INFO 2950 Final Project - Phase IV

Janice Shen (js3678), Khai Xin Kuan (kk996), Sandy Lin (sl2534), David Park (yp358)

Table of Content

Introduction

Background and Content

A common interest in both music and finance led the four of us to explore a unique intersection between these fields. Specifically, we asked: could individual music preferences be related to broader economic trends in the United States? Inspired by prior research that suggested promising correlations between music sentiment and market performance (e.g., weekly equity returns), we decided to investigate further. With this foundation, we posed the question: how do shifts in macroeconomic and sociopolitical factors manifest in the valence and danceability of popular music? Our study focuses on two key song attributes:

1. **Valence**: A metric ranging from 0 to 1, representing a song's emotional tone, with higher values indicating more positivity.
2. **Danceability**: A measure from 0 to 1, reflecting how suitable a track is for dancing, based on rhythm, tempo, and overall energy.

We collected yearly data for popular music scraped from Billboard, matched with Spotify danceability and valence data, and paired it with economic indicators, including US GDP, S&P 500 stock returns, unemployment rate, death rate, federal funds rate, recession, birth/death rate, incarceration rate and election status. Using statistical methods such as linear and multivariate regressions, we sought to uncover relationships between these variables and the evolution of music sentiment.

Research Questions:

1. **Can the valence of the general population's music preference be predicted by economics and sociopolitical indicators including US GDP, birth rate, death rate, and federal fund rate?**
2. **Can we predict the danceability of the general population's music preference based on key economic and socioeconomic variables including US unemployment rate, election status, recession, S&P 500 stock returns, and death rate?**

Summary Findings

For **valence**, our results showed that US GDP had a statistically significant negative relationship, suggesting that societal prosperity may lead to a preference for less upbeat or more introspective music. In contrast, death rate and federal funds rate exhibited positive correlations with valence, indicating that societal hardships or financial policy changes might drive a preference for uplifting music.

For **danceability**, our analysis highlighted a negative correlation with death rate, implying that economic and societal stressors may influence the consumption of less energetic and rhythmic music. Although variables like election cycles, S&P500 stock returns, unemployment and recessions were not significant individually, the overall regression model showed us there is some relationship between music and social conditions that would require more complex models and research outside the scope of this class.

However, several limitations should be noted. First, our reliance on yearly aggregated data significantly restricted our ability to capture short-term or delayed effects of economic and social changes on music trends. Second, using Spotify and Billboard data introduces biases by focusing on mainstream preferences, potentially overlooking the diversity of music consumption across different demographics and platforms. Lastly, while valence and danceability are useful proxies, they oversimplify the complexity of human emotions, limiting our understanding of how macroeconomic factors shape musical expression. These limitations underscore the need for further research to expand on our findings and explore the intricate connections between societal conditions and music trends.

Data Description and Cleaning

Part 1: Data Description

What are the observations (rows) and the attributes (columns)?

Observations: Each row represents a song or an economic indicator for a specific time period. In the music dataset (musicdata), rows represent individual songs with the corresponding sentiment values (danceability and valence) in a given year, while in the economic and sociopolitical datasets (such as undata, econdata, us_gdp_data, etc), each row is the value of that particular year.

Attributes: In the **music dataset**, the columns include variables like **valence** (a measure of positivity in music), **danceability**, **year**, and other sentiment-related metrics. For the **economic datasets** columns include **unemployment rate**, **GDP**, **federal funds rate**, and **S&P 500 Stock Returns**. For the **socio-economic datasets** columns include **incarceration rate**, **recession**, **election cycle**, **birth rates**, and **death rates**.

Attribute Definitions:

Year: The year that corresponds to rest of the following variables present.

Music sentiments: a dimension of how one feels about the given music.

- danceability: Describes how suitable a track is for dancing, based on a combination of musical elements, including tempo, rhythm stability, beat strength, and overall regularity. For example, a fast-paced pop song with consistent beats and rhythmic flow will likely score high on danceability, whereas a slow classical piece might score lower.
  - Scale of 0.00 to 1.00
- valence: A Spotify-specific metric used to gauge how musically positive a track sounds. A higher valence would indicate a more positive track, and it is computed based on a variety of musical elements, including melody, harmony, tempo, and rhythm (Spotify Community).
  - Also on a scale of 0.00 to 1.00

Economic datasets:

- Unemployment Rate: The percentage of people in the labor force who are unemployed and actively seeking work.
  - It is calculated as: $(UnemployedIndividuals/TotalLaborForce) \times 100$
  - Reflects the overall health of the economy. Higher unemployment rates often indicate economic downturns and reduced consumer spending.
- GDP: Gross Domestic Product.

    - The total monetary or market value of all finished goods and services produced within a country in a specific time period (e.g., annually).
        - Components include consumption, investment, government spending, exports, and imports. GDP is used to indicate the economic productivity and standard of living of a country.
- Federal Funds Rate: Also known as the interest rate, the interest rate that banks charge each other to borrow money overnight.
    - Higher rates make borrowing more expensive, reducing spending. Lower rates do the opposite.
    - Federal Funds Rates is also a tool used by central banks (i.e. Federal Reserve in the US) to either control inflation or stimulate the economy.
- S&P500: The Standard and Poor's 500 is an index tracking the stock performance of 500 of the largest companies listed on US stock exchanges.
    - It is considered a major indicator of overall stock market performance and investor sentiment in the US.
    - Larger companies, weighted by market capitalization, have more influence on the index.

Social/Socio-economical Variables:

- Recession: A recession, typically, is a significant decline in economic activity spread across the market, lasting more than a few months. Dummy variable of whether a US recession was recorded that year, based on FRED data ("NBER Based Recession Indicators").
    - 1 if there was a recession recorded that year.
    - 0 if not.
- Election: Binary variable of which is 1 if there was an election that year since 1961.
    - 1 if an election occured that year.
    - 0 if not.
- State_prisons: The imprisonment rates of state prisons at that year.
- fertility: Fertility rate, total (births per woman).
- Death Rate: Age-adjusted death rates (deaths per 100,000) per year (age-adjusted death rate means death rate if age composition of the population didn't change from year to year).

Why was this dataset created?

The **music sentiment datasets** were created to explore the relationship between musical attributes (such as valence, danceability, and popularity) and factors like popularity or music trends over time. The **economic datasets** were generated by governmental agencies and prestigious universities like the Federal Reserve or NYU Stern to monitor key economic indicators for research, policy analysis, and public transparency. The **socio-economic datasets** where compiled by government organizations or nonprofits for the betterment of society, such as the Prison Policy Initiative aiming to "uses research, advocacy, and organizing to dismantle mass incarceration;" the National Center under the Health Statistics (NCHS) under the US Centers for Disease Control and Prevention, to help public research on health; or the World Bank Group that has different "global partnership fighting poverty worldwide through sustainable solutions."

Who funded the creation of the dataset?

The **music sentiment dataset** were compiled by independent researchers, such as Caleb Elgut, likely with no explicit funding. These data sources rely on platforms like **Spotify**, **Billboard**, and **ARIA**, which collected the original data. The **economic datasets** were funded by the U.S. government through organizations like the **Federal Reserve** and the **Federal Reserve Bank of St. Louis (FRED)**, as part of their regular efforts to report and track national economic trends. The Stocks dataset is compiled by research done from **NYC Stern** and the socioeconomic datasets are gathered by non-part for government organizations: **Nation Center for Health Statistics**, **Prison Policy Initiative**

What processes might have influenced what data was observed and recorded and what was not?

In the **music datasets**, factors like availability on Spotify, scraping techniques, and modern-day popularity metrics may have influenced the data observed. For example, older songs that were popular in the 1990s may have lower popularity today, potentially skewing the results when comparing them to historical economic data. In the **economic**

datasets, revisions in how economic indicators like GDP and unemployment are calculated, government adjustments, and data collection methodologies could have impacted the recorded data. Additionally, the time series nature of the data means there may be adjustments due to seasonal variations or changes in reporting practices. Social variables, such as birth rates and incarceration rates, were only available on a yearly basis due to the high cost of gathering the data, limiting the granularity of the dataset. Additionally, the reliance on platforms like Spotify and Billboard charts means the data may be biased towards mainstream trends and digital listening habits, excluding other platforms and niche groups.

What preprocessing was done, and how did the data come to be in the form that you are using?

- **Music datasets**: We scraped the raw data from a Billboard archive website, and the two large music sentiment datasets scraped from Spotify, provided by Calebelgut and Joakim Arvidsson. Calebelgut and Joakim Arvidsson are independent contributors who scraped Spotify sentiment data. Since these two contributors independently scraped their data from Spotify, we verified the datasets by checking for consistency across overlapping songs. For example, we ensured that songs with the same title and artist (e.g., "What Do You Mean (Acoustic)" by Justin Bieber) had identical sentiment values in both datasets.

- **Economic datasets**: The raw data provided by the **Federal Reserve** and **NYU Stern** was merged and aligned with the music data for time-series analysis. Preprocessing involved cleaning missing values, ensuring consistent date formats, and aligning economic metrics with the appropriate historical periods for comparison with music data.

- **Socio-economic datasets**: The dataset was cleaned and aggregated to match the yearly format of available social and economic data. Preprocessing steps included handling missing values, normalizing data across variables, and potentially transforming metrics such as valence and danceability into averages by year. Furthermore, non-relevant variables may have been removed, and multicollinearity checks were conducted to ensure the usability of the dataset in regression models.

If people are involved, were they aware of the data collection and if so, what purpose did they expect the data to be used for?

- **Music datasets**: The data originates from public platforms like **Spotify**, where users are generally aware that their listening habits are tracked. However, artists and listeners would likely be unaware of this specific usage of the data for correlating music preferences with economic trends.
- **Economic and socio-economic datasets**: Data on these indicators is collected by government agencies or non-profits mentioned above with the understanding that it will be publicly available for research, policy analysis, and economic forecasting. Individuals were not directly involved in the data collection process.

Where can your raw source data be found, if applicable?

1. **Music sentiment datasets**:

    - Caleb Elgut's dataset scraped from Spotify can be found on GitHub: [Spotify-LSTM Data](https://github.com/calebelgut/spotify-lstm/tree/main/Data)
    - Billboard Archive Data: [Billboard top 100](https://billboardtop100of.com/)
    - Joakim Arvidsson dataset scraped from Spotify found on Kraggle: [top10000 songs](https://www.kaggle.com/datasets/joebeachcapital/top-10000-spotify-songs-1961-now)
2. **Economic datasets:**

    - Unemployment rate data from Jan 1948 – Sep 2024 is available through the Federal Reserve Economic Data (FRED) platform: [Unemployment Rate](https://fred.stlouisfed.org/series/UNRATE)
    - Quarterly GDP from Jan 1997 – 2019: [Quarterly GDP](https://fred.stlouisfed.org/series/GDP)
    - Yearly GDP of the Music/Entertainment Industry from 1997 – 2019: [Entertainment Industry GDP](https://fred.stlouisfed.org/series/USPRFRMSPRTMSMNGSP)
    - Annual Returns on Investment of US Stocks (including dividends) from 1928-2023: [Stock Returns](https://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/histretSP.html)
3. **Socio-Economic Datasets**:

- Mortality trends since 1900-2018 by NCHS: [Death Rate](https://www.cdc.gov/nchs/data-visualization/mortality-trends/index.htm)
- Fertility rate, total (births per woman) by the World Bank: [Birth Rate](https://data.worldbank.org/indicator/SP.DYN.TFRT.IN?locations=US)
- Recession generated by FRED, borrowing expterise from The National Bureau of Economic Research (NBER): [Recession](https://fred.stlouisfed.org/series/USREC)
- Incarceration data from non-profit and non-partisan Prison Policy Initative: [Incarceration](https://www.prisonpolicy.org/data/)
- election variable self-generated by assign 1 or 0 every 4 years (1 when had election and 0 when no election, more specific code in recession_and_election.ipynb.

Cleaning

Overall Summary: We chose to web scrape directly from Billboard and match the Billboard song data with Spotify's sentiment data. This was done by using song titles and artist names as common identifiers. Variations in how song names or artist names were spelled were resolved to make sure the match was accurate. Economic data(eg. Fed fund rate, unemployment rate) was collected through different government sources and merged all economic data into one dataset. Additional control variables were collected to provide context to the economic variables(eg, birth rate, death rate, recession) or other factors influencing music trends. The first notebook below is the scraping process and the second is for combining all of the music, sociopolitical and economic data.

1. billboard_scraping.ipynb: Scraped billboard hot 100 data, which provides a historically accurate metric for the most popular songs of each year.

2. final_cleaned_data(1).ipynb: Process where billboard data is matched to its corresponding Spotify music sentiment data. This music data is then matched to economic variables(such as GDP, Unemployment Rate, Federal Funds Rate, and S&P500 stock) and other contextual variables (such as election and recession) which might also influence public music sentiment.

Data Standardization:

- Ensuring consistent formats for song titles and artist names in billboard and spotify dataset
- Ensuring economic variables were comparable across different datasets.
- Handling Missing Data: Missing values were identified and addressed by either excluding records or using imputation methods depending on the context of the missing data and its potential impact on analysis.

Final Merged Dataset:

- After the cleaning and merging steps, a comprehensive, standardized dataset was created that included:
  - Billboard data (song rankings and deta)
  - Spotify sentiment data (emotional tone of songs)
  - Economic data (macroeconomic indicators)
  - Control variables (additional contextual factors).

After merging all of these variables in the file name final_clean_data.csv

In [1]:
```
import pandas as pd
import numpy as np
import duckdb
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import statsmodels.api as sm
from statsmodels.api import OLS, add_constant
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import (train_test_split, KFold,
cross_val_score, cross_validate)
```

Final dataset that contains a comprehensive overview of all the music sentiment information, economic information, and additional control variable factors

```
In [48]: final = pd.read_csv('final_clean_data.csv')
```

inspect the data and see the columns present in the table

```
In [49]: print(final.head())
```

```
   Unnamed: 0  danceability  valence  year  fedfundrate  UnemploymentRate  \
0           0         0.325   0.9130  1960     3.215833          5.541667   
1           1         0.533   0.1905  1960     3.215833          5.541667   
2           2         0.498   0.8660  1960     3.215833          5.541667   
3           3         0.447   0.3300  1960     3.215833          5.541667   
4           4         0.558   0.3030  1960     3.215833          5.541667   

   recession  election     US_GDP  year_1  State_prisons  fertility  \
0          1         1  542.38225    1960       189735.0      3.654   
1          1         1  542.38225    1960       189735.0      3.654   
2          1         1  542.38225    1960       189735.0      3.654   
3          1         1  542.38225    1960       189735.0      3.654   
4          1         1  542.38225    1960       189735.0      3.654   

   DeathRate  S&P500  
0     1339.2     0.0  
1     1339.2     0.0  
2     1339.2     0.0  
3     1339.2     0.0  
4     1339.2     0.0  
```

Dropping index column not used in our analysis:

```
In [50]: print(final.columns)
         final.drop(columns=['year_1','Unnamed: 0'], axis=1, inplace=True)
         print(final.columns)
         print(final.head())
```

```
Index(['Unnamed: 0', 'danceability', 'valence', 'year', 'fedfundrate',
       'UnemploymentRate', 'recession', 'election', 'US_GDP', 'year_1',
       'State_prisons', 'fertility', 'DeathRate', 'S&P500'],
      dtype='object')
Index(['danceability', 'valence', 'year', 'fedfundrate', 'UnemploymentRate',
       'recession', 'election', 'US_GDP', 'State_prisons', 'fertility',
       'DeathRate', 'S&P500'],
      dtype='object')
   danceability  valence  year  fedfundrate  UnemploymentRate  recession  \
0         0.325   0.9130  1960     3.215833          5.541667          1   
1         0.533   0.1905  1960     3.215833          5.541667          1   
2         0.498   0.8660  1960     3.215833          5.541667          1   
3         0.447   0.3300  1960     3.215833          5.541667          1   
4         0.558   0.3030  1960     3.215833          5.541667          1   

   election     US_GDP  State_prisons  fertility  DeathRate  S&P500  
0         1  542.38225       189735.0      3.654     1339.2     0.0  
1         1  542.38225       189735.0      3.654     1339.2     0.0  
2         1  542.38225       189735.0      3.654     1339.2     0.0  
3         1  542.38225       189735.0      3.654     1339.2     0.0  
4         1  542.38225       189735.0      3.654     1339.2     0.0  
```

There are rows in S&P500 where the value equal to 0. Drop the rows.

```
In [51]: final = final[final['S&P500'] != 0]
         print(final.head())
```

```
    danceability  valence  year  fedfundrate  UnemploymentRate  recession  \
83         0.565   0.9635  1961        1.955          6.691667          1   
84         0.525   0.6310  1961        1.955          6.691667          1   
85         0.529   0.7410  1961        1.955          6.691667          1   
86         0.354   0.6230  1961        1.955          6.691667          1   
87         0.481   0.9535  1961        1.955          6.691667          1   

    election     US_GDP  State_prisons  fertility  DeathRate  S&P500  
83         0  562.20975       196453.0       3.62     1298.8    0.27  
84         0  562.20975       196453.0       3.62     1298.8    0.27  
85         0  562.20975       196453.0       3.62     1298.8    0.27  
86         0  562.20975       196453.0       3.62     1298.8    0.27  
87         0  562.20975       196453.0       3.62     1298.8    0.27  
```

Run the correlation matrix to see the strength and direction between variables

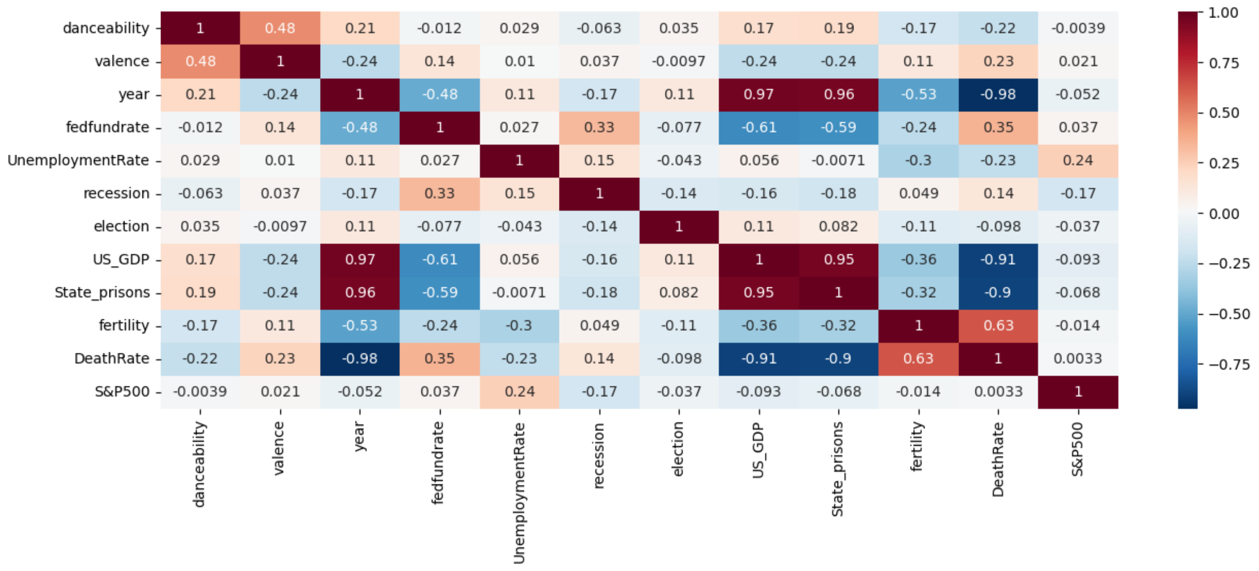
```
In [52]: print(final.corr())
```

```
                  danceability   valence      year  fedfundrate  \
danceability          1.000000  0.478538  0.208863    -0.012125   
valence               0.478538  1.000000 -0.244164     0.140404   
year                  0.208863 -0.244164  1.000000    -0.484333   
fedfundrate          -0.012125  0.140404 -0.484333     1.000000   
UnemploymentRate      0.028724  0.010264  0.112782     0.026551   
recession            -0.063176  0.037190 -0.168513     0.333548   
election              0.035226 -0.009652  0.114511    -0.077159   
US_GDP                0.173858 -0.239765  0.968421    -0.614557   
State_prisons         0.190238 -0.240677  0.958372    -0.593281   
fertility            -0.170664  0.113077 -0.528370    -0.243675   
DeathRate            -0.224345  0.232455 -0.976610     0.352490   
S&P500               -0.003924  0.020581 -0.052381     0.037377   

                  UnemploymentRate  recession  election    US_GDP  \
danceability              0.028724  -0.063176  0.035226  0.173858   
valence                   0.010264   0.037190 -0.009652 -0.239765   
year                      0.112782  -0.168513  0.114511  0.968421   
fedfundrate               0.026551   0.333548 -0.077159 -0.614557   
UnemploymentRate          1.000000   0.146262 -0.042659  0.056056   
recession                 0.146262   1.000000 -0.136231 -0.158056   
election                 -0.042659  -0.136231  1.000000  0.114523   
US_GDP                    0.056056  -0.158056  0.114523  1.000000   
State_prisons            -0.007095  -0.184399  0.082246  0.953831   
fertility                -0.304178   0.048557 -0.109818 -0.356798   
DeathRate                -0.230308   0.135190 -0.098228 -0.907178   
S&P500                    0.243424  -0.167556 -0.036805 -0.093406   

                  State_prisons  fertility  DeathRate    S&P500  
danceability           0.190238  -0.170664  -0.224345 -0.003924  
valence               -0.240677   0.113077   0.232455  0.020581  
year                   0.958372  -0.528370  -0.976610 -0.052381  
fedfundrate           -0.593281  -0.243675   0.352490  0.037377  
UnemploymentRate      -0.007095  -0.304178  -0.230308  0.243424  
recession             -0.184399   0.048557   0.135190 -0.167556  
election               0.082246  -0.109818  -0.098228 -0.036805  
US_GDP                 0.953831  -0.356798  -0.907178 -0.093406  
State_prisons          1.000000  -0.319037  -0.904853 -0.068427  
fertility             -0.319037   1.000000   0.630240 -0.014068  
DeathRate             -0.904853   0.630240   1.000000  0.003282  
S&P500                -0.068427  -0.014068   0.003282  1.000000  
```

Heatmap is used to better visualize the correlation

At first sight, the many area of high correlation (negative or positive), however, keep in mind we are comparing the danceability and valence to the other metrics. And across the board (not counting each other), is it not very correlated with any of the econometric indicators. Next, looking at the correlation between economic data, we see some relatively high correlation such as the correlation between fed fund rate and US GDP and then GDP with death rate, so later when we run the multivariate regression, we will try to avoid those combinations.

In [7]:
```
fig, ax = plt.subplots(figsize=(15, 5))
heatmap = sns.heatmap(final.corr(numeric_only=True), cmap="RdBu_r", annot=True)
```



	danceability	valence	year	fedfundrate	UnemploymentRate	recession	election	US_GDP	State_prisons	fertility	DeathRate	S&P500
danceability	1	0.48	0.21	-0.012	0.029	-0.063	0.035	0.17	0.19	-0.17	-0.22	-0.0039
valence	0.48	1	-0.24	0.14	0.01	0.037	-0.0097	-0.24	-0.24	0.11	0.23	0.021
year	0.21	-0.24	1	-0.48	0.11	-0.17	0.11	0.97	0.96	-0.53	-0.98	-0.052
fedfundrate	-0.012	0.14	-0.48	1	0.027	0.33	-0.077	-0.61	-0.59	-0.24	0.35	0.037
UnemploymentRate	0.029	0.01	0.11	0.027	1	0.15	-0.043	0.056	-0.0071	-0.3	-0.23	0.24
recession	-0.063	0.037	-0.17	0.33	0.15	1	-0.14	-0.16	-0.18	0.049	0.14	-0.17
election	0.035	-0.0097	0.11	-0.077	-0.043	-0.14	1	0.11	0.082	-0.11	-0.098	-0.037
US_GDP	0.17	-0.24	0.97	-0.61	0.056	-0.16	0.11	1	0.95	-0.36	-0.91	-0.093
State_prisons	0.19	-0.24	0.96	-0.59	-0.0071	-0.18	0.082	0.95	1	-0.32	-0.9	-0.068
fertility	-0.17	0.11	-0.53	-0.24	-0.3	0.049	-0.11	-0.36	-0.32	1	0.63	-0.014
DeathRate	-0.22	0.23	-0.98	0.35	-0.23	0.14	-0.098	-0.91	-0.9	0.63	1	0.0033
S&P500	-0.0039	0.021	-0.052	0.037	0.24	-0.17	-0.037	-0.093	-0.068	-0.014	0.0033	1

Using Averages

Based on our previous iteration (detailed in the appendix below), we concluded that using a model where a constant value is used to predict a range of varying values is not robust. For example, having only one GDP value per year makes it unsuitable to train a model to predict a broad range of music valence values between 0 and 1 in that given year. The regression plot in the appendix provides a clearer visualization of this issue.

To address this, we decided to average the valence and danceability of songs per year, acknowledging the trade-off of losing many data points, as discussed in the limitations section. This ensures that during model training, each independent value corresponds to a single dependent value, rather than multiple varying dependent values.

Created a data frame averaging danceability and valence of songs per year.

In [9]:
```
averagefinal = duckdb.sql("""
                                SELECT
                                    Year,
                                    ANY_VALUE(recession) AS recession,
                                    ANY_VALUE(election) AS election,
                                    AVG(danceability) AS avg_danceability,
                                    AVG(valence) AS avg_valence,
                                    ANY_VALUE(US_GDP) AS gdp,
                                    ANY_VALUE(UnemploymentRate) AS unemployment,
                                    ANY_VALUE(fedfundrate) AS fedfundrate,
                                    ANY_VALUE(State_prisons) AS imprisonment,
                                    ANY_VALUE(fertility) AS fertility,
                                    ANY_VALUE(DeathRate) AS deathrate,
                                    ANY_VALUE("S&P500") AS "S&P500"
                                FROM
                                    final
                                GROUP BY
                                    Year
                                ORDER BY
                                    Year
                                """).df()

averagefinal.head()
print(averagefinal.shape)
```

```
(55, 12)
```

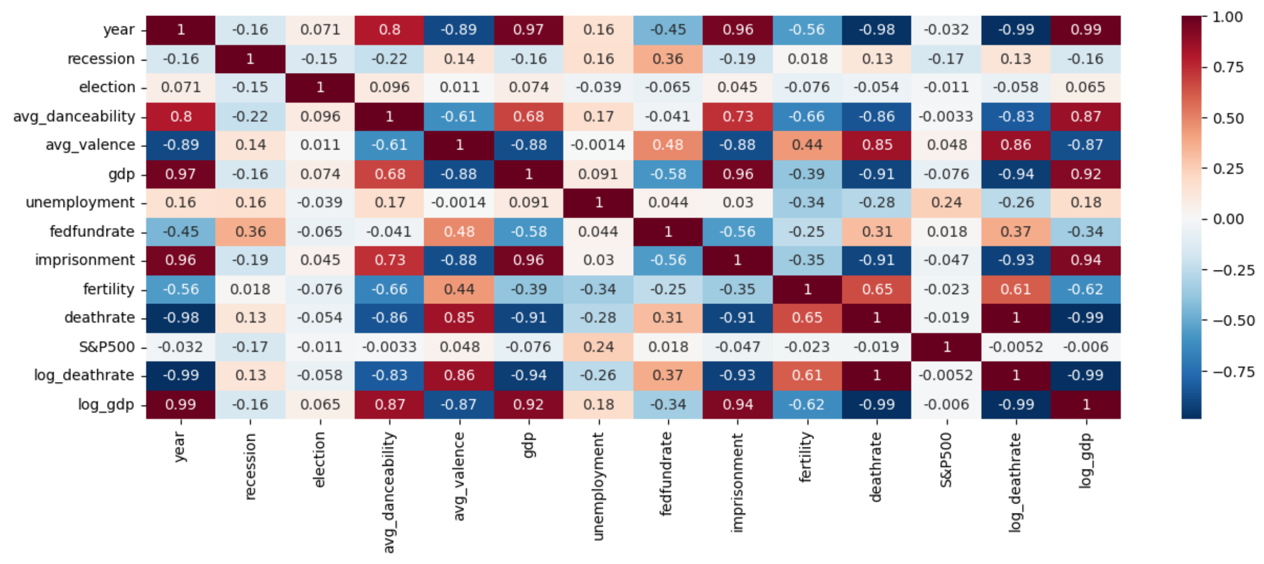
Based on the later analysis and modeling, we noticed multilinearity problem due to the large values we have for gdp and death rate. Hence, we will add 2 new columns for log_gdp and log_deathrate to smoosh the data into smaller range.

```
In [53]: averagefinal["log_deathrate"]=np.log(averagefinal["deathrate"])
         averagefinal["log_gdp"]=np.log(averagefinal["gdp"])
         print(averagefinal.head())
```

```
   year  recession  election  avg_danceability  avg_valence        gdp  \
0  1961          1         0          0.539533     0.714757  562.20975
1  1962          0         0          0.562227     0.701820  603.92150
2  1963          0         0          0.545594     0.714175  637.45150
3  1964          0         1          0.531146     0.692634  684.46150
4  1965          0         0          0.526300     0.652230  742.29025

   unemployment  fedfundrate  imprisonment  fertility  deathrate  S&P500  \
0      6.691667     1.955000      196453.0      3.620     1298.8    0.27
1      5.566667     2.708333      194886.0      3.461     1323.6   -0.09
2      5.641667     3.178333      194155.0      3.319     1346.3    0.23
3      5.158333     3.496667      192627.0      3.190     1303.8    0.16
4      4.508333     4.075000      189855.0      2.913     1306.5    0.12

   log_deathrate   log_gdp  trans_fertility  interaction  stock_election
0       7.169196  6.331875         0.210338         0.00            0.00
1       7.188111  6.403444         0.210226        -0.00           -0.00
2       7.205115  6.457478         0.210098         0.00            0.00
3       7.173038  6.528632         0.209951         0.16            0.16
4       7.175107  6.609740         0.209488         0.00            0.00
```

Created a heatmap to project the correlation matrix:

```
In [11]: fig, ax = plt.subplots(figsize=(15, 5))
         heatmap = sns.heatmap(averagefinal.corr(numeric_only=True), cmap="RdBu_r", annot=True)
```



Pre-registration Statement

Hypothesis 1: Music Valence vs. US GDP

The average valence of popular songs by year from 1961-2016 in the US has an inverse relationship with the corresponding US Gross Domestic Product (GDP) of the given year.

- H_0: The average valence of popular Billboard songs per year has no relationship to the US GDP each year ($\beta = 0$).
- H_A: The average valence of popular Billboard songs per year has an inverse relationship to the US GDP each year ($\beta < 0$).

We hypothesize a statistically significant negative coefficient ($\beta_1 < 0$) for GDP per capita. During periods of economic prosperity (higher GDP), cultural sentiments may reflect more subdued emotions in music, leading to lower valence scores. This relationship is supported by existing literature on the connection between societal well-being and cultural expressions, which indicates that wealthier societies may prioritize complexity or emotional depth in art over positivity.

To test this hypothesis, we will conduct a simple linear regression using GDP per capita as the independent variable and average valence of songs as the dependent variable. The regression will determine the direction and strength of the relationship, supported by visualizations such as scatterplots with fitted regression lines. A residual plot is plotted to visually assess how well a regression model fits the data by displaying the difference between the actual data points and the predicted values (residuals). If there is heteroskedasticity, transformation is required. Evaluation metrics like RMSE and MAE will be examined to ensure model's robustness. Finally, statistical metrics from the OLS model, such as p-values, will evaluate the significance of the model.

Hypothesis 2: Danceability vs Macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession)

The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) have relationship with the average danceability of the most popular songs per year from 1961-2016 in the US.

- H_0: The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) have no relationship with the average danceability of the most popular songs per year from 1961-2016 in the US. ($\beta_1, \beta_2, \beta_3, \beta_4, \beta_5 = 0$)
- H_A: The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) have relationship with the average danceability of the most popular songs per year from 1961-2016 in the US. (at least one $\beta_n \neq$ 0 for n=1,2,3,4,5)

We expect a statistically significant negative coefficient ($\beta_1 < 0$) for unemployment rate, reflecting an inverse relationship between unemployment and danceability. High unemployment rates may lower the average danceability of songs as societal stress and economic hardships can influence cultural output, leading to less upbeat and more somber music trends.

To test this hypothesis, we will conduct a multivariate linear regression using average danceability of songs as the dependent variable and unemployment rate as the primary independent variable. Additional economic and social variables, such as death rate, election years, S&P 500, and recession indicators, will be included as covariates to account for potential confounders. A residual plot is plotted to visually assess how well a regression model fits the data by displaying the difference between the actual data points and the predicted values (residuals). Evaluation metrics like RMSE and MAE will be examined to ensure model's robustness. Finally, statistical metrics from the OLS model, such as p-values, will evaluate the significance of the model.

- Note that our original hypothesis 2 was danceability vs unemployment rate. We decided to add in more variables to the model to make the model more robust and we were also interested in learning the relationship of other variables to danceability.

Data Exploration and Analysis

Hypothesis 1: Music Valence vs. US GDP

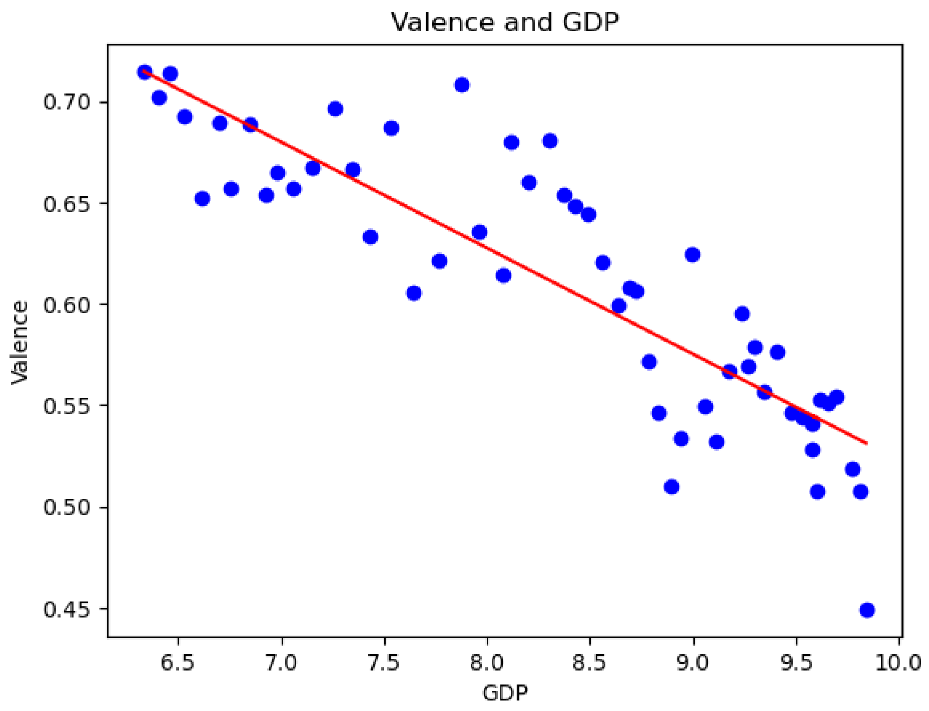
- H_0: The average valence of the top 100 Spotify songs per year has no relationship to the US GDP each year ($\beta = 0$).
- H_A: The average valence of the top 100 Spotify songs per year has an inverse relationship to the US GDP each year ($\beta < 0$).

Reject null hypothesis if p-value is less than 0.05.

Modeling and Visualization

In [12]:
```
X_gdp = averagefinal[["log_gdp"]].values
y_gdp = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_gdp, y_gdp)
print("GDP Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_gdp = model.predict(X_gdp)

plt.scatter(X_gdp, y_gdp, color="blue")
plt.plot(X_gdp, y_pred_gdp, color="red", label="Fitted line")
plt.title("Valence and GDP")
plt.xlabel("GDP")
plt.ylabel("Valence")
plt.show()
```

```
GDP Coeff: -0.0523
Intercept: 1.0458
```

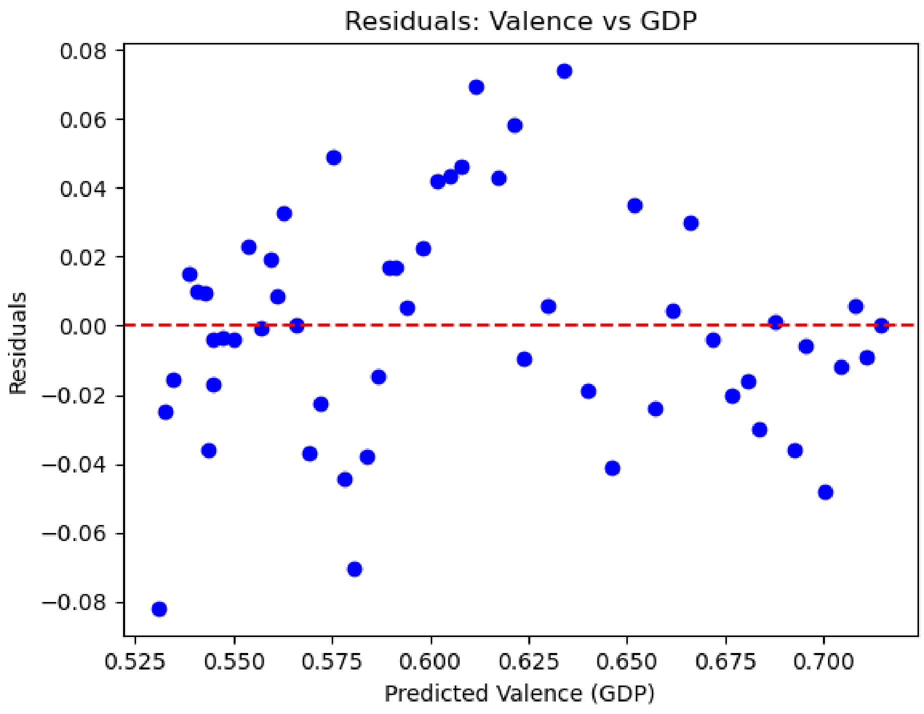


Residual Plot

We created a residual plot to assess the randomness of the residuals and determine if any transformations were necessary. The residuals appeared random, indicating that no transformations were needed.

In [13]:
```
#GDP and Valence
X_gdp = averagefinal[["log_gdp"]].values
y_gdp = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_gdp, y_gdp)
y_pred_gdp = model.predict(X_gdp)
residuals_gdp = y_gdp - y_pred_gdp

plt.scatter(y_pred_gdp, residuals_gdp, color="blue")
plt.axhline(y=0, color='red', linestyle='--')  #Add a line at zero
plt.title("Residuals: Valence vs GDP")
plt.xlabel("Predicted Valence (GDP)")
plt.ylabel("Residuals")
```

Out[13]:
```
Text(0, 0.5, 'Residuals')
```



Perform cross validation to examine model performance and generalizability.

In [14]:
```
X = averagefinal[['log_gdp']]
y = averagefinal['avg_valence']
kf = KFold(n_splits = 5)
results = cross_validate(
    LinearRegression(),X,y,
    scoring='neg_root_mean_squared_error',
    cv=kf,return_train_score=True)
results_df = pd.DataFrame({
    'Train_RMSE':np.round(-1*results['train_score'],4),
    'Test_RMSE':np.round(-1*results['test_score'],4)})
print('Table summarizing train and test RMSE values across splits:')
print(results_df)
print('Mean Train RMSE:',round(np.mean(results_df['Train_RMSE']),4))
print('Mean Test RMSE:', round(np.mean(results_df['Test_RMSE']),4))
print('Std Train RMSE:',round(np.std(results_df['Train_RMSE']),4))
print('Std Test RMSE:',round(np.std(results_df['Test_RMSE']),4))
print('Normalized Test RMSE:',round(
    np.mean(results_df['Test_RMSE'])/(averagefinal['avg_valence'].max() -
                                  averagefinal['avg_valence'].min()),4))
```

```
Table summarizing train and test RMSE values across splits:
   Train_RMSE  Test_RMSE
0      0.0327     0.0477
1      0.0313     0.0365
2      0.0304     0.0410
3      0.0316     0.0349
4      0.0321     0.0386
Mean Train RMSE: 0.0316
Mean Test RMSE: 0.0397
Std Train RMSE: 0.0008
Std Test RMSE: 0.0045
Normalized Test RMSE: 0.1496
```

Evaluation of Significance

Based on the RMSE values above, we can conclude:

- The model seems to have consistent performance across different splits, as indicated by the similar train and test RMSE values for each split. The RMSE values are contained in a small range, from 0.0304-0.0477.

- The mean train RMSE value is relatively lower than the mean test RMSE. This indicates that in general the model perform better in the training model. The mean test RMSE value of 0.0316 suggests that on average, the model's predictions are within a reasonable range of the actual values in the test data.
- Looking across splits, the test RMSE values are relatively close to the training RMSE values. Additionally, the train and test RMSE means across splits are approximately equal, with the mean train RMSE at 0.0316 and the mean test RMSE at 0.0397. Thus, we can assume reasonable generalization to unseen data.
- The standard deviation of the Train RMSE is 0.0008, showing relatively low variability in the model's predictions between training sets.
- The standard deviation of the Test RMSE is 0.0045, suggesting higher variability compared to the training set. Therefore, we can conclude that the model's ability to generalize to unseen data is less consistent. However, based on the fact that that the mean train RMSE and mean test RMSE are similar, and that the test standard deviation is still relatively low, we can still assume reasonable generalization.
- Normalized RMSE value is calculated to gain a better idea of if this RMSE value was “good.” For normalized RMSE, the range is between 0 and 1 where values closer to 0 are considered “better” results. This is helpful because RMSE values are sometimes hard to compare because of unit inconsistencies and the range of the dataset itself. From this normalization, we got an RMSE value of 0.1496, which is relatively close to 0 and thus suggests the regression model fits the test data well.

In [15]:
```
mae = mean_absolute_error(y, model.predict(X))
print(f'MAE: {round(mae,5)}')
```

```
MAE: 0.02503
/Users/kuankhaixin/anaconda3/envs/info2950/lib/python3.11/site-packages/sklearn/base.py:486: UserWa
rning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Note: RMSE is greater than MAE, so we conclude that the model is reliable.

OLS Regression to evaluate significance

Finally, we run an OLS regression. Running an OLS regression with statistical output provides detailed metrics, such as p-values and confidence intervals, which are crucial for hypothesis testing.

In [16]:
```
X = averagefinal[["log_gdp"]]
y = averagefinal["avg_valence"]
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            avg_valence   R-squared:                       0.755
Model:                            OLS   Adj. R-squared:                  0.750
Method:                 Least Squares   F-statistic:                     163.1
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           8.35e-18
Time:                        19:55:23   Log-Likelihood:                 110.80
No. Observations:                  55   AIC:                            -217.6
Df Residuals:                      53   BIC:                            -213.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0458      0.034     30.373      0.000       0.977       1.115
log_gdp       -0.0523      0.004    -12.771      0.000      -0.061      -0.044
==============================================================================
Omnibus:                        0.228   Durbin-Watson:                   1.195
Prob(Omnibus):                  0.892   Jarque-Bera (JB):                0.017
Skew:                           0.040   Prob(JB):                        0.992
Kurtosis:                       3.031   Cond. No.                         66.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

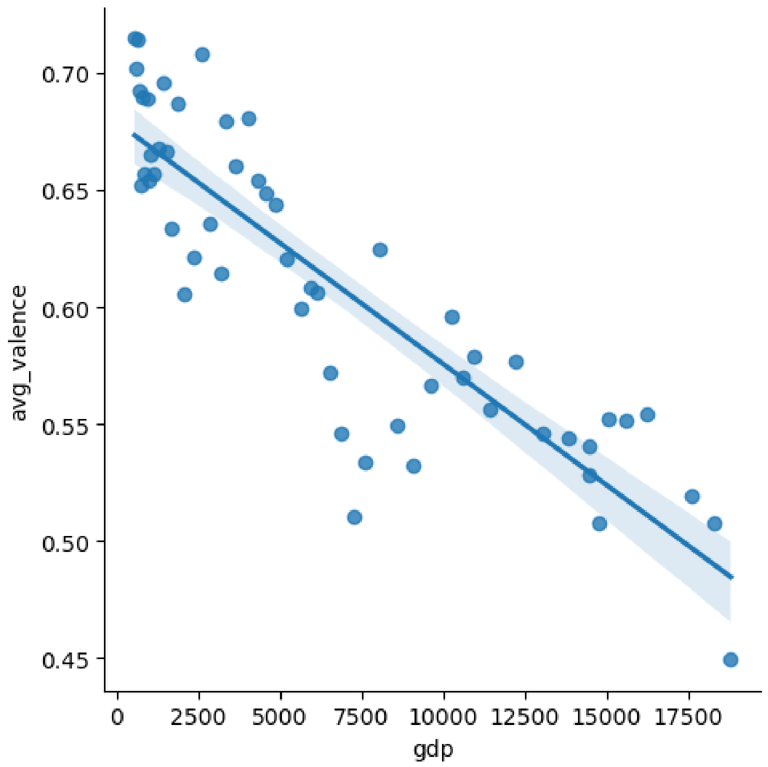
Summary

- Our significance level is set at 5%. Since the OLS regression model defaults to a two-sided t-test, we adjusted the p-value by dividing it by two to reflect a one-sided t-test instead, testing specifically for whether B<0. After adjustment, the p-value remains 0.000, indicating strong statistical significance and supporting the hypothesis that GDP negatively affects valence. Null hypothesis is rejected.
- The model suggests that a 1% increase in GDP is associated with a decrease of 0.000523 units in the average valence of popular songs in that year. While this effect size may appear small, it is reasonable given that the valence scale ranges from 0 to 1.

Bootstrap to get the confidence interval

In [17]:
```
sns.lmplot(x="gdp", y= "avg_valence" , data=averagefinal, fit_reg=True, ci=95, n_boot=1000)
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x2a0111090>
```



Light blue region shows regression lines within the 95% confidence level.

Expanding Hypothesis 1

We were also curious about the relationship between valence and other variables (such as fertility, fed fund rate and death rate), hence deciding to run more linear regression.

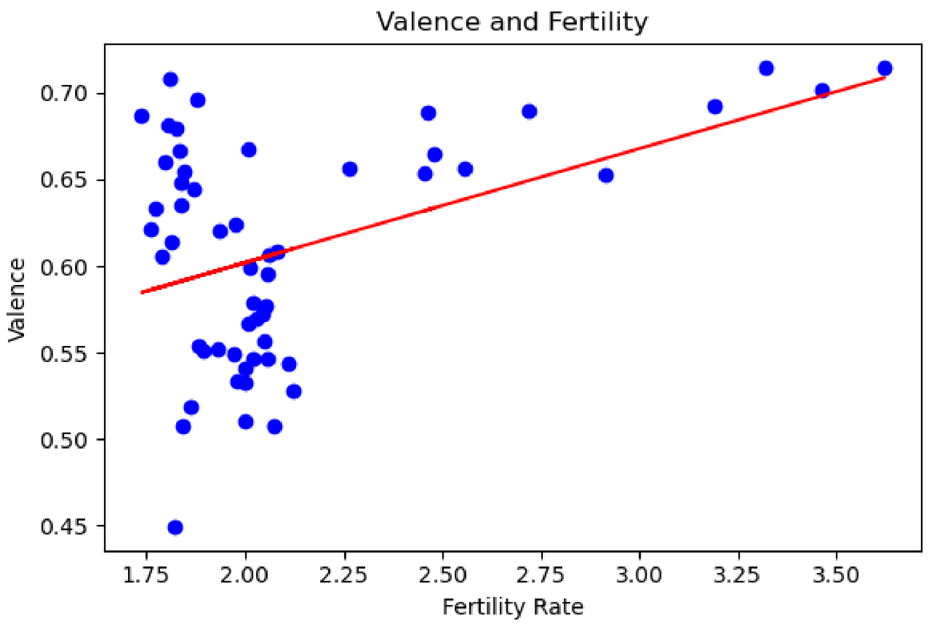
- Fertility
  - H_0: The average valence of the top 100 Spotify songs per year has no relationship to the fertility rate each year ($\beta = 0$).
  - H_A: The average valence of the top 100 Spotify songs per year has arelationship to the fertility rate each year ($\beta \neq 0$).
- Fed fund rate
  - H_0: The average valence of the top 100 Spotify songs per year has no relationship to the Fed fund rate each year ($\beta = 0$).
  - H_A: The average valence of the top 100 Spotify songs per year has a inverse relationship to the Fed fund rate each year ($\beta \neq 0$).
- Death rate
  - H_0: The average valence of the top 100 Spotify songs per year has no relationship to the Death rate each year ($\beta = 0$).
  - H_A: The average valence of the top 100 Spotify songs per year has a inverse relationship to the Death rate each year ($\beta \neq 0$).

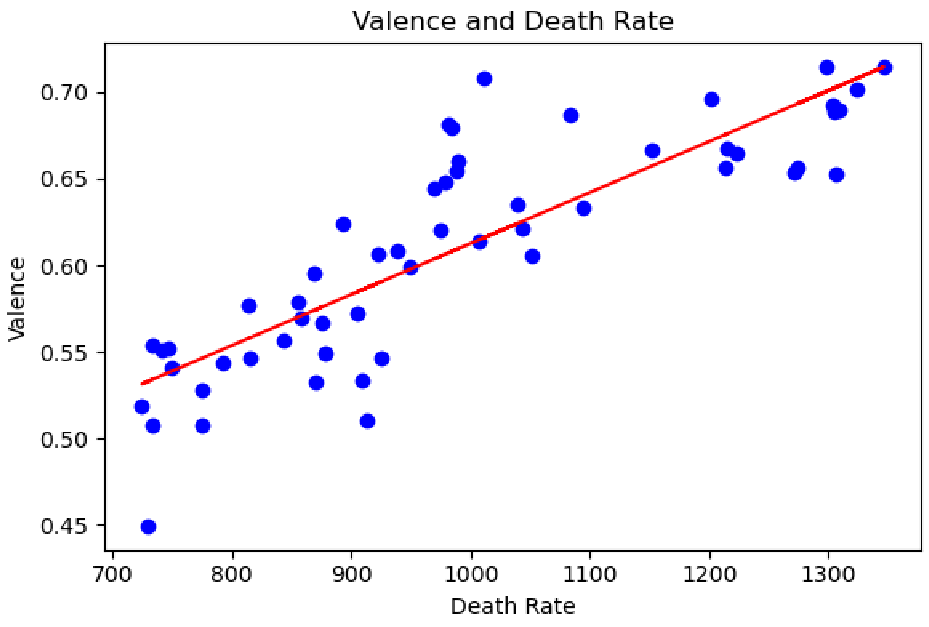
Modeling and visualization

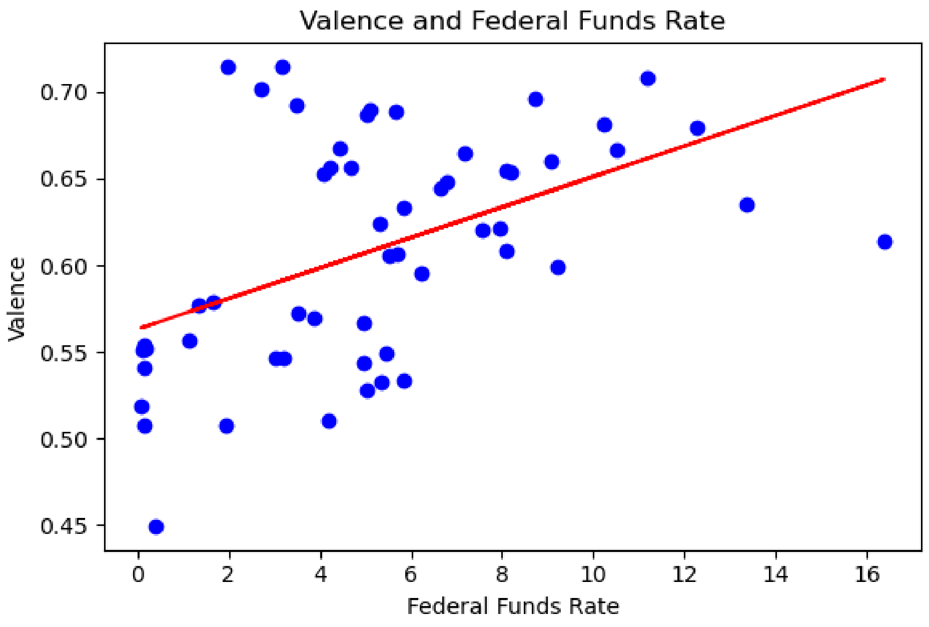
In [18]:
```
#Create a 3x1 grid of subplots
fig, axs = plt.subplots(3, figsize=(6, 12))

#Fertility and Valence
```

```python
X_fertility = averagefinal[["fertility"]].values
y_fertility = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fertility, y_fertility)
print("Fertility Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_fertility = model.predict(X_fertility)

axs[0].scatter(X_fertility, y_fertility, color="blue")
axs[0].plot(X_fertility, y_pred_fertility, color="red", label="Fitted line")
axs[0].set_title("Valence and Fertility")
axs[0].set_xlabel("Fertility Rate")
axs[0].set_ylabel("Valence")

#Death Rate and Valence
X_deathrate = averagefinal[["deathrate"]].values
y_deathrate = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_deathrate, y_deathrate)
print("Death Rate Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_deathrate = model.predict(X_deathrate)

axs[1].scatter(X_deathrate, y_deathrate, color="blue")
axs[1].plot(X_deathrate, y_pred_deathrate, color="red", label="Fitted line")
axs[1].set_title("Valence and Death Rate")
axs[1].set_xlabel("Death Rate")
axs[1].set_ylabel("Valence")

#Federal Funds Rate and Valence
X_fedfundrate = averagefinal[["fedfundrate"]].values
y_fedfundrate = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fedfundrate, y_fedfundrate)
print("Fed Funds Rate Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_fedfundrate = model.predict(X_fedfundrate)

axs[2].scatter(X_fedfundrate, y_fedfundrate, color="blue")
axs[2].plot(X_fedfundrate, y_pred_fedfundrate, color="red", label="Fitted line")
axs[2].set_title("Valence and Federal Funds Rate")
axs[2].set_xlabel("Federal Funds Rate")
axs[2].set_ylabel("Valence")

#Adjust layout to avoid overlap (in references)
plt.tight_layout()
```

```
Fertility Coeff: 0.0658
Intercept: 0.4702
Death Rate Coeff: 0.0003
Intercept: 0.3183
Fed Funds Rate Coeff: 0.0088
Intercept: 0.5632
```


Valence and Fertility
Valence
Fertility Rate
0.70
0.65
0.60
0.55
0.50
0.45
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50

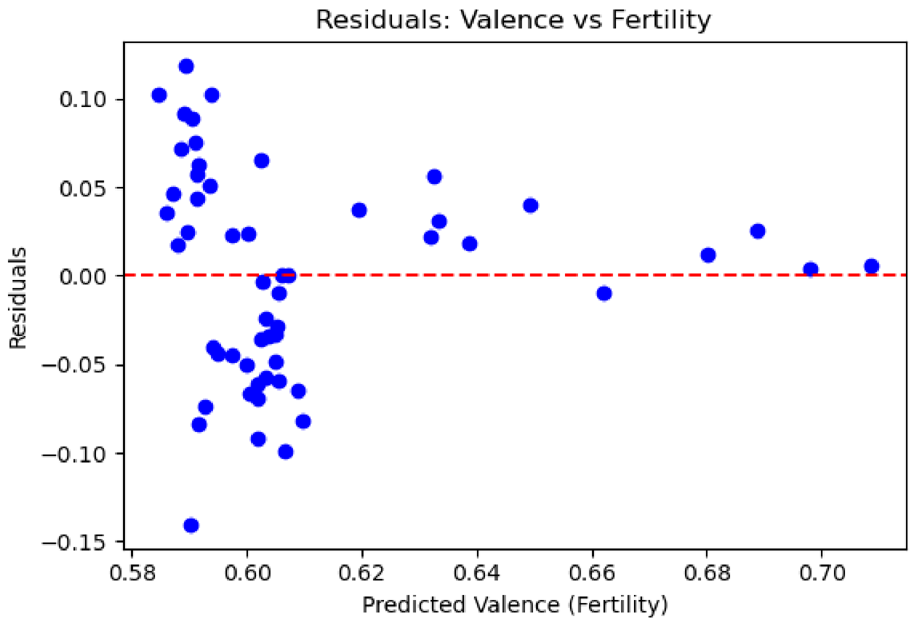

Valence and Death Rate
Valence
Death Rate
0.70
0.65
0.60
0.55
0.50
0.45
700
800
900
1000
1100
1200
1300

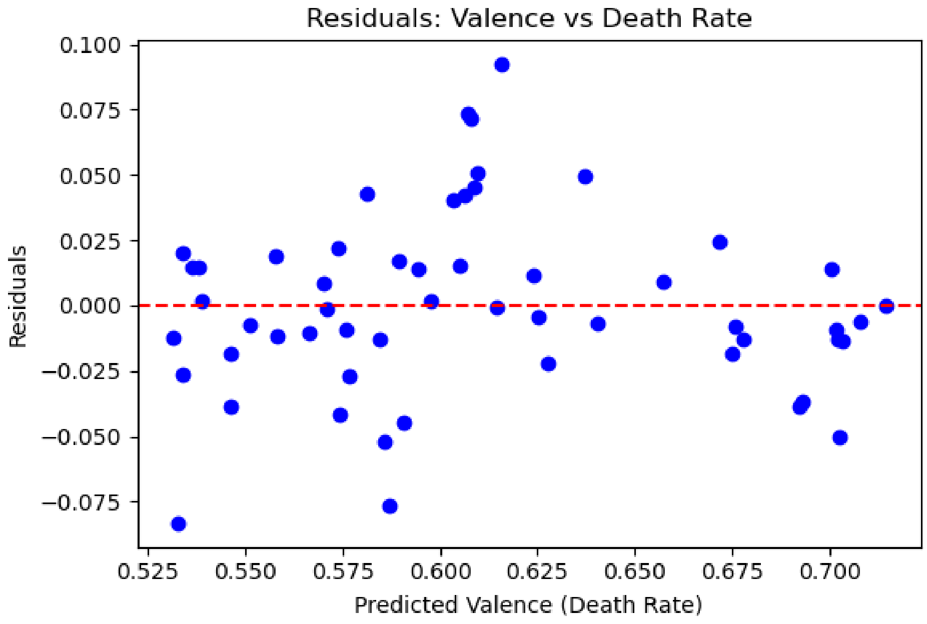

Valence and Federal Funds Rate
Valence
Federal Funds Rate
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8
10
12
14
16

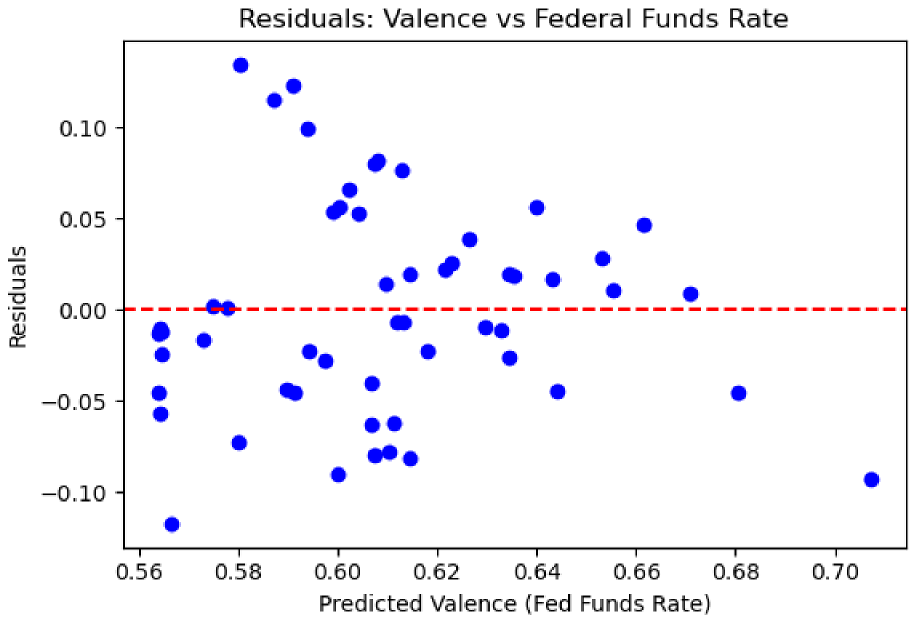
Residual Plot

We created residual plots to check the randomness of the residuals, which helps us determine whether transformations need to be made. In this case, the residual plot revealed that the fertility variable exhibited heteroskedasticity. To address this, we attempted transformation to improve the model's performance.

In [19]:
```python
fig, axs = plt.subplots(3, figsize=(6, 12))

# Fertility and Valence
X_fertility = averagefinal[["fertility"]].values
y_fertility = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fertility, y_fertility)
y_pred_fertility = model.predict(X_fertility)
residuals_fertility = y_fertility - y_pred_fertility

axs[0].scatter(y_pred_fertility, residuals_fertility, color="blue")
axs[0].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[0].set_title("Residuals: Valence vs Fertility")
axs[0].set_xlabel("Predicted Valence (Fertility)")
axs[0].set_ylabel("Residuals")

# Death Rate and Valence
X_deathrate = averagefinal[["deathrate"]].values
y_deathrate = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_deathrate, y_deathrate)
y_pred_deathrate = model.predict(X_deathrate)
residuals_deathrate = y_deathrate - y_pred_deathrate

axs[1].scatter(y_pred_deathrate, residuals_deathrate, color="blue")
axs[1].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[1].set_title("Residuals: Valence vs Death Rate")
axs[1].set_xlabel("Predicted Valence (Death Rate)")
axs[1].set_ylabel("Residuals")

# Federal Funds Rate and Valence
X_fedfundrate = averagefinal[["fedfundrate"]].values
y_fedfundrate = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fedfundrate, y_fedfundrate)
y_pred_fedfundrate = model.predict(X_fedfundrate)
residuals_fedfundrate = y_fedfundrate - y_pred_fedfundrate

axs[2].scatter(y_pred_fedfundrate, residuals_fedfundrate, color="blue")
axs[2].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[2].set_title("Residuals: Valence vs Federal Funds Rate")
axs[2].set_xlabel("Predicted Valence (Fed Funds Rate)")
axs[2].set_ylabel("Residuals")
plt.tight_layout()
plt.show()
```


Residuals: Valence vs Fertility
Residuals
Predicted Valence (Fertility)
0.10
0.05
0.00
−0.05
−0.10
−0.15
0.58
0.60
0.62
0.64
0.66
0.68
0.70

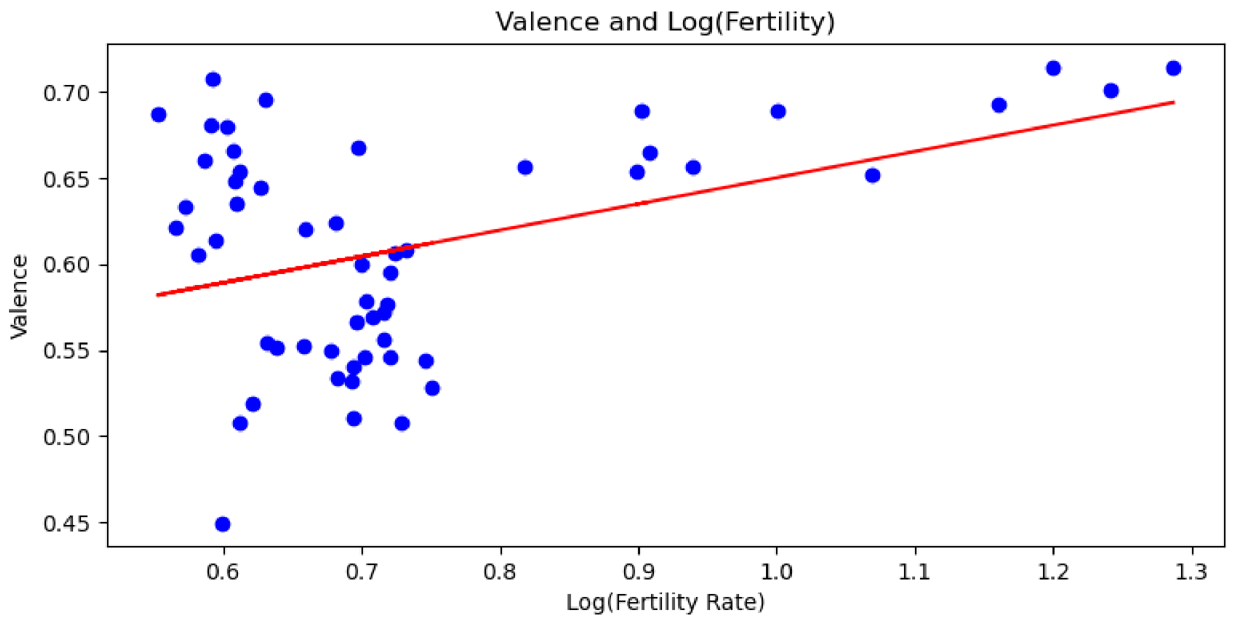

Residuals: Valence vs Death Rate
Residuals
Predicted Valence (Death Rate)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
0.525
0.550
0.575
0.600
0.625
0.650
0.675
0.700

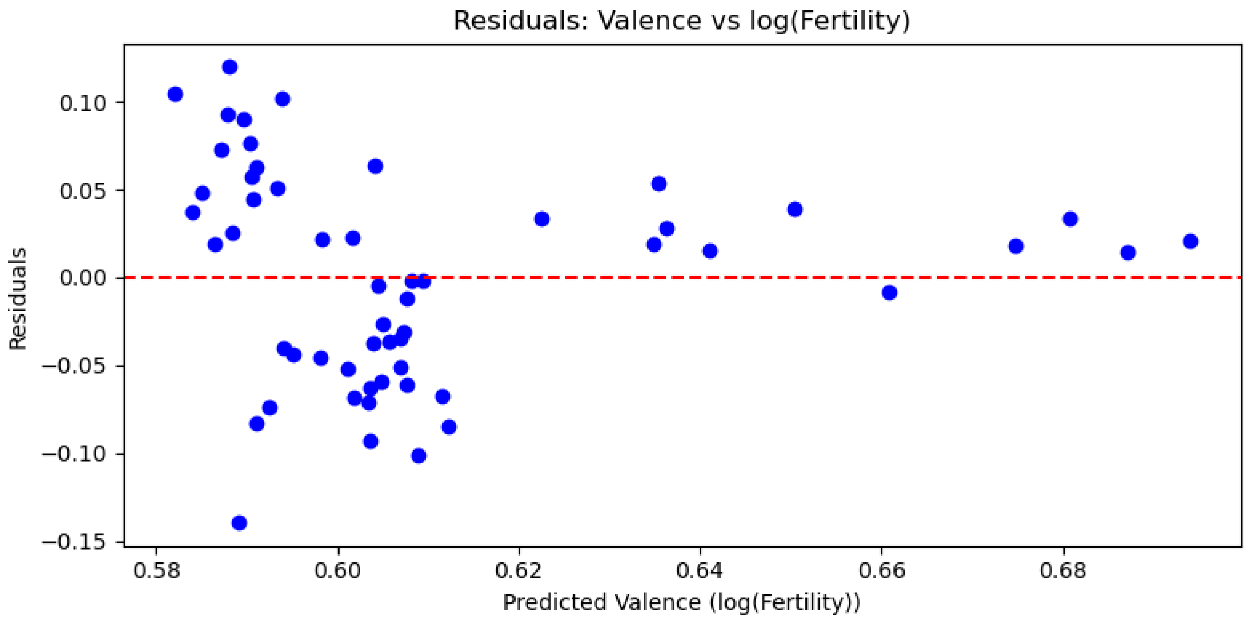

Residuals: Valence vs Federal Funds Rate
Residuals
Predicted Valence (Fed Funds Rate)
0.10
0.05
0.00
−0.05
−0.10
0.56
0.58
0.60
0.62
0.64
0.66
0.68
0.70

A log transformation was applied to the fertility variable, and another residual plot was made again to assess its effect. However, log transformation proved to be of little as heteroskedasticity is still present in the model.

In [20]:
```
fig, axs = plt.subplots(2,1, figsize=(8, 8))

#Logged Fertility and Valence
X_fer = np.log(averagefinal[["fertility"]].values)
y_fertility = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fer, y_fertility)
print("Log(Fertility) Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_fertility = model.predict(X_fer)

axs[0].scatter(X_fer, y_fertility, color="blue")
axs[0].plot(X_fer, y_pred_fertility, color="red", label="Fitted line")
axs[0].set_title("Valence and Log(Fertility)")
axs[0].set_xlabel("Log(Fertility Rate)")
axs[0].set_ylabel("Valence")
residuals_fertility = y_fertility - y_pred_fertility

axs[1].scatter(y_pred_fertility, residuals_fertility, color="blue")
axs[1].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[1].set_title("Residuals: Valence vs log(Fertility)")
axs[1].set_xlabel("Predicted Valence (log(Fertility))")
axs[1].set_ylabel("Residuals")
plt.tight_layout()
```

```
Log(Fertility) Coeff: 0.1528
Intercept: 0.4975
```





We then attempted the **boxcox**.transformation to transform the fertility data. Based on the reading online, if the lamda value is equal to 0, then log transformation will be done. If lambda value not equal to zero, it will perform transformation using the formula $(x^{lambda} - 1)/lamda$. In our case, lamda equal to -4.74

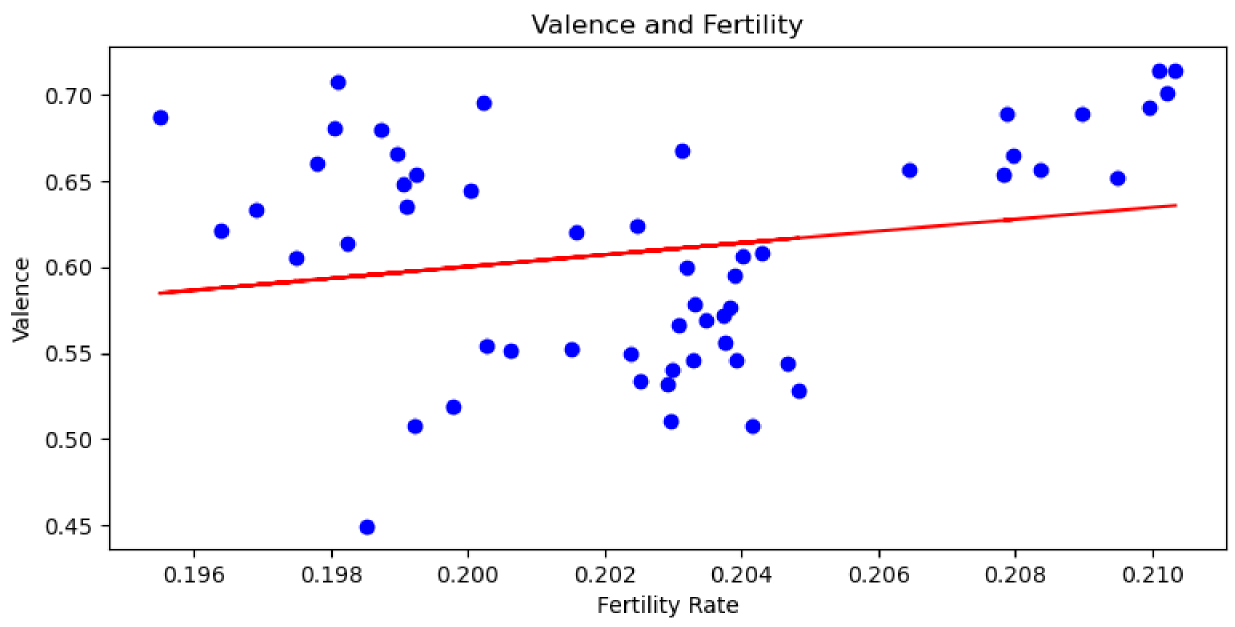
In [21]:
```
from scipy import stats
transformed_data, lambda_value = stats.boxcox(averagefinal["fertility"])
```

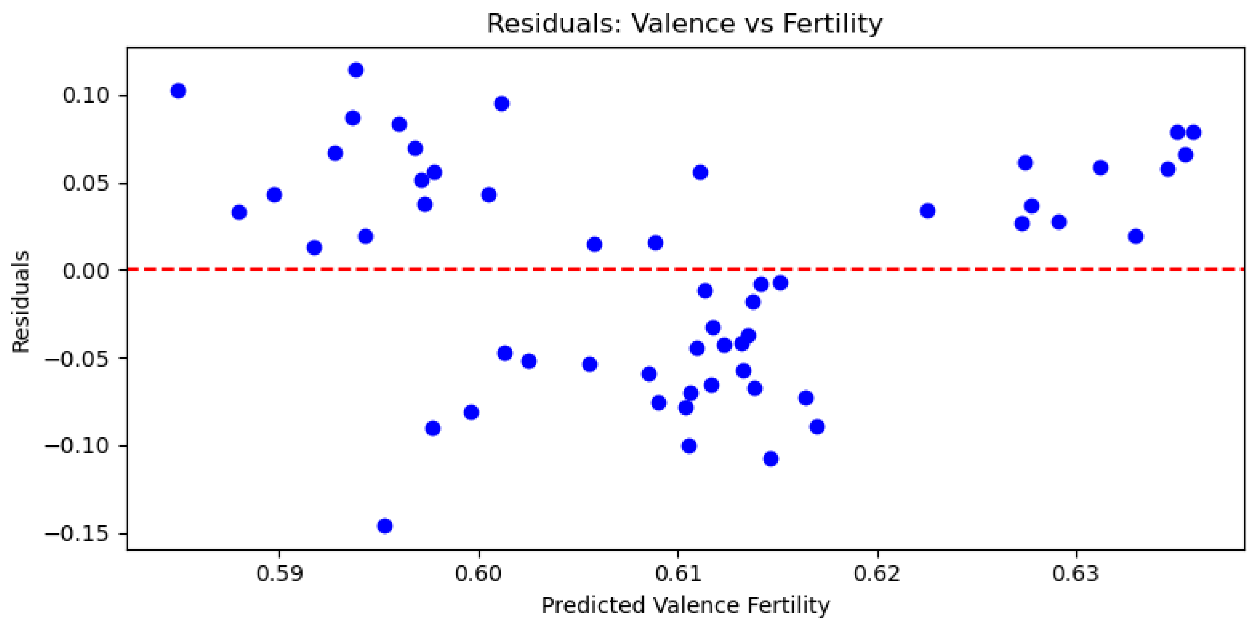
In [22]:
```
# if lambda is 1, then no transformation is needed. If it is 0, then log transformation.
# Else, x^{lambda}-1)/lamda transformation.
print(f'lambda value: {lambda_value}')
print(transformed_data)
```

```
lambda value: -4.743605571021893
[0.21033848 0.21022649 0.21009818 0.20995088 0.20948807 0.20898322
 0.2083612  0.20788527 0.2078398  0.2079737  0.20645821 0.20312502
 0.20022998 0.19897136 0.19691206 0.19549253 0.19749157 0.19637978
 0.19810894 0.19910811 0.1982414  0.19873911 0.19780469 0.19805884
 0.19924295 0.19904757 0.200041   0.20158323 0.20319715 0.20429181
 0.20400977 0.20374567 0.203295   0.20296896 0.20251712 0.20247723
 0.20237648 0.20292234 0.20307951 0.20390718 0.20348618 0.20331263
 0.20377018 0.20383506 0.20392308 0.20467846 0.20484137 0.20415641
 0.20297825 0.20151503 0.20063436 0.20026995 0.19977793 0.19922807
 0.19851735]
```

In [23]:
```
fig, axs = plt.subplots(2,1, figsize=(8, 8))

#Logged Fertility and Valence
X_fer = transformed_data.reshape(-1,1)
y_fertility = averagefinal["avg_valence"].values
model = LinearRegression().fit(X_fer, y_fertility)
print("Fertility Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
y_pred_fertility = model.predict(X_fer)

axs[0].scatter(X_fer, y_fertility, color="blue")
axs[0].plot(X_fer, y_pred_fertility, color="red", label="Fitted line")
axs[0].set_title("Valence and Fertility")
axs[0].set_xlabel("Fertility Rate")
axs[0].set_ylabel("Valence")
residuals_fertility = y_fertility - y_pred_fertility

axs[1].scatter(y_pred_fertility, residuals_fertility, color="blue")
axs[1].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[1].set_title("Residuals: Valence vs Fertility")
axs[1].set_xlabel("Predicted Valence Fertility")
axs[1].set_ylabel("Residuals")
plt.tight_layout()
```

```
Fertility Coeff: 3.4372
Intercept: -0.0871
```





After the transformation, fan shape pattern is no longer present in the fertility residual. However, at the same time, the residual plot is still not quite random. U shape is still observed. To totally resolve heteroskedasticity, we might need further transformation technique which is not covered in the class. Hence, we would include this in our limitation and proceed with the findings.

In [54]:
```
# add transformed data to the new col
averagefinal['trans_fertility']= transformed_data
print(averagefinal.head())
```

```
   year  recession  election  avg_danceability  avg_valence        gdp  \
0  1961          1         0          0.539533     0.714757  562.20975
1  1962          0         0          0.562227     0.701820  603.92150
2  1963          0         0          0.545594     0.714175  637.45150
3  1964          0         1          0.531146     0.692634  684.46150
4  1965          0         0          0.526300     0.652230  742.29025

   unemployment  fedfundrate  imprisonment  fertility  deathrate  S&P500  \
0      6.691667     1.955000      196453.0      3.620     1298.8    0.27
1      5.566667     2.708333      194886.0      3.461     1323.6   -0.09
2      5.641667     3.178333      194155.0      3.319     1346.3    0.23
3      5.158333     3.496667      192627.0      3.190     1303.8    0.16
4      4.508333     4.075000      189855.0      2.913     1306.5    0.12

   log_deathrate   log_gdp  trans_fertility  interaction  stock_election
0       7.169196  6.331875         0.210338         0.00            0.00
1       7.188111  6.403444         0.210226        -0.00           -0.00
2       7.205115  6.457478         0.210098         0.00            0.00
3       7.173038  6.528632         0.209951         0.16            0.16
4       7.175107  6.609740         0.209488         0.00            0.00
```

Perform cross validation to examine model performance and generalizability.

We perform cross-validation to evaluate how three variables—trans_fertility, deathrate, and fedfundrate—predict avg_valence. Although the residuals of trans_fertility are not fully random, we choose to include it in the model because its residuals have improved compared to the untransformed version. Additionally, we believe that trans_fertility may provide valuable contextual information that could enhance the model's predictive power."

In [25]:
```python
input = ['trans_fertility','deathrate','fedfundrate']
for factor in input:
    X = averagefinal[[factor]]
    y = averagefinal['avg_valence']
    kf = KFold(n_splits = 5)
    results = cross_validate(
        LinearRegression(),X,y,
        scoring='neg_root_mean_squared_error',
        cv=kf,return_train_score=True)
    print(f'Train and test RMSE values for {factor}:')
    print('Mean Train RMSE:',round(np.mean(-1*results['train_score']),4))
    print('Mean Test RMSE:', round(np.mean(-1*results['test_score']),4))
    print('Std Train RMSE:',round(np.std(-1*results['train_score']),4))
    print('Std Test RMSE:',round(np.std(-1*results['test_score']),4))
    print('Normalized Test RMSE:',round(
        np.mean(-1*results['test_score'])/(
            averagefinal['avg_valence'].max() - averagefinal['avg_valence'].min()),4))
    print(' ')
```

```
Train and test RMSE values for trans_fertility:
Mean Train RMSE: 0.0585
Mean Test RMSE: 0.1049
Std Train RMSE: 0.0057
Std Test RMSE: 0.0437
Normalized Test RMSE: 0.3951

Train and test RMSE values for deathrate:
Mean Train RMSE: 0.0336
Mean Test RMSE: 0.0441
Std Train RMSE: 0.0008
Std Test RMSE: 0.0078
Normalized Test RMSE: 0.1662

Train and test RMSE values for fedfundrate:
Mean Train RMSE: 0.055
Mean Test RMSE: 0.067
Std Train RMSE: 0.0067
Std Test RMSE: 0.0278
Normalized Test RMSE: 0.2524
```

Based on the RMSE values above, we can conclude:

- `Fertility` : The model shows inconsistent performance across split with the fertility data, likely due to non-random residuals. This is evident from the difference in mean RMSE values between the training set (0.0585) and test set (0.1049), as well as the higher variability in standard deviations (0.0057 for training and 0.0437 for test). These differences indicate that the fertility data introduces high variability, making it an unreliable predictor for the model. Other model (not linear regression) may be more suitable to find the relationship between the 2 variables.
- `DeathRate` : The model performs consistently across splits for the death rate data, as evidenced by the similar mean RMSE values of 0.325 for the training set and 0.038 for the test set. The low standard deviations of 0.0011 for the training set and 0.0035 for the test set suggest low variability between training and test sets, indicating that the death rate data is well-suited for the model.
- `fedfundrate` :The model demonstrates consistent performance for the fedfundrate data, with similar mean RMSE values of 0.055 for the training set and 0.067 for the test set. The standard deviations (0.0067 for training and 0.0278 for test) show relatively low variability, but there is slightly more variation compared to the death rate data. Nonetheless, the model is still performing reasonably well with this feature.

Overall: Fertility shows to be an unreliable predictor, and we will therefore exclude fertility from the model. We will focus on the death rate and fedfundrate data, as they demonstrate more consistent and reliable performance.

OLS Regression to evaluate significance

In [26]:
```
input = ['log_deathrate','fedfundrate']
for factor in input:
    X = averagefinal[[factor]]
    y = averagefinal['avg_valence']
    X_with_constant = sm.add_constant(X)
    model = sm.OLS(y,X_with_constant)
    results = model.fit()
    print(results.summary())
    print(' ')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            avg_valence   R-squared:                       0.745
Model:                            OLS   Adj. R-squared:                  0.740
Method:                 Least Squares   F-statistic:                     154.9
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           2.34e-17
Time:                        19:55:28   Log-Likelihood:                 109.73
No. Observations:                  55   AIC:                            -215.5
Df Residuals:                      53   BIC:                            -211.5
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const            -1.4612      0.166     -8.778      0.000      -1.795      -1.127
log_deathrate     0.3010      0.024     12.445      0.000       0.252       0.349
==============================================================================
Omnibus:                        1.059   Durbin-Watson:                   1.181
Prob(Omnibus):                  0.589   Jarque-Bera (JB):                0.414
Skew:                           0.095   Prob(JB):                        0.813
Kurtosis:                       3.380   Cond. No.                         259.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

                            OLS Regression Results
==============================================================================
Dep. Variable:            avg_valence   R-squared:                       0.231
Model:                            OLS   Adj. R-squared:                  0.216
Method:                 Least Squares   F-statistic:                     15.88
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           0.000207
Time:                        19:55:28   Log-Likelihood:                 79.356
No. Observations:                  55   AIC:                            -154.7
Df Residuals:                      53   BIC:                            -150.7
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.5632      0.014     39.999      0.000       0.535       0.591
fedfundrate     0.0088      0.002      3.985      0.000       0.004       0.013
==============================================================================
Omnibus:                        1.122   Durbin-Watson:                   0.454
Prob(Omnibus):                  0.571   Jarque-Bera (JB):                1.166
Skew:                           0.306   Prob(JB):                        0.558
Kurtosis:                       2.633   Cond. No.                         11.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Evaluation of Significance

- `DeathRate (log)`:
  - Deathrate coefficient (log): Every 1% increase in death rate of a year with result in 0.003010 unit of increase in a avg popular song's valence in that particular year.
  - P-value: The p-value for deathrate is 0.000, indicating that this model is significant.
  - Null hypothesis is rejected.
- `fedfundrate`:
  - fedfundrate coefficient: Every unit increase in fedfundrate of a year with result in 0.0088 unit of increase in a avg popular song's valence in that particular year.
  - P-value: The p-value for fedfundrate is 0.000, indicating that this model is significant.
  - Null hypothesis is rejected.

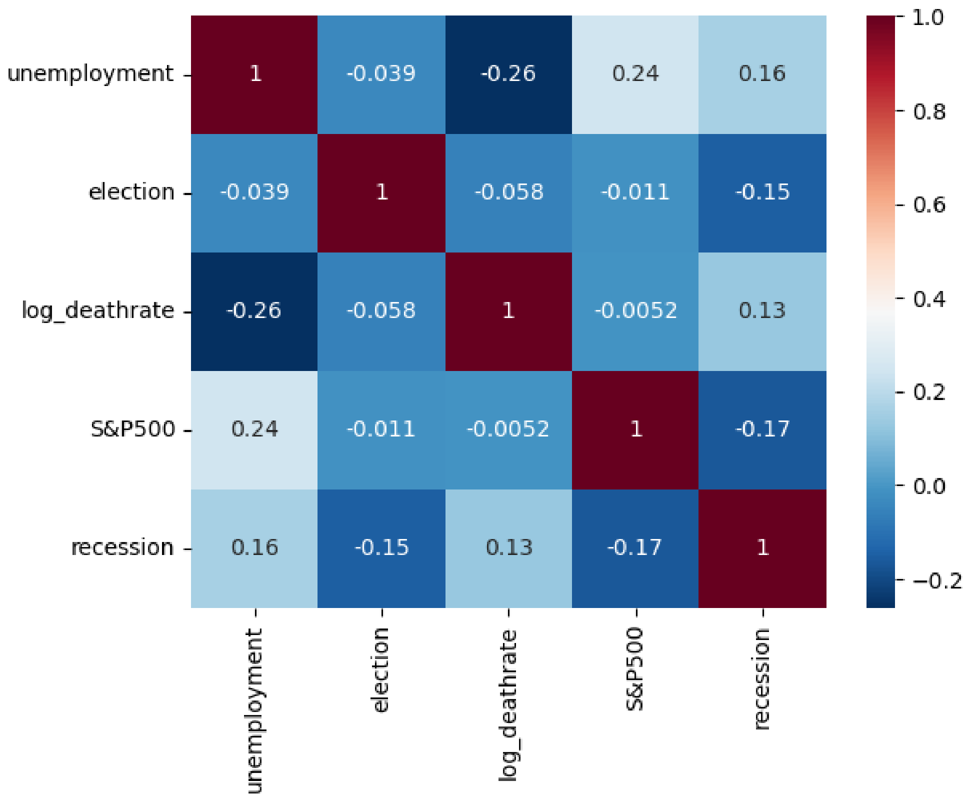
Hypothesis 2: Danceability with Macroeconomic and sociopolitical factors

- H_0: The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) have no relationship with the average danceability of the most popular songs per year from 1961-2016 in the US. ($\beta_1, \beta_2, \beta_3, \beta_4, \beta_5 = 0$)
- H_A: The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) has relationship with the average danceability of the most popular songs per year from 1961-2016 in the US. (at least one $\beta_n \neq 0$ for n=1,2,3,4,5)

Note: **Bonferroni correction** was done to the alpha value since we have multiple independent variables. Our original alpha value 0.05 is divided by 5 since we have 5 independent variables. Hence our adjusted alpha value is 0.01. Reject null hypothesis if any β has p-value less than 0.01.

We create a correlation matrix to check for collinearity between variables, focusing on higher values:

In [27]:
```
avg_final_2= averagefinal[["unemployment", "election", "log_deathrate", "S&P500", "recession"]]
heatmap = sns.heatmap(avg_final_2.corr(numeric_only=True), cmap="RdBu_r", annot=True)
```



No high correlation is spotted between the variables. Hence, we will proceed with training the data.

Modeling

Running initial regression and gauging coefficient and intercepts.

In [28]:
```
input_vars = ["unemployment", "election", "log_deathrate", "S&P500", "recession"]
X = averagefinal[input_vars]
y = averagefinal["avg_danceability"]
X = sm.add_constant(X)
model = LinearRegression().fit(X, y)
```

```
for var_name, var_coef in zip(input_vars, model.coef_):
    print(f"{var_name}: {var_coef:.2f}")
print(f"Intercept: {model.intercept_:.2f}")
```

```
unemployment: 0.00
election: -0.00
log_deathrate: 0.00
S&P500: -0.18
recession: -0.01
Intercept: 1.89
```

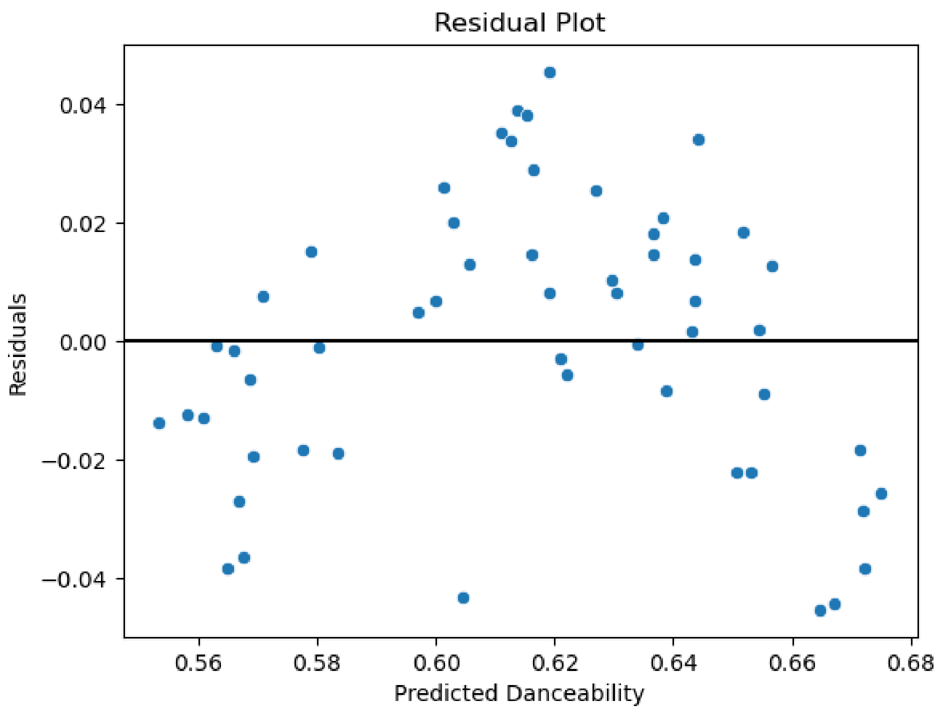
Residual Plot

We calculate residuals (the difference between actual and predicted valence) to assess the goodness of fit of the regression model. A residual plot is generated to visualize the pattern of residuals against predicted values. This step is crucial for checking model assumptions like homoskedasticity (constant variance of errors) and identifying potential model issues like nonlinearity or outliers.

In [29]:
```
train_predictions = model.predict(X)
residuals = averagefinal["avg_danceability"] - train_predictions
residuals.head()
```

Out[29]:
```
0   -0.013852
1   -0.000777
2   -0.012549
3   -0.036531
4   -0.038564
Name: avg_danceability, dtype: float64
```

In [30]:
```
def generate_residual_plot(pred, resid):
    sns.scatterplot(x= pred, y=resid, marker="o")
    plt.axhline(y=0 ,color="black")
    plt.xlabel("Predicted Danceability")
    plt.ylabel("Residuals")
    plt.title("Residual Plot")
    plt.show()
generate_residual_plot(train_predictions, residuals)
```



Based on the residual plot, no fan shape is noticed. However, there is a quadratic pattern. We tried to square the independent variables but it didn't work. Since it is out of scope we covered in class, we will proceed with our findings. More details in limitations.

Perform cross validation to examine model performance and generalizability.

In [31]:
```
kf = KFold(n_splits = 5)
results = cross_validate(
    LinearRegression(),X,y,
    scoring='neg_root_mean_squared_error',
    cv=kf,return_train_score=True)
results_df = pd.DataFrame({
    'Train_RMSE':np.round(-1*results['train_score'],4),
    'Test_RMSE':np.round(-1*results['test_score'],4)})
print('Table summarizing train and test RMSE values across splits:')
print(results_df)
print('Mean Train RMSE:',round(np.mean(results_df['Train_RMSE']),4))
print('Mean Test RMSE:', round(np.mean(results_df['Test_RMSE']),4))
print('Std Train RMSE:',round(np.std(results_df['Train_RMSE']),4))
print('Std Test RMSE:',round(np.std(results_df['Test_RMSE']),4))
print('Normalized Mean Test RMSE:',round(
    np.mean(results_df['Test_RMSE'])/(averagefinal['avg_danceability'].max() -
                                  averagefinal['avg_danceability'].min()),4))
```

```
Table summarizing train and test RMSE values across splits:
   Train_RMSE  Test_RMSE
0      0.0200     0.0559
1      0.0233     0.0238
2      0.0208     0.0354
3      0.0242     0.0225
4      0.0167     0.0576
Mean Train RMSE: 0.021
Mean Test RMSE: 0.039
Std Train RMSE: 0.0026
Std Test RMSE: 0.0152
Normalized Mean Test RMSE: 0.2569
```

Based on the RMSE values above, we can conclude:

- The train RMSE values are overall relatively low, indicating that the model fits the training data well. The Mean Train RMSE of 0.021 suggests that on average the model's predictions are within a reasonable range.
- The test RMSE values are also overall relatively low and similar to the train RMSE values. The Mean Test RMSE of 0.039 is also approximately close to the mean train RMSE of 0.021, meaning we can assume reasonable generalization to unseen data.
- The standard deviation of the Train RMSE values is 0.0026, indicating a relatively low variability between training sets.
- The standard deviation of the Test RMSE values is 0.0152 showing higher variability between the test sets than the training sets. This indicates that the model's ability to generalize to unseen data is slightly less consistent, but it is usually expected that a model's test RMSE values will have higher variation than its train RMSE values.

In [32]:
```
mae = mean_absolute_error(y, model.predict(X))
print(f'MAE: {round(mae,5)}')
```

```
MAE: 0.01902
```

Again, RMSE is greater than MAE, which confirms the model is reliable.

OLS Regression to evaluate significance

Using statsmodels, we run an Ordinary Least Squares (OLS) regression to validate the linear regression results and provide more detailed statistical output, including p-values . This step ensures the statistical robustness of our findings and identifies which predictors are significant contributors to the model.

In [33]:
```
input_vars = ["unemployment", "election", "log_deathrate", "S&P500", "recession"]
X = averagefinal[input_vars]  # The independent variables (features)
y = averagefinal["avg_danceability"]  # The dependent variable (target)
```

```
X = sm.add_constant(X)
est = sm.OLS(y, X).fit()
print('Multivar OLS Regression for Danceability:')

print(est.summary())
```

```
Multivar OLS Regression for Danceability:
                            OLS Regression Results                            
==============================================================================
Dep. Variable:       avg_danceability   R-squared:                       0.699
Model:                            OLS   Adj. R-squared:                  0.668
Method:                 Least Squares   F-statistic:                     22.74
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           1.00e-11
Time:                        19:55:32   Log-Likelihood:                 129.16
No. Observations:                  55   AIC:                            -246.3
Df Residuals:                      49   BIC:                            -234.3
Df Model:                           5                                         
Covariance Type:            nonrobust                                         
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             1.8879      0.133     14.159      0.000       1.620       2.156
unemployment     -0.0005      0.002     -0.234      0.816      -0.005       0.004
election          0.0030      0.008      0.391      0.698      -0.012       0.018
log_deathrate    -0.1840      0.019     -9.821      0.000      -0.222      -0.146
S&P500           -0.0053      0.022     -0.245      0.807      -0.049       0.038
recession        -0.0106      0.008     -1.306      0.198      -0.027       0.006
==============================================================================
Omnibus:                        1.991   Durbin-Watson:                   0.806
Prob(Omnibus):                  0.370   Jarque-Bera (JB):                1.404
Skew:                          -0.152   Prob(JB):                        0.496
Kurtosis:                       2.279   Cond. No.                         379.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Evaluation of significance:

- **Interpretation of coefficients**:
  - `Unemployment Rate` : For a one unit increase in Unemployment Rate, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.0005 units.
  - `election` : When a country is having an election, all else equal, we expect the avg danceability of a popular song in a particular year to increase by 0.0030 units.
  - `Death rate (log)` : For a 1% increase in Death Rate, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.00184 units.
  - `S&P500` : For a one unit increase in the S&P 500 index, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.0053 units.
  - `Recession` : When a country is having a recession, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.0106 units.
- **Interpretation of p value**:
  - **Bonferroni correction** needs to be performed as there are multiple independent variables. Alpha value which is 0.05 is divided by 5 which is equal to 0.01.
  - Only log_deathrate has a p-value less than the alpha value 0.01 which is 0.000. Hence, we can conclude that the model is statistical significance and that we can reject the null hypothesis for pre-registration statement of hypothesis 2.
  - Additionally, unemployment rate, election, S&P 500 and recession have a p-value greater than the alpha value 0.01 which are 0.816, 0.698 and 0.807 and 0.198 respectively. This suggests that the coefficient of these variables are not statistically significant.

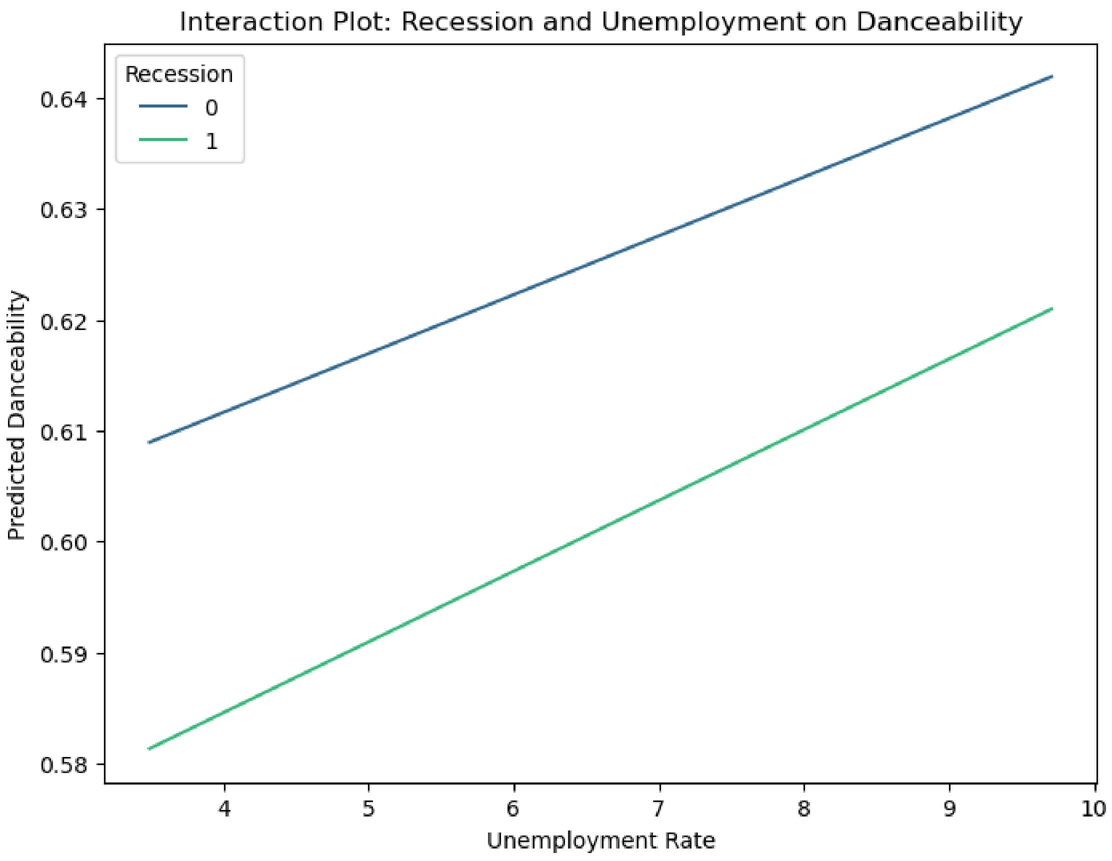
- The model for hypothesis 2 is overall statistically significant in predicting danceability with an extremely low p-value associated with the F-statistic of 1e-11. Null hypothesis is rejected.

Expanding Hypothesis 2

From the heatmap, we can see that collinearity between unemployment and recession is not an issue, as their correlation is only 0.15. Since recessions often bring heightened anxiety, uncertainty, and other negative sentiments, they may amplify the effect of unemployment on people's music preferences (such as effect on music danceability). This led us to hypothesize that there might be a relationship between unemployment and recession that influences their combined impact on danceability. To test this, we included an interaction term between unemployment and recession in the model, allowing us to explore whether the effect of unemployment on danceability depends on whether the economy is in a recession.

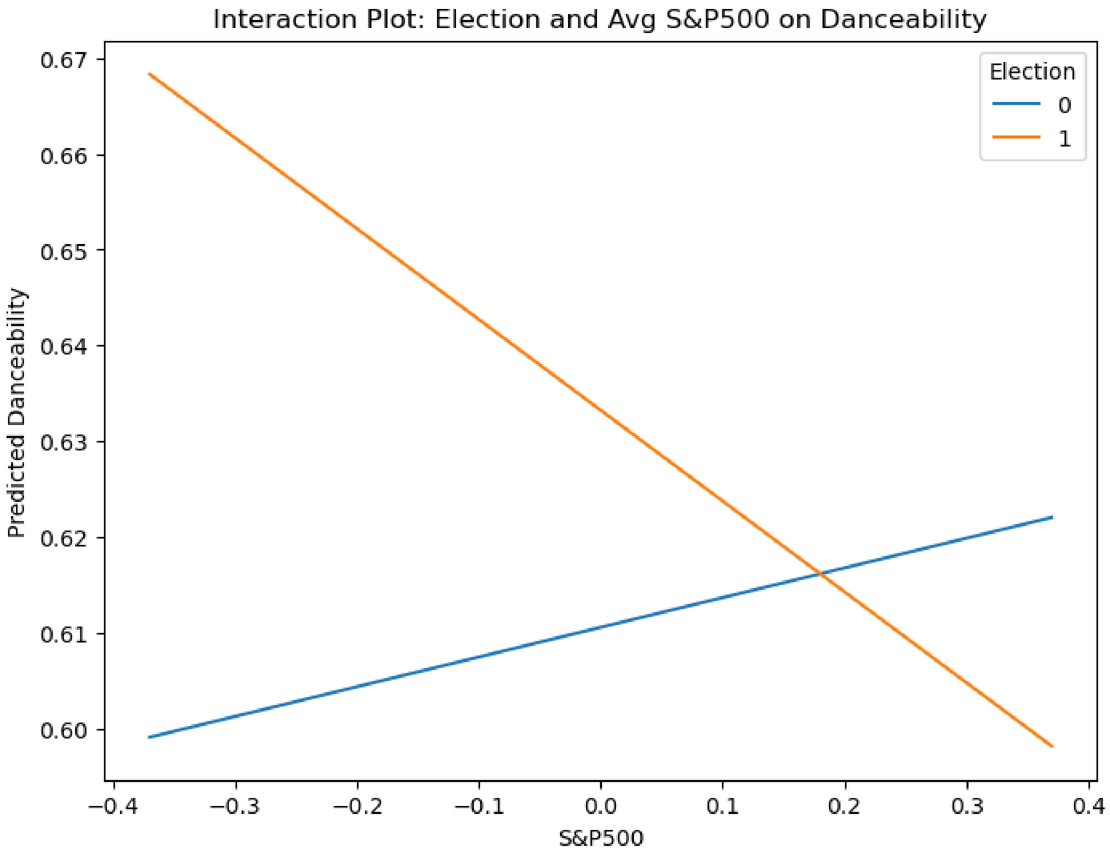
However, the interaction plot showed no crossing lines, indicating that the effect of unemployment on danceability does not significantly differ between recession and non-recession years.

In [34]:
```
averagefinal['interaction'] = averagefinal['unemployment'] * averagefinal['recession']
X = averagefinal[['unemployment', 'recession', 'interaction']]
y = averagefinal['avg_danceability']
X = sm.add_constant(X)
interaction_model = sm.OLS(y, X).fit()
recession_levels = [0, 1]
unemployment_range = np.linspace(averagefinal['unemployment'].min(), averagefinal['unemployment'].m

plot_data = pd.DataFrame({
    'unemployment': np.tile(unemployment_range, len(recession_levels)),
    'recession': np.repeat(recession_levels, len(unemployment_range))
})
plot_data['interaction'] = plot_data['unemployment'] * plot_data['recession']
plot_data = sm.add_constant(plot_data)
plot_data['predicted_danceability'] = interaction_model.predict(plot_data)
plt.figure(figsize=(8, 6))

sns.lineplot(
    data=plot_data,
    x='unemployment',  # Use avg_unemployment for the x-axis
    y='predicted_danceability',  # Predicted valence as the y-axis
    hue='recession',  # Recession levels as different lines
    palette="viridis"  # Optional: Adjust color scheme
)

plt.title("Interaction Plot: Recession and Unemployment on Danceability")
plt.xlabel("Unemployment Rate")
plt.ylabel("Predicted Danceability")
plt.legend(title="Recession")
plt.show()
```



We hypothesize that S&P500 stock returns may have varying effects on danceability depending on whether or not it is an election year, drawing on domain knowledge. Elections often bring a lot of political and economic uncertainty, which could change how people's music preferences respond to stock market changes. For instance, during election years, people might react more strongly to market shifts, while in non-election years, the reactions might be less intense. To explore this idea, we created an interaction plot to show how predicted danceability changes with stock returns in election years compared to non-election years. This helps us see if the effect of stock returns on danceability stays the same or changes between these two scenarios.

In [35]:
```
averagefinal['interaction'] = averagefinal['S&P500'] * averagefinal['election']
X = averagefinal[['S&P500', 'election', 'interaction']]
y = averagefinal['avg_danceability']
X = sm.add_constant(X)
interaction_model = sm.OLS(y, X).fit()

election_levels = [0, 1]
stockreturn_range = np.linspace(averagefinal['S&P500'].min(), averagefinal['S&P500'].max(), 100)
plot_data = pd.DataFrame({
    'S&P500': np.tile(stockreturn_range, len(election_levels)),
    'election': np.repeat(election_levels, len(stockreturn_range))
})
plot_data['interaction'] = plot_data['S&P500'] * plot_data['election']
plot_data = sm.add_constant(plot_data)
plot_data['predicted_danceability'] = interaction_model.predict(plot_data)

plt.figure(figsize=(8, 6))
sns.lineplot(
    data=plot_data,
    x='S&P500',
    y='predicted_danceability',
    hue='election'
)
plt.title("Interaction Plot: Election and Avg S&P500 on Danceability")
plt.xlabel('S&P500')
plt.ylabel("Predicted Danceability")
plt.legend(title="Election")
plt.show()
```



As shown in the interaction plot above, the lines representing average stock returns during election years and non-election years intersect, suggesting that stock returns may influence public sentiment (danceability) differently depending on whether it is an election year. To test this, we included an interaction term in the OLS multivariable regression model to examine if the effect was statistically significant. Our hypothesis was that a significant interaction term would indicate a different impact of stock returns on danceability during election years. However, the regression output showed a p-value for the interaction term (stock_election) greater than 0.025. This result suggests that the interaction between stock returns and election years does not significantly influence danceability. Therefore, the stock_election term is not necessary and should be excluded from the final model.

Update Hypothesis 2 model

In [36]:
```
averagefinal['stock_election'] = averagefinal['S&P500'] * averagefinal['election']
Xmar = averagefinal[['unemployment', 'log_deathrate',
                     'recession', 'stock_election']]
ymar = averagefinal['avg_danceability']
Xmar = sm.add_constant(Xmar)
est_interaction = sm.OLS(ymar, Xmar).fit()

print('Multivar OLS Regression for danceability:')
print(est_interaction.summary())
```

```
Multivar OLS Regression for danceability:
                            OLS Regression Results
==============================================================================
Dep. Variable:       avg_danceability   R-squared:                       0.698
Model:                            OLS   Adj. R-squared:                  0.673
Method:                 Least Squares   F-statistic:                     28.84
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           1.90e-12
Time:                        19:55:33   Log-Likelihood:                 129.05
No. Observations:                  55   AIC:                            -248.1
Df Residuals:                      50   BIC:                            -238.1
Df Model:                           4
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              1.8898      0.134     14.071      0.000       1.620       2.160
unemployment      -0.0007      0.002     -0.300      0.766      -0.005       0.004
log_deathrate     -0.1841      0.019     -9.737      0.000      -0.222      -0.146
recession         -0.0109      0.008     -1.363      0.179      -0.027       0.005
stock_election    -0.0064      0.037     -0.174      0.863      -0.080       0.067
==============================================================================
Omnibus:                        1.678   Durbin-Watson:                   0.852
Prob(Omnibus):                  0.432   Jarque-Bera (JB):                1.294
Skew:                          -0.159   Prob(JB):                        0.524
Kurtosis:                       2.320   Cond. No.                         385.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Evaluation of significance:

- **Interpretation of coefficients**:
  - `Unemployment Rate` : For a one unit increase in Unemployment Rate, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.0007 unit.
  - `Death rate (log)` : For a 1% increase in Death Rate, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.001841 unit.
  - `stock_election` : For one unit increase in the interaction term stock_election, all else equal, the predicted value of danceability deceases by 0.0064 units.
  - `Recession` : When a country is having recession, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.0109 unit.
- **Interpretation of p value**:
  - **Bonferroni correction** needs to be performed as there are multiple independent variables. Alpha value which is 0.05 is divided by 4 which is equal to 0.0125.
  - Only log_deathrate have a p-value less than the alpha value 0.0125 which are 0.000. Hence, we can conclude that the model is statistical significance and that we can reject the null hypothesis for preregistration hypothesis 2.
  - Additionally, unemployment, recession and stock_election have a p-value greater than the alpha value 0.0125 which are 0.766, 0.179 and 0.863 respectively. This suggest that the coef of these variables are are not satistically significant.
  - The model for hypothesis 2 is overall statistically significant in predicting danceability with an extremely low p-value associated with the F-statistic of 1.90e-12. Null hypothesis is rejected.

Conclusion

Hypothesis 1: As GDP increases, the average valence of popular songs in a given year decreases.

For this hypothesis, we analyzed data by year, running a linear regression with GDP per capita as the input and the average valence (a measure of positivity in music) of popular songs in that year as the output. Using a single-sided t-test (we interpret it by dividing the p value by 2) and a significance level (α) of 5%, we found that the relationship between GDP per capita and average song valence was statistically significant, with a negative coefficient (B<0). Specifically, our model estimates that every unit increase in GDP per capita causes the average valence of popular songs in that year to decrease by 0.05 units. This result is reasonable given that the valence scale ranges from 0 to 1.

This finding aligns with our expectations and provides an intriguing perspective on how economic factors may influence cultural outputs such as music. A plausible explanation for this trend could be that as GDP per capita increases, societies may experience greater emotional complexity or stressors tied to wealth and modernity, leading to a preference for or production of less upbeat and more introspective or melancholic music.

To expand our understanding, we examined how other economic and social factors, such as fertility, death rates and federal funds rates, might impact song valence. We found that both death rates and federal funds rates variables also had statistically significant relationships with valence:

- `Death Rate`: Every 1% increase in death rate of a year with result in 0.003010 unit of increase in a avg popular song's valence in that particular year. This strong positive correlation, with a p-value of 0.000, suggests that higher death rates may drive the production or popularity of more positive or uplifting music, perhaps as a response to collective grief or hardship.
- `Federal Funds Rate`: For every unit increase in the federal funds rate during a given year, the average valence of popular songs increased by 0.0088 units. With a p-value of 0.000, this suggests a subtler but still significant association, potentially reflecting the influence of economic policy and financial conditions on music sentiment.

Our linear regression model is not suitable to study the relationship between fertility and avg valence as the data remains heteroskedastic after log and boxcox transformation and there is a significant difference between RSME within the train and test data.

Overall,these findings illustrate the complex ways in which economic and social factors can influence the tone and emotional content of popular music, with GDP per capita driving a decrease in valence, while death rates and federal funds rates correspond to increases.

Hyphothesis 2: Macroeconomic and sociopolitical factors influence the average danceability of popular songs in a given year.

For this hypothesis, we analyzed data using a multivariate regression model with the average danceability of popular songs as the output variable and macroeconomic and sociopolitical factor such as unemployment rate, election occurrence, death rate, S&P 500 index, and recession status, as input variables.

Based on the results of the regression analysis, we found that for a 1% increase in Death Rate, all else equal, we expect the avg danceability of a popular song in a particular year to decrease by 0.00184 unit. The p-value for this variable was 0.000, indicating that this relationship is statistically significant at the aplha value 0.01 after Bonferroni correction. This suggests that higher death rates may lead to less danceable music, possibly reflecting collective societal moods or shifts in cultural output during times of hardship.

In contrast, the coefficients for unemployment rate, election, S&P 500 index, and recession status were not statistically significant at the alpha level 0.01. Their respective p-values - 0.816, 0.698, 0.807, and 0.198—were all greater than 0.01, indicating insufficient evidence to conclude that these variables influence danceability. For example, while the model suggests that danceability decreases slightly during recessions (−0.0106) and increases slightly during election years (+0.0030), these effects were not statistically significant.

Despite the limited significance of individual predictors, the overall model for hypothesis 2 was statistically significant in predicting danceability, as indicated by an extremely low p-value associated with the F-statistic (p=1e−11). Null hypothesis is rejected. The macroeconomic and sociopolitical factors (unemployment rate, election indicator, death rate, S&P 500, and recession) has relationship with the average danceability of the most popular songs per year from 1961-2016 in the US. This suggests that, collectively, the variables included in the model provide meaningful insights into variations in song danceability across years.

These results provide support for our hypothesis, with the death rate emerging as the most impactful factor on danceability among the variables tested.

Inclusion of Interaction Term:

As observed in the interaction plot above, there is a noticeable intersection between the lines representing average stock returns during election years and non-election years. This suggests that stock returns might influence public sentiment (and, by extension, song danceability) differently depending on whether it is an election year or not. The interaction term (stock_election) was included in the model to determine whether this effect is statistically significant.

The results of the updated regression are as follows:

The coefficients for unemployment, recession and the stock_election interaction term were not statistically significant at the adjusted alpha level.

Significant Variables

- `Death Rate (log-transformed)`: For a 1% increase in the death rate, all else equal, the average danceability of popular songs is expected to decrease by 0.001841 units. This relationship is statistically significant (p=0.000) at the adjusted alpha level of 0.0125.

Special Note on Death rate
This caught our attention because death rate was tested to be significant for both valence and danceability. However, as shown, a 1% increase in death rate, all else equal, increases average valence by 0.003010 units and decreases average danceability by 0.001841 units. A plausible explanation could be that when people pass away, people want to consume more positive, calm songs that provide healing to the mourning but at the same time, not the positive type that people dance to. Although our model stops here, the death rate's effect on music would be an interesting point to further explore.

The overall model, including the interaction term, was statistically significant in predicting danceability, with an extremely low p-value associated with the F-statistic (1.90e-12).

These results suggest that while the interaction term provides insights into potential differences in the effect of stock returns during election years, it was not statistically significant. The death rate (log-transformed) emerged as the most impactful predictors of danceability in this model.

Application to the Real World The above research or similar projects allow policymakers and economists to gauge societal sentiment during economic shifts, while the music industry can better anticipate and respond to audience preferences. For example, when there are higher death rates or federal fund rates, artists might want to consider producing more happier and positive songs to foster healing in the population. For sociologists and cultural analysts, the study reveals how collective experiences shape creative outputs, enriching our understanding of the dynamic relationship between economics and culture. This connection underscores the broader societal impact of economic trends, making the research relevant across disciplines.

Limitations

Limitation of Yearly Aggregation

Our data is aggregated by year rather than month and year due to limitations in the availability of social variables such as birth rate, death rate, and incarceration rate on a monthly basis. This temporal limitation significantly impacts our ability to account for **lagging variables**, meaning that we cannot assess the delayed impact of economic or social events on music trends. For instance, we are unable to analyze how a recession in one month may influence the music released or consumed in subsequent months. Similarly, this yearly aggregation restricts us from capturing the effects of short-term, time-sensitive events, such as elections or other socio-political shifts, which may play a substantial role in shaping music trends. As a result, our findings may overlook nuanced relationships between social variables and music, leading to broader and less precise conclusions.

Limited Data Points

One limitation of our analysis is the decision to average song valence and danceability values per year to ensure that each independent variable corresponds to a single dependent value during model training. While this approach addresses issues like using a constant GDP value to predict a wide range of music valence values, it comes at the cost of losing data granularity and reducing the number of data points. This trade-off may impact the model's accuracy in predicting the relationship between the variables.

Imperfect Representation of Human Emotions

The valence and danceability metrics from Spotify data, while valuable proxies, do not comprehensively encapsulate the complexity of human emotions. Valence measures perceived positivity of a track, and danceability reflects rhythm and beat, but neither metric can fully capture nuanced emotional responses, such as melancholy or nostalgia, that music can evoke. This oversimplification may lead to discrepancies between the analyzed data and the actual emotional experiences of listeners. Consequently, conclusions drawn about how social and economic factors influence music sentiment may miss subtleties and fail to accurately reflect the broader spectrum of human emotional expression.

Limited Representativeness of Billboard and Spotify Data

The reliance on Billboard’s top charts and Spotify data presents another significant limitation in our study. These platforms, while widely recognized, are not fully representative of the diverse listening habits of the US population. Billboard charts primarily reflect popular music consumption, which may disproportionately highlight mainstream trends, while Spotify data is limited to users of the platform and does not account for listeners on other platforms such as Apple Music, YouTube, or traditional radio. This lack of inclusivity could introduce biases, excluding niche or underrepresented genres and demographics, and ultimately skew our results to favor the preferences of certain groups rather than the population as a whole.

Hypothesis 1: Heteroskedastic Residual

In our first hypothesis, we observed a heteroskedastic residual plot between fertility and valence. Initially, we applied a log transformation, but the pattern remained. Next, we used a Box-Cox transformation, which successfully reduced heteroskedasticity and improved the residual appearance compared to the untransformed fertility. However, the residuals now exhibit a quadratic pattern. While we understand that addressing this issue is essential for ensuring the robustness of our regression analysis, we also realized that fixing the residual requires more advanced techniques outside the scope of this class.

Hypothesis 2: Quadratic Residual

In our second regression analysis, we noticed a U-shaped pattern in the residual plot of our danceability. Such a quadratic pattern indicates heteroskedasticity. Common remedies, of which we've tried, include transforming the independent variables, by squaring each values (quadratic transformation), to stabilize variance. However, this transformation is ineffective. We need for more advanced techniques which are not yet covered in our coursework. These could involve applying weighted least squares regression or using heteroskedasticity-consistent standard errors to obtain reliable estimates despite the presence of heteroskedasticity as suggest by the internet. Addressing this issue is crucial for ensuring the robustness and reliability of our regression analyses.

Fertility VS Average Valence

In expanding hypothesis 1, a linear regression model is trained to find the relationship between fertility rate and average song’s valence. However, we spotted fan shape in residual plot which indicates heteroscedasticity. Log transformation was performed but it doesn't work. Nonetheless, we found out that by using boxcox transformation the problem was solved. However, after looking at the huge difference we have for RMSE between train and test data, we concluded that a linear regression model is not suitable to find the relationship between these 2 variables.

Acknowledgements and Bibliography

1. Federal Reserve Bank of St. Louis. "NBER Based Recession Indicators for the United States from the Period Following the Peak through the Trough (USREC)." *FRED*, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/series/USREC.
2. DonSolare. "Valence as a Measure of Happiness." *Spotify Community*, 11 Feb. 2018, https://community.spotify.com/t5/Spotify-for-Developers/Valence-as-a-measure-of-happiness/td-p/4385221?utm_source=chatgpt.com.
3. Hessing, Ted. "Box-Cox Transformation." *Six Sigma Study Guide*, https://sixsigmastudyguide.com/box-cox-transformation/.
4. Prudential Financial. "Glossary of Key Economic and Stock Market Terms." *Prudential Financial*, https://www.prudential.com/financial-education/glossary-of-economic-and-stock-market-terms.
5. Matplotlib. "Tight Layout Guide." Matplotlib, 5 Dec. 2024, https://matplotlib.org/stable/users/explain/axes/tight_layout_guide.html
6. Stack Overflow. "Seaborn Plot with Second Y-Axis." *Stack Overflow*, 12 Apr. 2019, https://stackoverflow.com/questions/55654500/seaborn-plot-with-second-y-axis.
7. Statology. "What Is a Good RMSE?" Statology, 23 Sept. 2020, https://www.statology.org/what-is-a-good-rmse#:~:text=Normalizing%20the%20RMSE%20Value&text=This%20produces%20a%20value%20between,is%20
8. Waskom, Michael L. "Choosing Color Palettes." *Seaborn 0.13.2 Documentation*, 2 Dec. 2024, https://seaborn.pydata.org/tutorial/color_palettes.html.
9. "How to Adjust the Size of Heatmaps in Seaborn." *Scales*, https://scales.arabpsychology.com/stats/how-to-adjust-the-size-of-heatmaps-in-seaborn/

Appendix

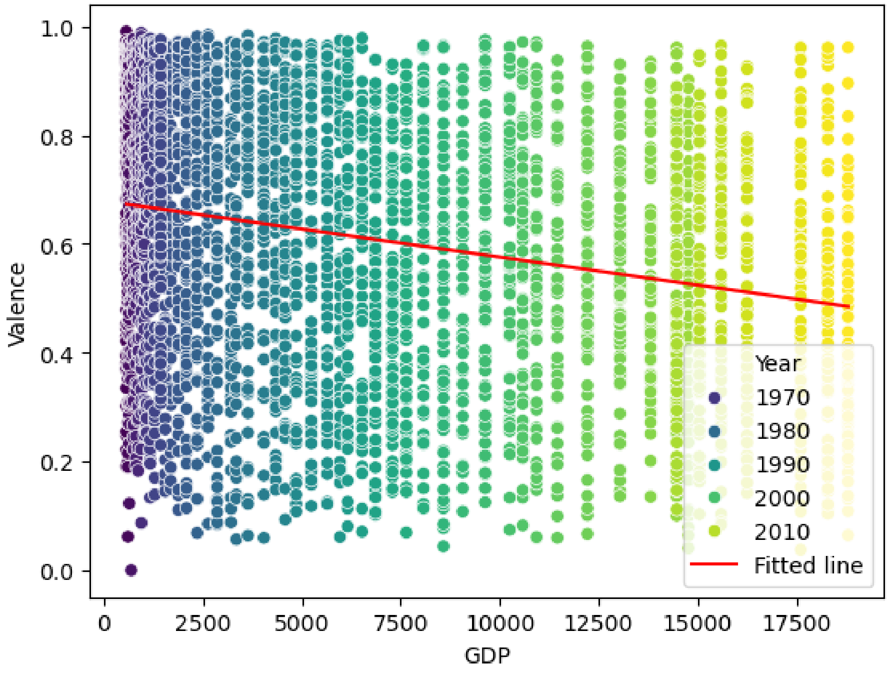
As mentioned earlier, our first approach is to use all data points for each song from 1961 to 2016. Based on the 1st iteration of hypothesis 1, we conclude that a model that uses one value to predict a range of data will not be robust. For instance, in the first regression visualization, we can see that there is only one GDP value for all the popular songs per year. However, with that constant GDP value, there are a wide range of song valence values from 0 to 1. Hence, we decide to average the valence and danceability of songs per year eventually. This will ensure that our model takes it one value for each independent variable and predicts one value for the dependent variable. On the other hand, an obvious disadvantage to the average method we have performed above was that we lost hundreds of data points. Thus, in the appendix below, we will run an analysis on our data without averaging and see if any findings prove to be of value.

Modeling and Visualization

In [37]:
```
# GDP and Valence
X_gdp = final[["US_GDP"]].values
y_gdp = final["valence"].values
model = LinearRegression().fit(X_gdp, y_gdp)
y_pred_gdp = model.predict(X_gdp)
print("GDP Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")
```

```
sns.scatterplot(data=final, x="US_GDP", y="valence", hue="year", palette="viridis")
sns.lineplot(x=final["US_GDP"], y=y_pred_gdp, color="red", label="Fitted line")
plt.xlabel("GDP")
plt.ylabel("Valence")
plt.legend(loc="lower right", title="Year")
plt.show()
```

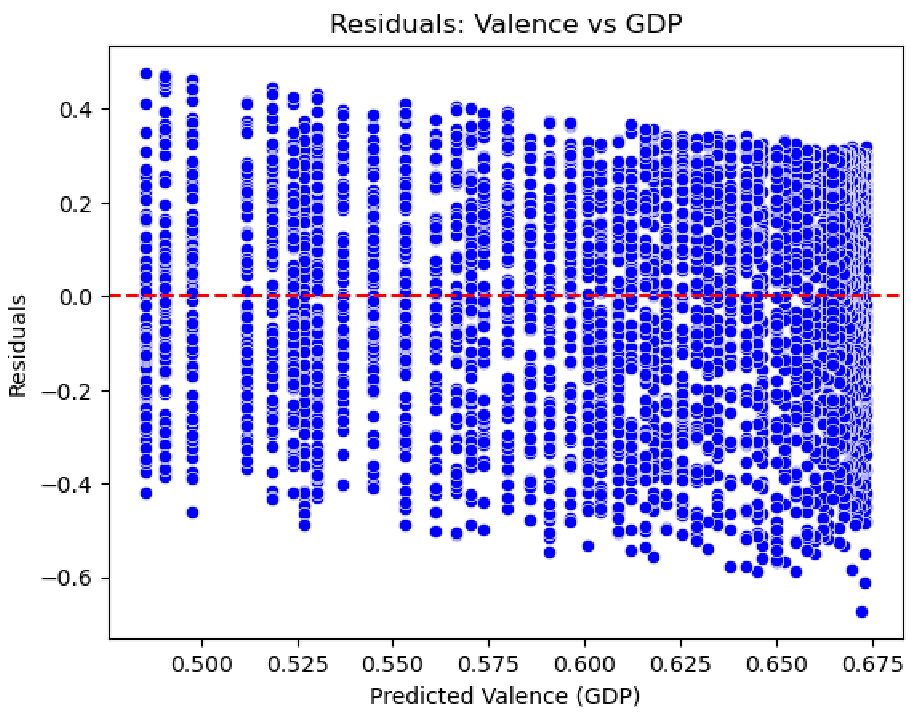
```
GDP Coeff: -0.0
Intercept: 0.6793
```



Residual Plot

Residual is plotted to check if there is any heteroskedasticity and if so, log transformation will be performed. In this case, no heteroskedasticity is noticed.

In [38]:
```
# GDP and Valence
X_gdp = final[["US_GDP"]].values
y_gdp = final["valence"].values
model = LinearRegression().fit(X_gdp, y_gdp)
y_pred_gdp = model.predict(X_gdp)
residuals_gdp = y_gdp - y_pred_gdp

sns.scatterplot(data= final,x= y_pred_gdp, y= residuals_gdp, color="blue")
plt.axhline(y=0, color='red', linestyle='--')  # Add a line at zero
plt.title("Residuals: Valence vs GDP")
plt.xlabel("Predicted Valence (GDP)")
plt.ylabel("Residuals")
plt.show()
```



Perform cross validation to examine model performance and generalizability.

```
In [39]: X = final[['US_GDP']]
         y = final['valence']
         kf = KFold(n_splits = 5)
         results = cross_validate(
             LinearRegression(),X,y,
             scoring='neg_root_mean_squared_error',
             cv=kf,return_train_score=True)
         results_df = pd.DataFrame({
             'Train_RMSE':np.round(-1*results['train_score'],4),
             'Test_RMSE':np.round(-1*results['test_score'],4)})
         print('Table summarizing train and test RMSE values across splits:')
         print(results_df)
         print('Mean Train RMSE:',round(np.mean(results_df['Train_RMSE']),4))
         print('Mean Test RMSE:', round(np.mean(results_df['Test_RMSE']),4))
         print('Std Train RMSE:',round(np.std(results_df['Train_RMSE']),4))
         print('Std Test RMSE:',round(np.std(results_df['Test_RMSE']),4))
         print('Normalized Test RMSE:',round(
             np.mean(results_df['Test_RMSE'])/(final['valence'].max() -
                                               final['valence'].min()),4))
```

```
Table summarizing train and test RMSE values across splits:
   Train_RMSE  Test_RMSE
0      0.2353     0.2235
1      0.2290     0.2479
2      0.2314     0.2392
3      0.2337     0.2296
4      0.2347     0.2266
Mean Train RMSE: 0.2328
Mean Test RMSE: 0.2334
Std Train RMSE: 0.0023
Std Test RMSE: 0.009
Normalized Test RMSE: 0.2355
```

Based on the RMSE values above, we can conclude:

- The model seems to have consistent performance across different splits, as indicated by the similar train and test RMSE values for each split. The RMSE values are contained in a small range, from 0.2263-0.2385.
- The mean train RMSE value is relatively lower than the mean test RMSE. This indicates that in general the model perform better in the training model. The mean test RMSE value of 0.2353 suggests that on average, the

model's predictions are within a reasonable range of the actual values in the test data.

- Looking across splits, the test RMSE values are relatively close to the training RMSE values. Additionally, the train and test RMSE means across splits are approximately equal, with the mean train RMSE at 0.2337 and the mean test RMSE at 0.2353. Thus, we can assume reasonable generalization to unseen data.
- The standard deviation of the Train RMSE is 0.0019, showing relatively low variability in the model's predictions between training sets.
- The standard deviation of the Test RMSE is 0.0068, suggesting higher variability compared to the training set. Therefore, we can conclude that the model's ability to generalize to unseen data is less consistent. However, based on the fact that that the mean train RMSE and mean test RMSE are similar, and that the test standard deviation is still relatively low, we can still assume reasonable generalization.
- We decided to calculate and analyze the normalized RMSE value for hypothesis 1 to gain a better idea of if this RMSE value was “good.” Based on an article we found about normalizing RMSE, we calculated the normalized RMSE to get a value between 0 and 1 where values closer to 0 are considered “better” results. This is helpful because RMSE values are sometimes hard to compare because of unit inconsistencies and the range of the dataset itself. From this normalization, we got an RMSE value of 0.2374, which is relatively close to 0 and thus suggests the regression model fits the test data well.

In [40]:
```
mae = mean_absolute_error(y, model.predict(X))
print(f'MAE: {round(mae,5)}')
```

```
MAE: 0.19743
/Users/kuankhaixin/anaconda3/envs/info2950/lib/python3.11/site-packages/sklearn/base.py:486: UserWa
rning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

RMSE greater than MAE. Model is reliable

OLS regression for significance valuation

In [41]:
```
X = final[["US_GDP"]]
y = final["valence"]

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                valence   R-squared:                       0.057
Model:                            OLS   Adj. R-squared:                  0.057
Method:                 Least Squares   F-statistic:                     239.1
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           2.12e-52
Time:                        19:55:38   Log-Likelihood:                 150.50
No. Observations:                3922   AIC:                            -297.0
Df Residuals:                    3920   BIC:                            -284.5
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.6793      0.006    112.573      0.000       0.668       0.691
US_GDP     -1.032e-05   6.67e-07    -15.463      0.000   -1.16e-05   -9.01e-06
==============================================================================
Omnibus:                      474.781   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              197.545
Skew:                          -0.355   Prob(JB):                     1.27e-43
Kurtosis:                       2.161   Cond. No.                     1.47e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary

- Our alpha value is 5%. However, since the statistics model we used by default gives double sided t test, we divide the p value by half. After dividing by half, the p value remains 0.000 which shows that it is statistically significant, B<0
- Every unit increase in gdp will cause average valence of popular songs in the year to decrease by -1.032e-05 unit.

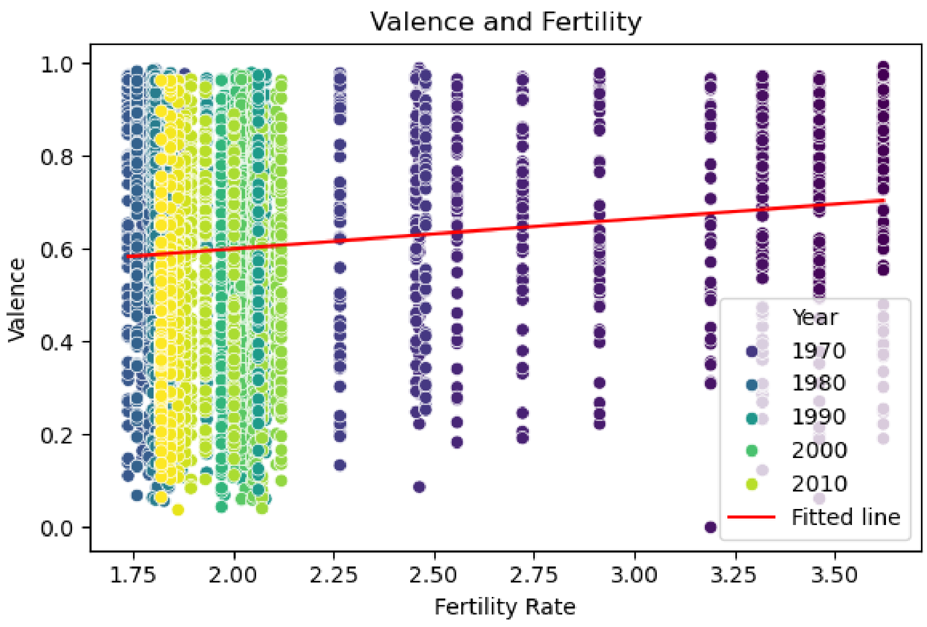
Expanding Hypothesis 1

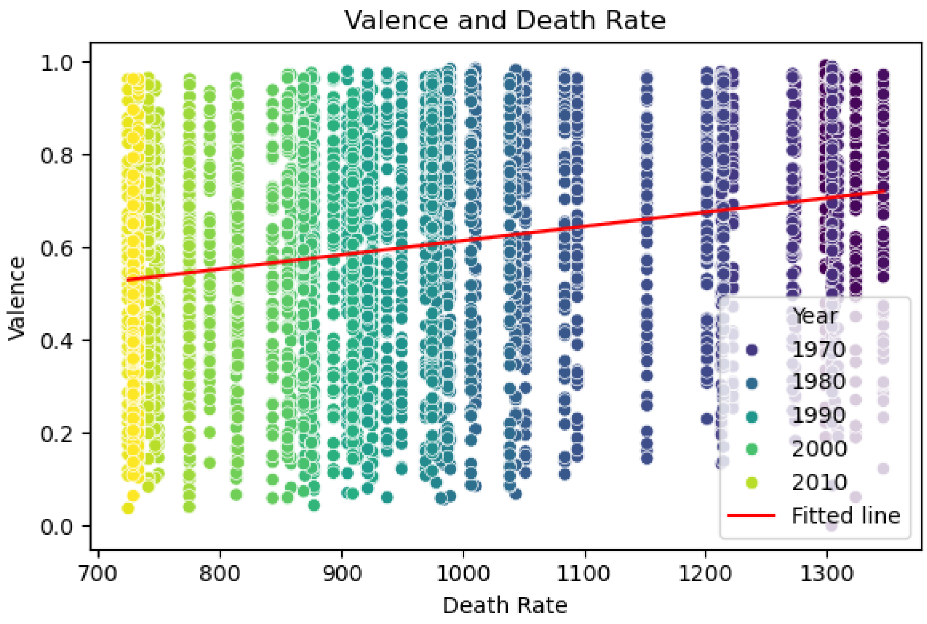
Visualization

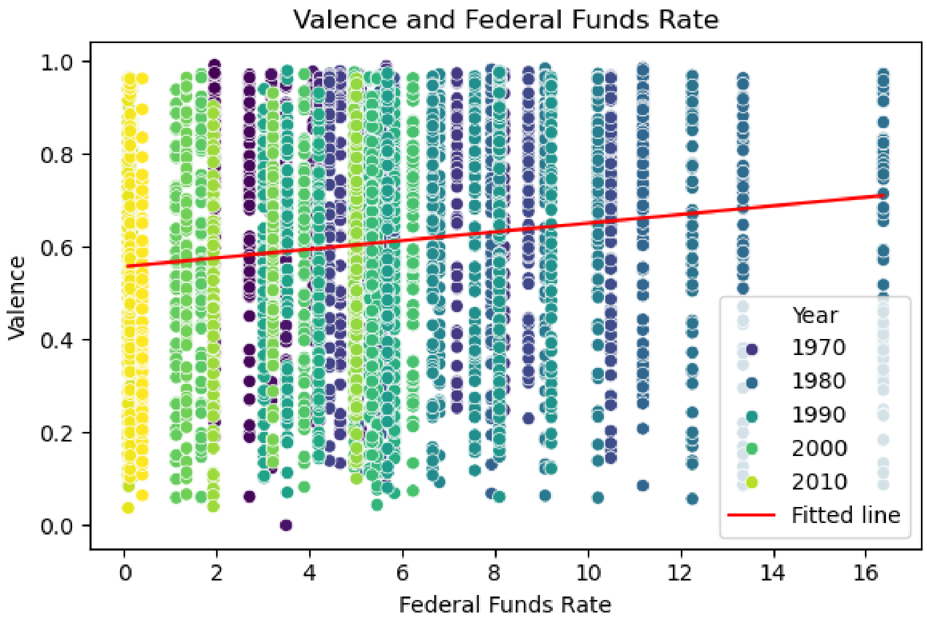
In [42]:
```python
# Create a 2x2 grid of subplots
fig, axs = plt.subplots(3, figsize=(6, 12))  # Adjust figsize for better spacing

#Fertility and Valence
X_fertility = final[["fertility"]].values
y_fertility = final["valence"].values
model = LinearRegression().fit(X_fertility, y_fertility)
y_pred_fertility = model.predict(X_fertility)
print("Fertility Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")

sns.scatterplot(data=final, x="fertility", y="valence", hue="year", \
                palette="viridis", ax=axs[0])
sns.lineplot(x=final["fertility"], y=y_pred_fertility, color="red", \
             label="Fitted line", ax=axs[0])
axs[0].set_title("Valence and Fertility")
axs[0].set_xlabel("Fertility Rate")
axs[0].set_ylabel("Valence")
axs[0].legend(loc="lower right", title="Year")

#Death Rate and Valence
X_deathrate = final[["DeathRate"]].values
y_deathrate = final["valence"].values
model = LinearRegression().fit(X_deathrate, y_deathrate)
y_pred_deathrate = model.predict(X_deathrate)
```

```
print("Death Rate Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")

sns.scatterplot(data=final, x="DeathRate", y="valence", hue="year",
                palette="viridis", ax=axs[1])
sns.lineplot(x=final["DeathRate"], y=y_pred_deathrate, color="red",
             label="Fitted line", ax=axs[1])
axs[1].set_title("Valence and Death Rate")
axs[1].set_xlabel("Death Rate")
axs[1].set_ylabel("Valence")
axs[1].legend(loc="lower right", title="Year")

#Federal Funds Rate and Valence
X_fedfundrate = final[["fedfundrate"]].values
y_fedfundrate = final["valence"].values
model = LinearRegression().fit(X_fedfundrate, y_fedfundrate)
y_pred_fedfundrate = model.predict(X_fedfundrate)
print("Fed Funds Rate Coeff:", round(model.coef_[0], 4))
print(f"Intercept: {model.intercept_:.4f}")

sns.scatterplot(data=final, x="fedfundrate", y="valence", hue="year", \
                palette="viridis", ax=axs[2])
sns.lineplot(x=final["fedfundrate"], y=y_pred_fedfundrate, color="red",
             label="Fitted line", ax=axs[2])
axs[2].set_title("Valence and Federal Funds Rate")
axs[2].set_xlabel("Federal Funds Rate")
axs[2].set_ylabel("Valence")
axs[2].legend(loc="lower right", title="Year")
plt.tight_layout()
plt.show()
```

```
Fertility Coeff: 0.0642
Intercept: 0.4714
Death Rate Coeff: 0.0003
Intercept: 0.3080
Fed Funds Rate Coeff: 0.0093
Intercept: 0.5572
```


Valence and Fertility
Valence
Fertility Rate
1.0
0.8
0.6
0.4
0.2
0.0
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
Year
1970
1980
1990
2000
2010
Fitted line

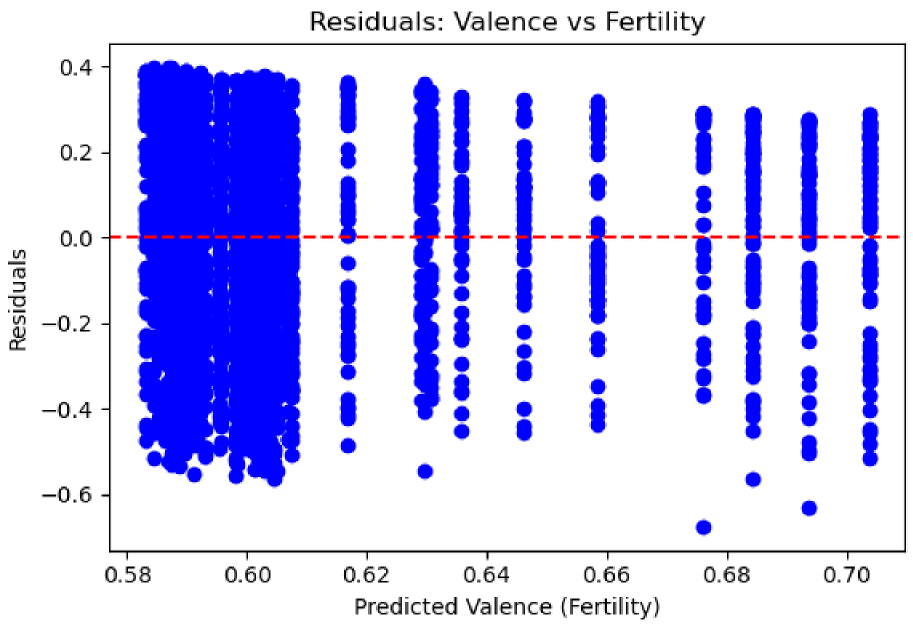

Valence and Death Rate
Valence
Death Rate
1.0
0.8
0.6
0.4
0.2
0.0
700
800
900
1000
1100
1200
1300
Year
1970
1980
1990
2000
2010
Fitted line

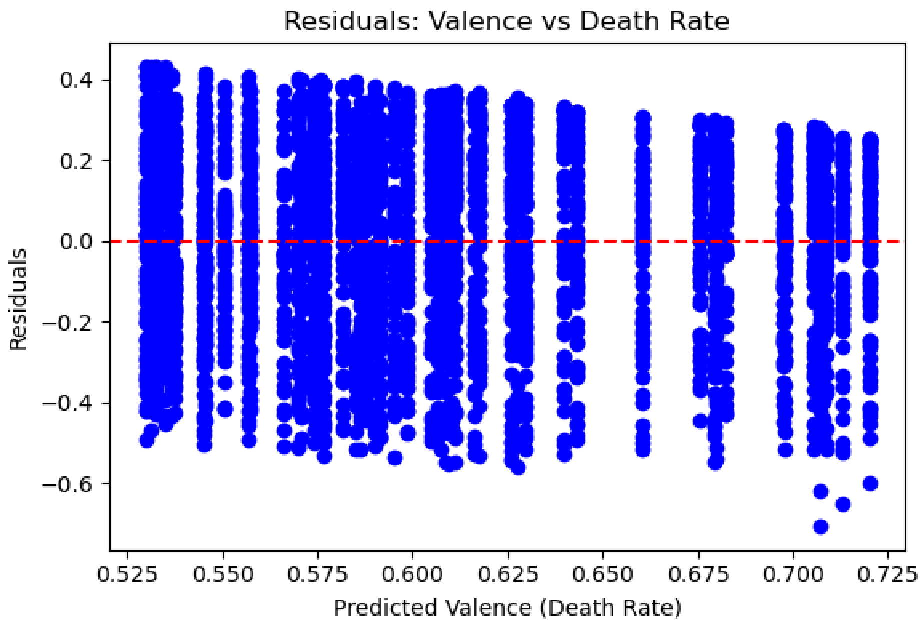

Valence and Federal Funds Rate
Valence
Federal Funds Rate
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
16
Year
1970
1980
1990
2000
2010
Fitted line

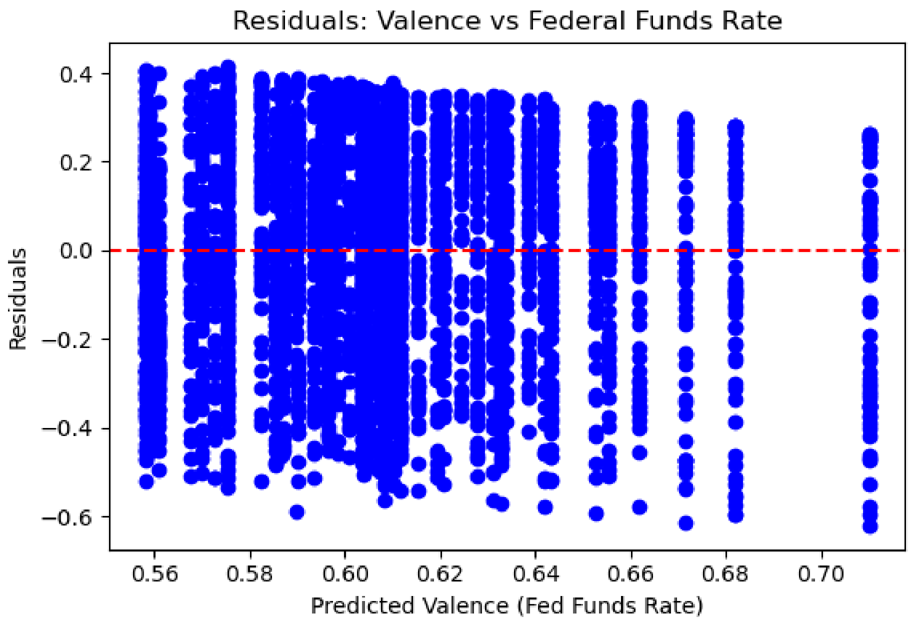
Residual Plot

Residual plot to check the presence of heteroskedasticity. No heteroskedastic data is spotted.

In [43]:
```
fig, axs = plt.subplots(3, figsize=(6, 12))

# Fertility and Valence
X_fertility = final[["fertility"]].values
y_fertility = final["valence"].values
model = LinearRegression().fit(X_fertility, y_fertility)
y_pred_fertility = model.predict(X_fertility)
residuals_fertility = y_fertility - y_pred_fertility

axs[0].scatter(y_pred_fertility, residuals_fertility, color="blue")
axs[0].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[0].set_title("Residuals: Valence vs Fertility")
axs[0].set_xlabel("Predicted Valence (Fertility)")
axs[0].set_ylabel("Residuals")

# Death Rate and Valence
X_deathrate = final[["DeathRate"]].values
y_deathrate = final["valence"].values
model = LinearRegression().fit(X_deathrate, y_deathrate)
y_pred_deathrate = model.predict(X_deathrate)
residuals_deathrate = y_deathrate - y_pred_deathrate

axs[1].scatter(y_pred_deathrate, residuals_deathrate, color="blue")
axs[1].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[1].set_title("Residuals: Valence vs Death Rate")
axs[1].set_xlabel("Predicted Valence (Death Rate)")
axs[1].set_ylabel("Residuals")

# Federal Funds Rate and Valence
X_fedfundrate = final[["fedfundrate"]].values
y_fedfundrate = final["valence"].values
model = LinearRegression().fit(X_fedfundrate, y_fedfundrate)
y_pred_fedfundrate = model.predict(X_fedfundrate)
residuals_fedfundrate = y_fedfundrate - y_pred_fedfundrate

axs[2].scatter(y_pred_fedfundrate, residuals_fedfundrate, color="blue")
axs[2].axhline(y=0, color='red', linestyle='--')  # Add a line at zero
axs[2].set_title("Residuals: Valence vs Federal Funds Rate")
axs[2].set_xlabel("Predicted Valence (Fed Funds Rate)")
axs[2].set_ylabel("Residuals")
plt.tight_layout()
```


Residuals: Valence vs Fertility
Residuals
Predicted Valence (Fertility)
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.58
0.60
0.62
0.64
0.66
0.68
0.70


Residuals: Valence vs Death Rate
Residuals
Predicted Valence (Death Rate)
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.525
0.550
0.575
0.600
0.625
0.650
0.675
0.700
0.725


Residuals: Valence vs Federal Funds Rate
Residuals
Predicted Valence (Fed Funds Rate)
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.56
0.58
0.60
0.62
0.64
0.66
0.68
0.70

Perform cross validation to examine model performance and generalizability.

In [44]:
```
input = ['fertility','DeathRate','fedfundrate']
for factor in input:
    X = final[[factor]]
    y = final['valence']
    kf = KFold(n_splits = 5)
    results = cross_validate(
        LinearRegression(),X,y,
        scoring='neg_root_mean_squared_error',
        cv=kf,return_train_score=True)
    print(f'Train and test RMSE values for {factor}:')
    print('Mean Train RMSE:',round(np.mean(-1*results['train_score']),4))
    print('Mean Test RMSE:', round(np.mean(-1*results['test_score']),4))
    print('Std Train RMSE:',round(np.std(-1*results['train_score']),4))
    print('Std Test RMSE:',round(np.std(-1*results['test_score']),4))
    print('Normalized Test RMSE:',round(
        np.mean(-1*results['test_score'])/(
            final['valence'].max() - final['valence'].min()),4))
    print(' ')
```

```
Train and test RMSE values for fertility:
Mean Train RMSE: 0.2374
Mean Test RMSE: 0.2771
Std Train RMSE: 0.0023
Std Test RMSE: 0.065
Normalized Test RMSE: 0.2797

Train and test RMSE values for DeathRate:
Mean Train RMSE: 0.2332
Mean Test RMSE: 0.2349
Std Train RMSE: 0.0025
Std Test RMSE: 0.0088
Normalized Test RMSE: 0.2371

Train and test RMSE values for fedfundrate:
Mean Train RMSE: 0.2371
Mean Test RMSE: 0.2409
Std Train RMSE: 0.0016
Std Test RMSE: 0.0063
Normalized Test RMSE: 0.2431
```

Based on the RMSE values above, we can conclude:

- `Fertility` : The model seems to have consistent performance across different splits for the fertility data, as indicated by the similar mean train and mean test RMSE values of 0.238 and 0.2866, respectively. The low standard deviations of 0.002 for the training sets and 0.083 for the test sets also indicate that there is low variability between training sets and test sets.
- `DeathRate` : The model appears to be very consistent across different splits for the deathrate data with similar mean train and mean test RMSE values of 0.2338 and 0.2347, respectively. The low standard deviations of 0.0019 for the training sets and 0.0075 for the test sets also indicate that there is low variability between training sets and test sets.
- `fedfundrate` : The model seems to have consistent performance across splits for the fedfundrate data, seen by the similar mean train and test RMSE values of 0.2379 and 0.242. The low standard deviation of 0.0015 indicates low variability across training sets, as does the still relatively low value of 0.0062 for the standard deviation across test sets.

Overall, the model seems to perform consistently across these three factors.

OLS Regression to evaluate significance:

In [45]:
```
for factor in input:
    X = final[[factor]]
    y = final['valence']
    X_with_constant = sm.add_constant(X)
    model = sm.OLS(y,X_with_constant)
    results = model.fit()
    print(results.summary())
    print(' ')
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                valence   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     50.77
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           1.23e-12
Time:                        19:55:41   Log-Likelihood:                 59.636
No. Observations:                3922   AIC:                            -115.3
Df Residuals:                    3920   BIC:                            -102.7
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4714      0.019     24.498      0.000       0.434       0.509
fertility      0.0642      0.009      7.125      0.000       0.047       0.082
==============================================================================
Omnibus:                      719.020   Durbin-Watson:                   1.909
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              201.671
Skew:                          -0.292   Prob(JB):                     1.61e-44
Kurtosis:                       2.055   Cond. No.                         13.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

                            OLS Regression Results                            
==============================================================================
Dep. Variable:                valence   R-squared:                       0.054
Model:                            OLS   Adj. R-squared:                  0.054
Method:                 Least Squares   F-statistic:                     223.9
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           2.83e-49
Time:                        19:55:41   Log-Likelihood:                 143.33
No. Observations:                3922   AIC:                            -282.7
Df Residuals:                    3920   BIC:                            -270.1
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3080      0.020     15.206      0.000       0.268       0.348
DeathRate      0.0003   2.05e-05     14.964      0.000       0.000       0.000
==============================================================================
Omnibus:                      551.800   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              195.699
Skew:                          -0.324   Prob(JB):                     3.19e-43
Kurtosis:                       2.119   Cond. No.                     5.38e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.38e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

                            OLS Regression Results                            
==============================================================================
Dep. Variable:                valence   R-squared:                       0.020
Model:                            OLS   Adj. R-squared:                  0.019
Method:                 Least Squares   F-statistic:                     78.83
Date:                Sat, 07 Dec 2024   Prob (F-statistic):           1.01e-18
Time:                        19:55:41   Log-Likelihood:                 73.444
No. Observations:                3922   AIC:                            -142.9
Df Residuals:                    3920   BIC:                            -130.3
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.5572      0.007     83.625      0.000       0.544       0.570
fedfundrate     0.0093      0.001      8.879      0.000       0.007       0.011
```

```
==============================================================================
Omnibus:                      569.348   Durbin-Watson:                   1.923
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              199.722
Skew:                          -0.329   Prob(JB):                     4.28e-44
Kurtosis:                       2.112   Cond. No.                         11.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Evaluation of Significance

- `Fertility`:
  - Fertility coefficient: Every unit increase in fertility rate of a year with result in 0.0642 unit of increase in a popular song's valence in that particular year.
  - P-value: The p-value for fertility is 0.000, indicating that this model is significant.
- `DeathRate`:
  - Deathrate coefficient: Every unit increase in death rate of a year with result in 0.0003 unit of increase in a popular song's valence in that particular year.
  - P-value: The p-value for deathrate is 0.000, indicating that this model is significant.
- `fedfundrate`:
  - fedfundrate coefficient: Every unit increase in fedfundrate of a year with result in 0.0093 unit of increase in a popular song's valence in that particular year.
  - P-value: The p-value for fedfundrate is 0.000, indicating that this model is significant.