learned seaborn to build the graphs on the api and implemented a handful of the most useful graphs in a "get visual" function
- (Davis) Added dope general one hot encoding function (its dope trust me)
- (Davis) Wrote a function to generate a list of descriptive statistics that builds a nice dictionary of all the all the requested descriptive statistics.

Ideas For Later + Concerns:

- Typically in statistics, if two variables are too closely correlated, you throw out one, because closely correlated variables can compromise the model (Depending on what kind of model). As a later feature, we might want to create api calls that check for correlation between X variables to try to help the users pick which variables to use.
- Measuring Accuracy is contingent on remembering the specific train/test split we used. So we cannot store the model and then reload it in isolation to calculate accuracy. I think that Accuracy might have to be calculated at model inception.
- Visualizations cannot be created from models. We need to build them from the data. Tomorrow I'll dig in to building some functions that can create graphs images, given a csv, and some specifications like cols

May 15th:

Recap:

- Damn we made a lot of progress
- (Davis) Pretty much wrote all of machine learning
- (Len) Pretty much wrote all of website

Discussion Points:

- Stats
  - One Hot Encoding
  - Graph Sizing
  - Send pictures / data for javascript charts #illiano
  - Data dumping vs walkthrough
  - Image route, upload to storage, API sends a reference storage then front end references the storage / links
  - Send data route, figure out the arguments for Charts.js, use python to calculate those, send it as a dictionary. return jsonify({rows : 2, color : plaid })
- Website
  - Private Projects
  - “Publication” / Legit Papers maybe?
  - Front page to see other works
  - Your own page to see yours
  - “Projects” will display all the visualizations / game
  - “Create new” will be a (for now) one page upload
- API / Backend
  - From front end, upload csv
  - CSV uploads to Firebase storage
  - onComplete, make call to API, pass in storage location
  - Inside API, pd.read_csv() references storage location

Goals for Tomorrow:

- (Davis)
  - Upload a CSV in firebase and get to firebase storage
  - Draw out EDA Stats process
- (Len)
  - Get calls to fb storage via python SDK in the API
  - Find way to just show your own projects (text)

May 16th:

Things Completed:

- (Len) Completed a notebook that successfully integrates basic SciKit learn functions with Cloud Storage. Will need to combine this with Pythonanywhere API.
- (Len) Added a new page to allow users to see only their own projects. Some things of concern include a necessity for pagination on the front page, making the notifications bar more like a news ticker, and finding a better way to return a specific user's projects instead of just filtering the entire list of projects bc that will grow to be slow and inefficient.

Goals for Tomorrow:

- (Davis) Upload CSV and find good ways to render your initial "display" / "exploratory" info.
- (Len) Compile pyfb notebook into a python file and get a rough prototype of pythonanywhere going. Inside here, I need to update user information (blobs) as well as adding a csv field in the firestore.

Workflow for the Future: MWF 4-6pm

May 17th:

Things Completed:

- (Len) Successfully got predictions from local hosting of FlaskAPI. Tried transferring to pythonanywhere, but had some issues, panicked, and deleted important stuff. Emailed the support team so we'll see what they can do. There are version issues with pythonanywhere and downloading certain modules so we'll need to figure that out.
- (Len) Also did hella housekeeping with the github. Ideally we stick to having one giant .gitignore and then Readme's in every relevant folder.
- (Len) Discovered https://material-ui.com/ and want to use.

Discussion Details:

- More of a break day.

Goals:

- Mess around on our own and reconvene on Tuesday May 19th.

May 18th:

Lol so much for meeting on tuesday
Things Completed:

- (Len) Significant front-end redesign, added auto-generated rows for cards. Added sliders for existing projects. Added "variables" parameter for firestore. Redesigned view project details page. Redeployed.
- (Davis) Upgraded to Cloud Run and successfully deployed a functional API!!!!!!!! LETS GOOOO WOOOO GOD SAVE THE QUEEN

Discussion Details:

- Spend money on Cloud Run to host python API
- Create Project Page is Davis' Baby
- Project details is Len's Baby
- Descriptive Statistics for IQR for Sliders
- Maintain "models", "variables", and "csvName" parameters for fireStore
- Find way to display / render CSV on page

Goals for **Thursday 5/21:**

- (Davis) Baby MVP Submit Page for CSV
- (Len) Add the modeling functions to the API
- (Both) Descriptive statistics for sliders

May 19th:

Len's Musings:

- One thing we could do as a post-mvp side project is to just develop a general use REST API for people to access models. Except in this case it'd just be models / sklearn and not anything with Firebase, so like our notebooks. Although this may be pointless because that's the point of sklearn.
- Project mutability? I wonder if we should allow users some way to edit the projects, as in change the variables used, and have the models change immediately. This may be difficult to accomplish technically and we would have to reorganize things to allow for this UI.
- There are very real applications for this in both high schools and CMU Data Science Club
- Have way for people to upload CSV from Kaggle without having to download them locally

Things Completed:

- (Davis) Implemented csv upload. Adds csv name to project and puts csv in the right storage location.

Note:

- After the csv upload update. There is a new version of fbConfig.js that needs to be grabbed before you can run the csv upload version.
- Firebase has options for upload progress bars, but at the moment, the upload is handled on form submit, so it is impossible to show the progressbar. **Though I do have ideas for this**
- Add csv verification

May 20th:

Discussion Points

- (Render PDF): https://bvaughn.github.io/react-virtualized/#/components/Table
- Fix Default Value Parameter
- API for uploading pickles
- FUCKIN CORSSSSSSSSSS wHYYYYYYYY https://stackoverflow.com/questions/55775832/google-cloud-run-how-to-set-access-control-allow-origin-header