Patters in Popular Songs - Dev Guide

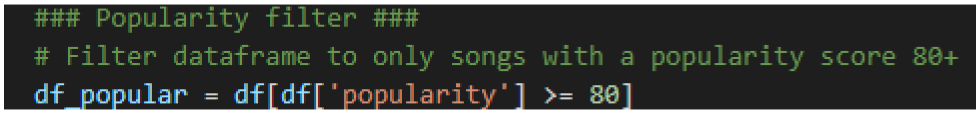
Davis Asano

Overview

This application deploys via Streamlit which launches a GUI in the user's browser. Data was initially analyzed using Jupyter Notebook (the notebook can be found in the old files file under cleaned-notebook.ipynb). Using this notebook all significant correlations (P value +/- 0.30) in the music data file were found and recorded. These correlations are then displayed as seaborn regression plots. Users may input parameters to filter the data in Streamlit to choose which correlation they would like to view.

Planning Spec

- Filters initial data set to just those songs with a popularity rating of 80+.



```
### Popularity filter ###
# Filter dataframe to only songs with a popularity score 80+
df_popular = df[df['popularity'] >= 80]
```

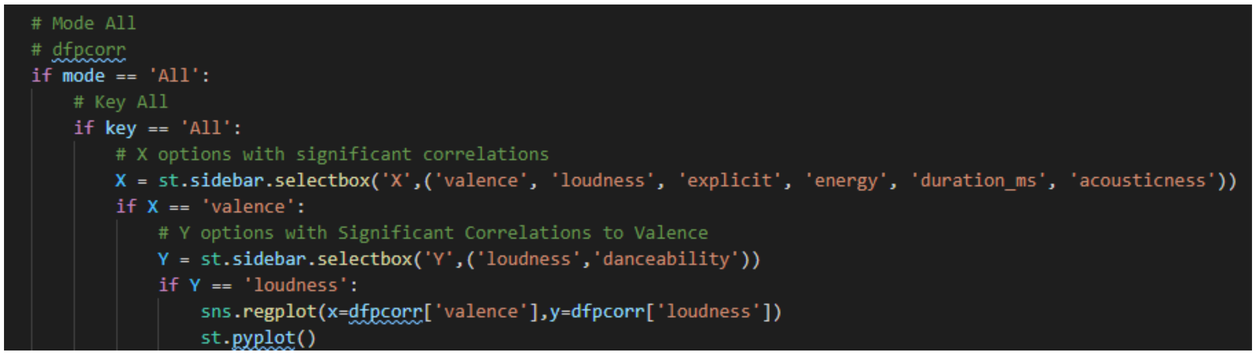
- Streamlit GUI allows users to filter data by using input parameters on the side bar.
- Side bar parameters are dependent on each other (i.e X determines what Y's available).
- Parameters are restricted to those with P values deemed significant (P = +/- 0.3).
- Displays a regression plot for each correlation.

User Flow

1) Installation – User installs Streamlit and Seaborn :
    a. Pip install streamlit
    b. Pip install seaborn

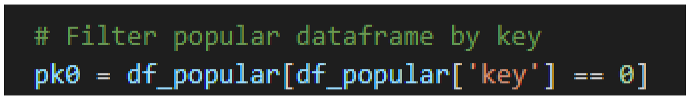
2) Launch – User uses the command prompt to launch application:
    a. Streamlit run main.py

3) User Input – The user uses the side bar to select filters. Mode, Key, X, and Y.
   a. Mode and key determine what X options are supplied
   b. X determines what Y options are supplied

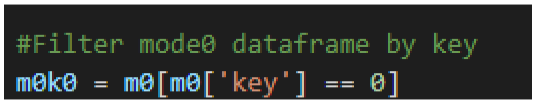


```
# Mode All
# dfpcorr
if mode == 'All':
    # Key All
    if key == 'All':
        # X options with significant correlations
        X = st.sidebar.selectbox('X',('valence', 'loudness', 'explicit', 'energy', 'duration_ms', 'acousticness'))
        if X == 'valence':
            # Y options with Significant Correlations to Valence
            Y = st.sidebar.selectbox('Y',('loudness','danceability'))
            if Y == 'loudness':
                sns.regplot(x=dfpcorr['valence'],y=dfpcorr['loudness'])
                st.pyplot()
```

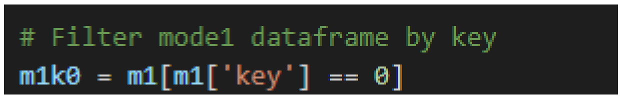
4) Graphical Representation – Seaborn plots a regression plot based on the user input and automatically displays it.
   a. Seaborn is supplied a dataframe filtered by user input.



```
# Filter popular dataframe by key
pk0 = df_popular[df_popular['key'] == 0]
```

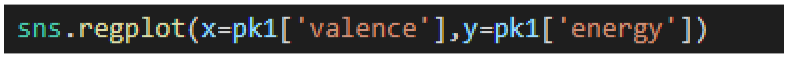


```
#Filter mode0 dataframe by key
m0k0 = m0[m0['key'] == 0]
```



```
# Filter mode1 dataframe by key
m1k0 = m1[m1['key'] == 0]
```

   b. Seaborn then runs a regplot based on the X and Y.



```
sns.regplot(x=pk1['valence'],y=pk1['energy'])
```

Known Issues

Two minor issues:

1) The P level is hard coded making it extremely tedious to adjust and doesn't allow user input.
2) The drop downs with only one option available require a empty string as a second option, this seems to be a streamlit issue. Maybe drop downs aren't the best choice for streamlit.

Future Work

Future work should rework how the correlation threshold is supplied. Creating a dataframe that stores the correlation data and can filter by user input would be ideal. As .3 is a low correlation and .5 is considered high and reliable. Further, figuring out a way to use drop downs without having a empty option for those with only one choice. It may be best to switch from drop downs to check boxes. Lastly, since the page auto refreshes to display the new options available based on their dependency to the previous user input it would be ideal to implement a “Run” button. This may require using a different package than Streamlit as I read that having dependencies in variables does not allow for such a button.