Team 12 - Turnup
Cosi Jackson, Zhaojian Li, Yi Man, David Persico, Daniel Trahan, Connor Hudson

Project Reflection

Communication - We used slack for communicating. It was downloaded onto our phones so anytime someone needed to communicate with the group we would all receive notifications. Communication, however, still persisted to be a problem in our group. We also used Trello for agile methods and creating a sprint planning board, a retrospectives board, and project planning board. We also included checklists on trello to better visualize all of the work we had completed and work that need to be completed. Trello was very useful and helped us to organize the progress of our project and we used it successfully.

Vision Statement - Our goal was to make a web based app which would successfully generate playlists of songs that flowed well together. With new music being released everyday, it is nearly impossible to organize your music into perfect playlists for specific occasions, such as: parties, the gym, relaxing at home, doing homework, etc. With turnup, we can generate playlists for you based on music that flows well together.

Machine learning trial and error - We wanted to use a machine learning tool to do our project, so we initially chose to try and use Tensor Flow. We spent more than half of the project simply trying to understand a neural network and then trying to figure out how it could interpret music, since it is based on learning through the visualizations and the visual cortex. Another issue resided in the lack of sizable datasets for Tensor Flow, which would be both a large time sink and financially unrealistic reality. Because of the amount of time we had on this project, we decided to switch to Vowpal Wabbit, a much more basic machine learning tool that Connor had been introduced to in his machine learning class. We were able to successfully get Vowpal Wabbit to interpret the data AND communicate with the back end. This was a huge success for us. Vowpal Wabbit interpreted the music with the data we provided which was beats per minute, key, and major/minor.

Tensor Flow is written in Python, so we first tried to learn TF before we decided to use it. When we actually got to learning about TF, we found out that it is pretty much based on the visualization, so we met our first problem: how to visualize the music and do we need to visualize the entire song or just some parts of the song. And also TF needs a lot of data to get itself to learn, but we don't have such big data. The algorithm of using TF to find the similarity of two songs is a lot of work and time, and we used a lot of time to figure it out but the result didn't seem very well. That's why we changed to Vowpal Wabbit, that's a wise choice for us, and it went well.

Although Tensor Flow was unsuccessful for our project, we were still able to learn a lot about it through all of the resources and tutorials Google provides. However, there were some

communication problems from the machine learning team that could have contributed to more progress in using TF.

Front end / Back end stuff - Connor and Daniel had gotten the front and back end communicating through AJAX requests to the back end API, which was a huge success. More work still needed to be done in connecting the site's frontend to backend, but initial work on this was proving to be fruitful. Our biggest setback was using Tensor Flow, because we couldn't keep progressing without figuring out how it worked. On the front end, a simple use of javascript, html, and css worked well on constructing the site to be user friendly. There was contemplation on formatting the site using one of the many javascript frameworks that allow for easier asynchronous action throughout the site, but was ultimately let go in the interest of time. On the backend side, node JS and restify worked really well for making the API, as it handled most of the HTTP details in the framework. Unit and code style testing with Travis CI was really nice, it integrated with GitHub and handled deployment to Heroku seamlessly. Azure/mySQL didn't work too well, as the SQL database had a limited number of concurrent connections, causing the API to fail to connect at unpredictable times (decreasing stability of the backend). Vowpal Wabbit was a bit tricky to connect to the server, and we ended up having a somewhat insecure connection to the VW server for music predictions. Given more time to work on infrastructure, I think it could easily be made more secure and stable.

We also used GitHub for version control which contains all of our milestones and progress made within our project.

Project Report

Status - based on goals and objectives - We were able to create a working website. The machine learning aspect of our project delayed a lot of our potential progress and additional features we could have added to the website. We have Vowpal Wabbit nearly working and if we were to populate our database with more songs, we would be able to test out our product and train the machine learning algorithm to work the way we hope to.

Outstanding Issues - Communication from machine learning group, how can the TensorFlow neural network visualize music and learn from it? We did not follow the risk mitigation plan that we originally came up with either: Make a full outline of process that thinks of all possible failures. Replace complexities with simpler algorithms to ensure project's progression.
The machine learning TF leader did not fully communicate or update the team and the other machine learning group members on the problems and progressions made with TF and things needed to be worked on.

Accomplishments - In the end we managed to create a relatively user friendly website. The website was mostly functional, playing requested playlists easily. We managed to get Vowpal Wabbit working in dispensing the next song in a playlists, allowing for dynamic playlists. The sites backend was also up and running, which had api calls that could dispense the information.

Plan to reach in future - In the future, we would like to expand our library of songs to better cater to our users' needs. We also will need to implement some form of obfuscation or DRM to prevent the song files from being directly downloaded. We would also implement a playlist search feature, a user feedback system, and tailored dynamic playlists as utilities in the far future. We would also like to improve the user experience to encourage our users to stick with us (i.e. adding a position scrubber for the music).