THE COMPLEX NETWORKS OF OLD-TIME AND IRISH FIDDLE TUNES

Identification, Classification, and Song Generation

Abstract

Creating complex networks and aggregate graphs from popular Old-Time and Irish music websites, that were then used to predict keys and genres for songs or generate new tunes. Genres were predicted accurately 77.43% of the time, while keys were predicted accurately 93.73% of the time.

Trevarthan, James

jtrevar@siue.edu

1 Introduction

The focus of this project is the identification/classification and generation of songs in two musical genres, Old-Time Fiddle and Irish. These genres were chosen for several reasons, among them being the availability of songs of around the same length for creating aggregate graphs, the author's expertise in both styles, and because no other authors had explored these two particular genres through complex networks. Songs in general are well-suited to be modeled as complex networks, with a node representing a uniquely played pitch (or note), and each directed edge between two nodes representing one note played after another. A small complex network representation of the tune *The Little Beggarman* can be seen in Figure 1.

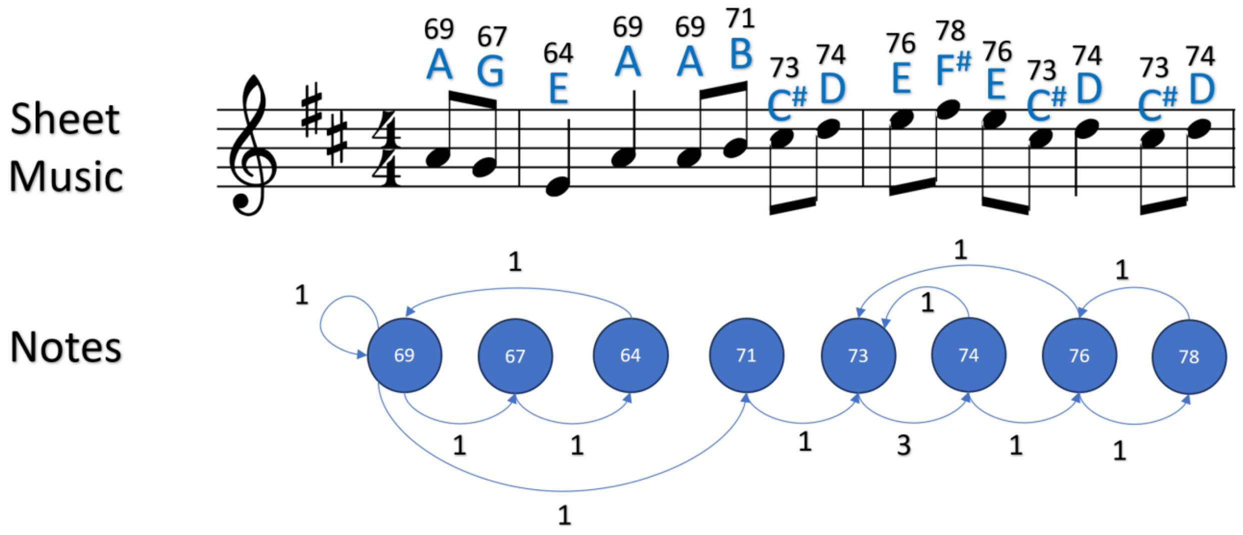


Figure 1 - Complex Network Representation of Notes for "The Little Beggarman"

The edges for these complex networks are weighted, since a specific note being played after another note can happen multiple times in a musical piece. Self-loops are also possible since the same note can be played after another (a B note being played after a B note for example). Each song's notes can be modified to sound slightly different by its choice of key. The keys presented in this paper are A Major, C Major, D Major, and G Major. How these keys appear in sheet music can be seen in Figure 2. The key of C Major has no modifications to notes, while the key of G Major has every F modified to be a sharp (a slightly higher pitch than just a normally played F note). D Major has both F's and C's modified to be sharp, and A Major has sharps on every F, C, and G note. Keys are important for determining how a song will sound overall, and it's important to compare songs with similar keys, which has been overlooked in previous papers.

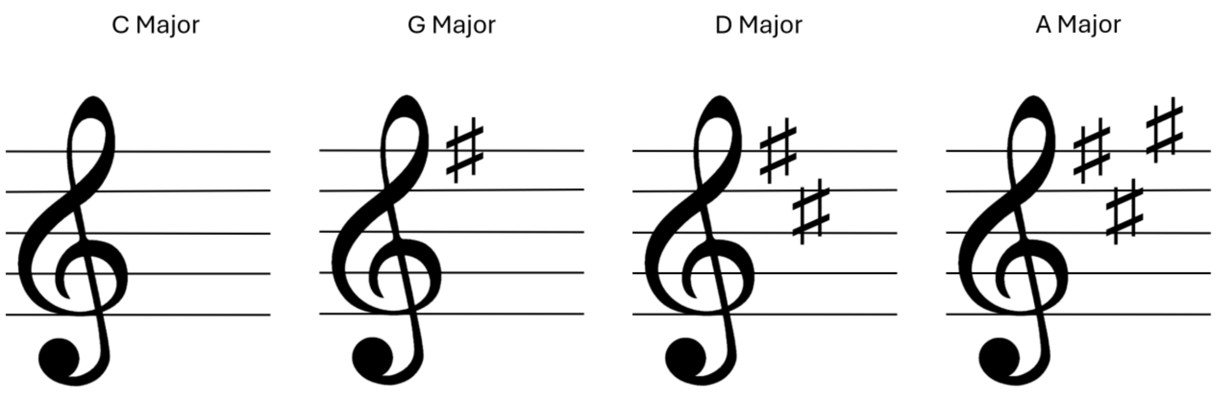


Figure 2 - Musical Keys for Treble Clef (A Major, C Major, D Major, and G Major)

MIDI (Musical Instrument Digital Interface) files play a sizeable role in this paper. These files contain instructions about which note is to be played, for how long, in what key, and so on. This makes them extremely useful as they are easily readable by computers and code with the right modifications. Each possible note that can be played in music is given a MIDI note number, with the MIDI note numbers that appear in Old-Time and Irish music shown in Figure 3. In between most notes, the reader may notice missing numbers (for example between G3[55] and A3[57]). These missing numbers represent sharps for the notes bordering to the left, or flats for the notes bordering to the right. For example, 56 could represent a G3 sharp (a G3 pitched slightly higher) or an A3 flat (an A3 pitched slightly lower).

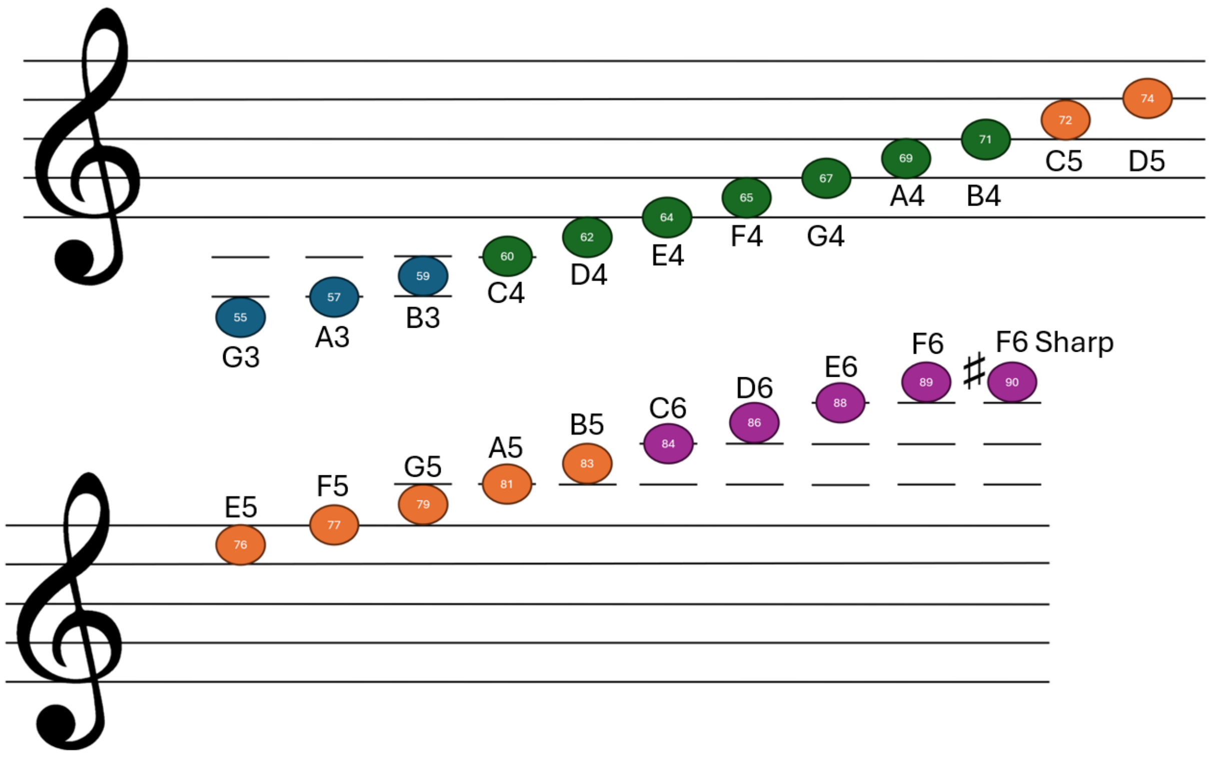


Figure 3 - Common Notes on the Violin for Irish and Fiddle Tunes with Corresponding MIDI Numbers

2 Process

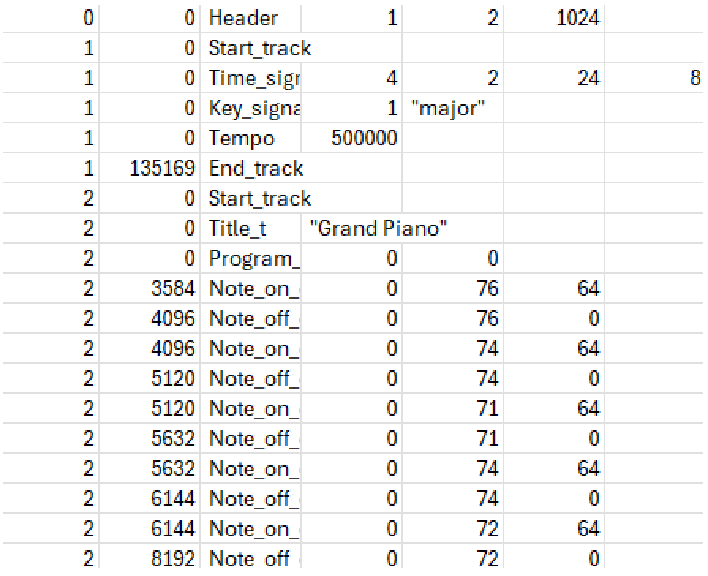
2.1 Acquiring the Data

The songs (MIDI files) used for this project were acquired from two websites. The Old-Time tunes were downloaded from https://www.oldtimefiddletunes.net/ (referred to as OTFT in this paper) and the Irish tunes from https://thesession.org/ (referred to as TS in this paper). The OTFT website had tunes in three formats with those being sheet music, MIDI files, and live musician recordings. The sheet music was discarded as a possible data source due to the unnecessary difficulty of training a convolutional neural network (CNN) to recognize the different notes (though, to increase the dataset size in future works, CNN would be helpful due to a lot of old-time tunes being bound in books and not having accompanying MIDI files). Live recordings were also discarded as a possible data source, since again a machine learning model would have to be trained to pick out the musician's notes, with the musician recordings having differing levels of audio quality and non-violin instruments in the background. For this project's goals, MIDI files were found to be acceptable, especially as they were already in machine readable format. MIDI files were not without their own issues, which will be detailed later. For the TS website the tunes also came in three formats: sheet music, MIDI files, and ABC notation. Sheet music was not used for the same reason as described for OTFT. ABC notation would have been acceptable as a data format since it's also a form of musical notation readable by computers. However, since OTFT did not have songs in ABC notation, and for conciseness in code design, MIDI files were also chosen as the data format for TS songs.

When it came to the distribution of songs downloaded in MIDI format, 557 Old-Time and 557 Irish tunes were downloaded for 1,114 tunes overall. Out of those 557 tunes for both genres, 103 tunes were in A Major (18.5%), 55 in C Major (9.9%), 220 in D Major (39.5%), and 179 in G Major (32.1%). OTFT was the limiting factor in tunes chosen in those specific keys since it had less available songs to choose from (892 fiddle tunes total when the project began, though that number has grown as more songs have been added to the website). OTFT had songs in other keys such as B Flat, F Major, E Minor, A Minor, and many more. However, these keys only had 10 or less songs per key, which wasn't enough to for creating decent aggregate graphs and for prediction of genres and keys. A, C, D, and G Major were found to have the highest number of tunes per key. TS had more possible Irish tunes than OTFT (830 in A Major, 310 in C Major, 2120 in D Major, and 1880 in G Major), so to make for more equal comparisons between Irish and Old-Time, 557 tunes were chosen from TS with the same distribution as OTFT. The MIDI files were downloaded by hand due to convoluted website design hindering the attempted use of web scrapers. A spreadsheet was created containing the song name, filename for the MIDI file, Key, and Artist (for Old-Time tunes only, The Session didn't provide artist information).

2.2 Converting MIDI to CSV

Once the MIDI files had been downloaded for both genres, it was time to convert them into a format that could then be imported into and easily read by a program using Python. Since the main goal of this project was analysis of tune data and networks, and not creating a midi to csv conversion tool, one was found online that achieved this purpose called midiCSV. (Walker, 2008) Figure 4 shows the first few notes from the old-time tune *22nd of February* in CSV format from the original MIDI file. The important information that was utilized in Figure 4 was column 2 which contained the time stamp of the note, column 3 which stated whether a particular note had begun or ended (Note_on_c or Note_off_c), and column 5 that gave the MIDI note number of the note played. All the MIDI files were batch run through the midiCSV conversion tool for old-time and then Irish. There were no complications with this process.



0	0	Header	1	2	1024	
1	0	Start_track				
1	0	Time_sigr	4	2	24	8
1	0	Key_signa	1	"major"		
1	0	Tempo	500000			
1	135169	End_track				
2	0	Start_track				
2	0	Title_t	"Grand Piano"			
2	0	Program_	0	0		
2	3584	Note_on_	0	76	64	
2	4096	Note_off_	0	76	0	
2	4096	Note_on_	0	74	64	
2	5120	Note_off_	0	74	0	
2	5120	Note_on_	0	71	64	
2	5632	Note_off_	0	71	0	
2	5632	Note_on_	0	74	64	
2	6144	Note_off_	0	74	0	
2	6144	Note_on_	0	72	64	
2	8192	Note_off_	0	72	0	

Figure 4 – First Few Lines of the CSV Format of the "22nd of February" Tune MIDI File

2.3 Turning the CSVs into Adjacency Matrices

Now that all the songs were in CSV format, turning these songs into adjacency matrices was the next goal. Using the spreadsheet created in section 2.1, and the columns from Figure 4 in section 2.2, a program called CSVtoGraph.py was written to convert those CSVs into adjacency matrices which would eventually be used to create graphs. It took advantage of the linear pattern of tunes (one note is followed by another note, which is then followed by another note, and so on) and went down the list of notes in the CSV, taking into account when a note turned on and off again, and only after a note had been turned off would it then create a connection between that and the next note turning on, marking that in an adjacency matrix. These markings in the adjacency matrix were additive, so each time a note followed another note, the correspond adjacency matrix cell was incremented by 1. Drones were not included in the adjacency matrices (a drone is when two notes are played at the same time). Drones are usually used to add more flavor to the melody, but the focus of the project was on the core tune's notes and not extra notes included. An example of a completed adjacency matrix for the tune *22nd of February* can be seen in Figure 5. The darker green a square is in this example, the more times that combination of the note on the vertical axis followed by the note on the horizontal axis occurs. The reader may notice that most of the adjacency matrix is filled with red squares (0's). This is due to the fact that most old-time and Irish tunes all take place lower on the fingerboard on the 4 violin strings. There are many notes that can be reached by shifting the hand higher up the E string, but this is generally done more in Classical pieces, and not in the tunes being studied.

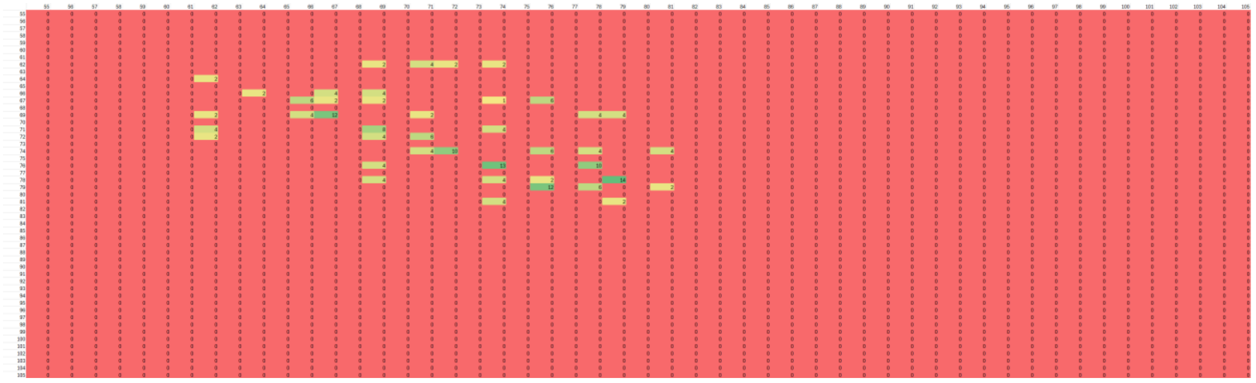


Figure 5 - Adjacency Matrix for the Old-Time tune "22nd of February"

During this process of creating adjacency matrices, several tunes were discarded due to having notes that were lower than MIDI number 55 (or the lowest note on a G string). The old-time songs Boatman and Bonaparte's Retreat were both found to have this low note, and upon further study were discovered to be for violins in DDAD tuning (the G string is tuned down even lower to be a D string). Most musicians don't tune their violin's G string down this low, so these tunes were safely discarded as outliers. As tunes were being turned into adjacency matrices, some information was collected and placed into a spreadsheet such as number of notes, highest note, lowest note, first note in the song, and last note in the song. This allowed the author to find another faulty tune called Blue Eagle, that only had 2 notes total. When listening to the MIDI file, it indeed only had 2 notes before cutting off, a transcription error by whoever uploaded it to the website. It was also safely discarded. Six Irish tunes were discarded during the process because they were found to have notes lower than MIDI note number 55. These tunes (St. Anne's Reel, Father O Gradys Visit to Bocca, Carolan's Concerto, Carrickfergus, Lord Ramsey's, and The Derry Air) all had an underlying piano low gchord track (a piano accompaniment embedded into the MIDI file) that contained notes lower than a violin could play. These tunes were discarded since only MIDI files containing one instrument (The Violin) were wanted for analysis.

A function was created during this step to guarantee that the tunes were being correctly imported. This function was called print_sheet_music, and can be found in ProjectFunctions.py. It outputs a text-based piece of sheet music from the imported song.

This output was compared to the original sheet music on OTFT and TS, allowing confirmation that the tunes had been correctly imported into the adjacency matrices.

2.4 Transforming Adjacency Matrices into Complex Networks

With each tune now having its own adjacency matrix, complex network representations of those adjacency matrices could be created. A program called SongAnalysis.py was written to both turn each adjacency network into graph form, but also to collect data on each unique song's complex network. The networks were created using networkx and drawn in a shell layout, with a few of the tunes' graphs being depicted in Figure 6. A spreadsheet was created in the program containing key data on each tune's network such as song name, key, genre, number of nodes, number of edges, shortest path length, average degree, average clustering, graph density, network diameter, average weighted degree, number of notes, highest note, lowest note, first note played, and last note played.

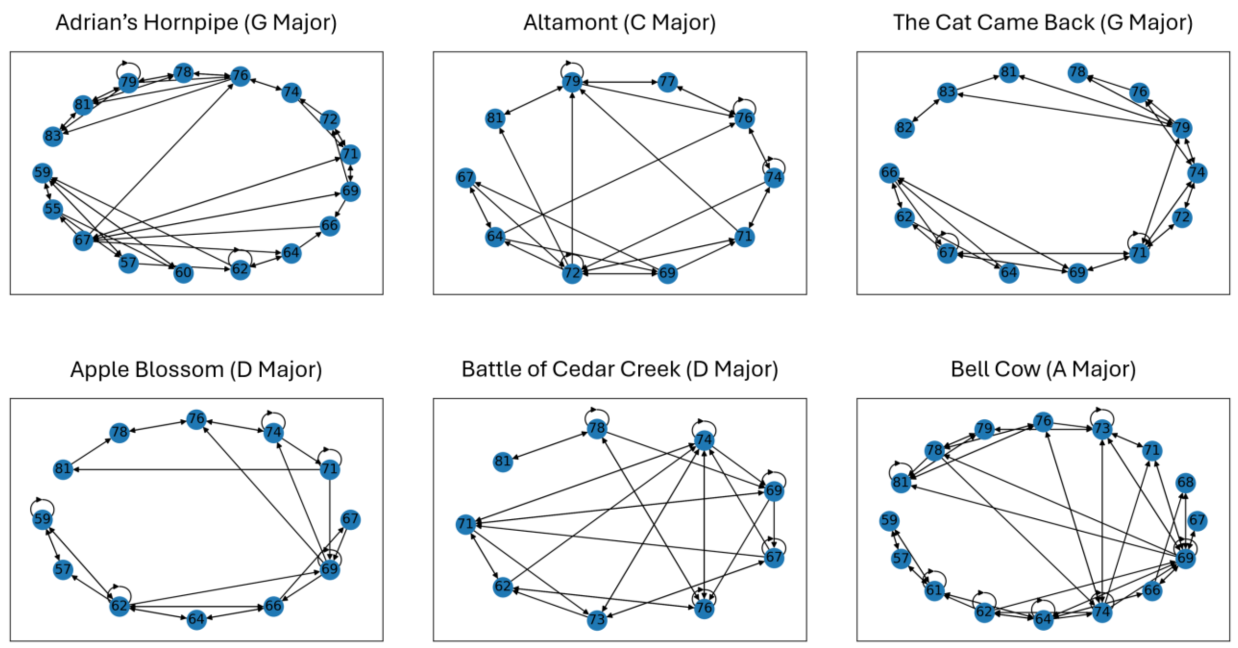

Adrian's Hornpipe (G Major)
Altamont (C Major)
The Cat Came Back (G Major)
Apple Blossom (D Major)
Battle of Cedar Creek (D Major)
Bell Cow (A Major)

Figure 6 - Complex Networks for Singular Old-Time Tunes

2.5 Combining the Complex Networks into Aggregate Graphs

One of the goals of this project was to create aggregate complex networks for each key and genre, for a total of 8 aggregate complex networks. The aggregate complex networks are created in SongAnalysis.py by adding all the adjacency matrices of a key and genre together to make a total weighted adjacency matrix. For example, all D Major Old-Time tune adjacency matrices were added together to create its aggregate complex network. These aggregate network adjacency matrices for each key/genre were then imported into Gephi and converted into a visual representation, which can be seen in Figure 7. The key information for each aggregate graph such as number of nodes, number of edges, average degree, average weighted degree, network diameter, graph density, connected components, modularity, average clustering coefficient, and average path length can be found in Figure 8.

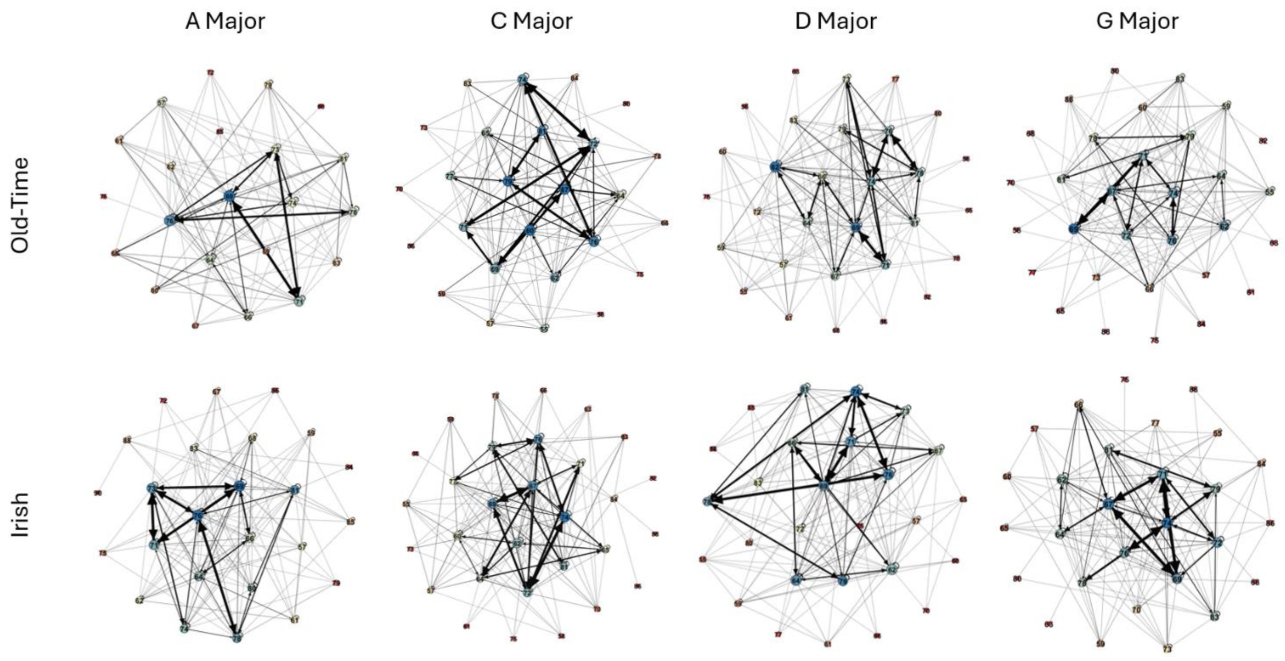


Figure 7 - Aggregate Graphs for the Keys of A Major, C Major, D Major, and G Major for Two Genres, Old-Time and Irish

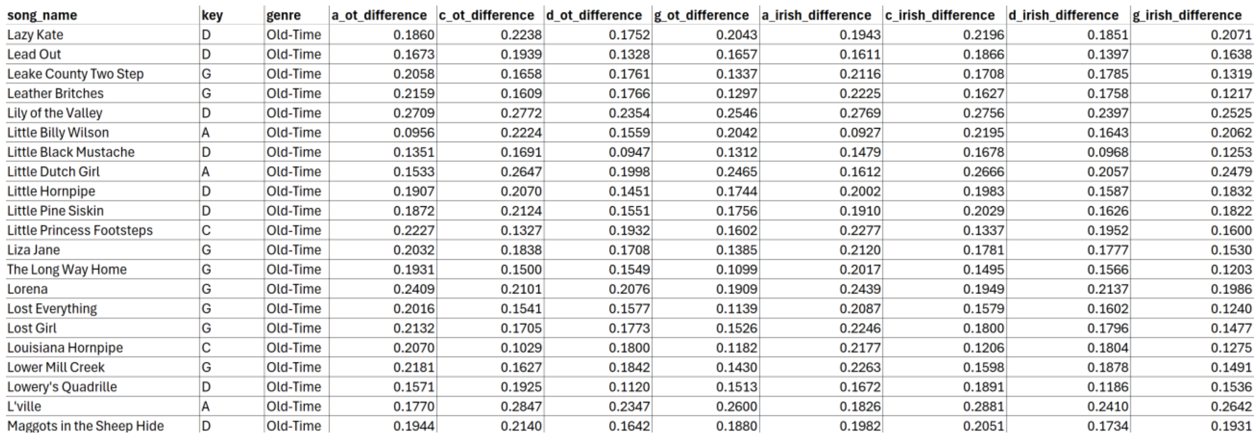
	A Major (Old-time)	C Major (Old-time)	D Major (Old-time)	G Major (Old-time)	A Major (Irish)	C Major (Irish)	D Major (Irish)	G Major (Irish)
Nodes	22	26	30	31	25	30	27	28
Edges	209	234	288	270	228	242	261	274
Average Degree	9.5	9	9.6	8.71	9.12	8.067	9.667	9.786
Average Weighted Degree	795.091	411.192	1376.167	1085.613	869.88	372.533	1762.148	1358.857
Network Diameter	3	5	4	5	4	4	4	4
Graph Density	0.452	0.36	0.331	0.29	0.38	0.278	0.372	0.362
Connected Components	1	1	1	1	1	1	1	1
Modularity	0.343	0.359	0.321	0.332	0.282	0.335	0.335	0.319
Average Clustering Coefficient	0.661	0.677	0.731	0.707	0.691	0.617	0.741	0.706
Average Path Length	1.632	1.875	1.876	2.044	1.842	2.064	1.819	1.848

Figure 8 - Key Data for Aggregate Graphs in Figure 7

After the adjacency matrices had been created for the all aggregate graphs, SongAnalysis.py then went back through every song's adjacency matrix and calculated their difference from each aggregate graph using an equation that (Chen Xin, 2016) utilized for comparing a Classical tune to an aggregate graph of a Classical composer, to see if that song was played by said composer. In this situation, it's comparing a tune to a genre/key aggregate graph to see if a tune is that key and genre. This equation, edited for this project, can be found below. AdjacencyPoint in a song is one cell of the song's adjacency matrix, while AdjacencyPoint in an aggregate graph is one cell in a chosen genre/key aggregate graph. The subtraction and squaring of one from another is done 2,601 times (51 notes x 51 notes) over the entire adjacency matrix for each song.

$$Difference = \sqrt{\sum_{n=1}^{2601}\left(AdjacencyPoint_n^{Song} - AdjacencyPoint_n^{Aggregate\ Graph}\right)^2}$$

A snapshot of these calculated differences can be seen in Figure 9. Looking at the first tune in that figure, Lazy Kate, it can be seen that the lowest value in that row is for d_ot_difference, which means the song is most likely an Old-Time tune in the key of D Major. Double-checking that against the tune's actual key and genre, this is found to be correct. The lower the value is for the difference, the closer it is to that aggregate's graph properties (In this case D Major and Old-Time). This isn't always the case though, sometimes a tune has been found to be closer to the other genre's (but with same key) aggregate graph. For instance, Saint Ann's Reel is a D Major tune that was on OTFT, but according to the difference equation is closer to the D Major Irish aggregate graph. Looking on The Session's website, a tune is found called Saint Anne's Reel, which could possibly be where the Old-Time Saint Ann's Reel originally came from. Since Old-Time music has taken inspiration and borrowed tunes from Irish songs, it's possible a few tunes that are classified as Old-Time are in reality Irish tunes reskinned. Even with this, genre is still generally accurately predicted (more on this in Section 3.2).



song_name	key	genre	a_ot_difference	c_ot_difference	d_ot_difference	g_ot_difference	a_irish_difference	c_irish_difference	d_irish_difference	g_irish_difference
Lazy Kate	D	Old-Time	0.1860	0.2238	0.1752	0.2043	0.1943	0.2196	0.1851	0.2071
Lead Out	D	Old-Time	0.1673	0.1939	0.1328	0.1657	0.1611	0.1866	0.1397	0.1638
Leake County Two Step	G	Old-Time	0.2058	0.1658	0.1761	0.1337	0.2116	0.1708	0.1785	0.1319
Leather Britches	G	Old-Time	0.2159	0.1609	0.1766	0.1297	0.2225	0.1627	0.1758	0.1217
Lily of the Valley	D	Old-Time	0.2709	0.2772	0.2354	0.2546	0.2769	0.2756	0.2397	0.2525
Little Billy Wilson	A	Old-Time	0.0956	0.2224	0.1559	0.2042	0.0927	0.2195	0.1643	0.2062
Little Black Mustache	D	Old-Time	0.1351	0.1691	0.0947	0.1312	0.1479	0.1678	0.0968	0.1253
Little Dutch Girl	A	Old-Time	0.1533	0.2647	0.1998	0.2465	0.1612	0.2666	0.2057	0.2479
Little Hornpipe	D	Old-Time	0.1907	0.2070	0.1451	0.1744	0.2002	0.1983	0.1587	0.1832
Little Pine Siskin	D	Old-Time	0.1872	0.2124	0.1551	0.1756	0.1910	0.2029	0.1626	0.1822
Little Princess Footsteps	C	Old-Time	0.2227	0.1327	0.1932	0.1602	0.2277	0.1337	0.1952	0.1600
Liza Jane	G	Old-Time	0.2032	0.1838	0.1708	0.1385	0.2120	0.1781	0.1777	0.1530
The Long Way Home	G	Old-Time	0.1931	0.1500	0.1549	0.1099	0.2017	0.1495	0.1566	0.1203
Lorena	G	Old-Time	0.2409	0.2101	0.2076	0.1909	0.2439	0.1949	0.2137	0.1986
Lost Everything	G	Old-Time	0.2016	0.1541	0.1577	0.1139	0.2087	0.1579	0.1602	0.1240
Lost Girl	G	Old-Time	0.2132	0.1705	0.1773	0.1526	0.2246	0.1800	0.1796	0.1477
Louisiana Hornpipe	C	Old-Time	0.2070	0.1029	0.1800	0.1182	0.2177	0.1206	0.1804	0.1275
Lower Mill Creek	G	Old-Time	0.2181	0.1627	0.1842	0.1430	0.2263	0.1598	0.1878	0.1491
Lowery's Quadrille	D	Old-Time	0.1571	0.1925	0.1120	0.1513	0.1672	0.1891	0.1186	0.1536
L'ville	A	Old-Time	0.1770	0.2847	0.2347	0.2600	0.1826	0.2881	0.2410	0.2642
Maggots in the Sheep Hide	D	Old-Time	0.1944	0.2140	0.1642	0.1880	0.1982	0.2051	0.1734	0.1931

Figure 9 - Calculated Differences Between a Song's Adjacency Matrix and Various Aggregate Graphs

3 Identification and Classification

3.1 Initial Observations

Analyzing the various statistics from Gephi for the aggregate graphs from Figure 8, these were the initial insights into those networks.

Nodes – For all the aggregate graphs, the number of nodes present ranged from 22 to 31. The max number of nodes there could have been was 51 (the total possible playable notes on a violin tuned in GDAE). These aggregate graphs ranged from covering 43% to 61% of the total playable notes. If Classical tunes had been included in this project, they would have probably been found to have more nodes for their aggregate graphs, due to Classical's propensity for moving the hand up the violin's fingerboard (shifting) to reach higher notes, that Old-Time and Irish avoided.

Connected Components – This was more of a fail-safe statistic, since every graph should be completely connected due to each note connected to the preceding note by an edge. If at some point a note was not connected to the previously played note, this would be cause for an in-depth investigation into the code for errors.

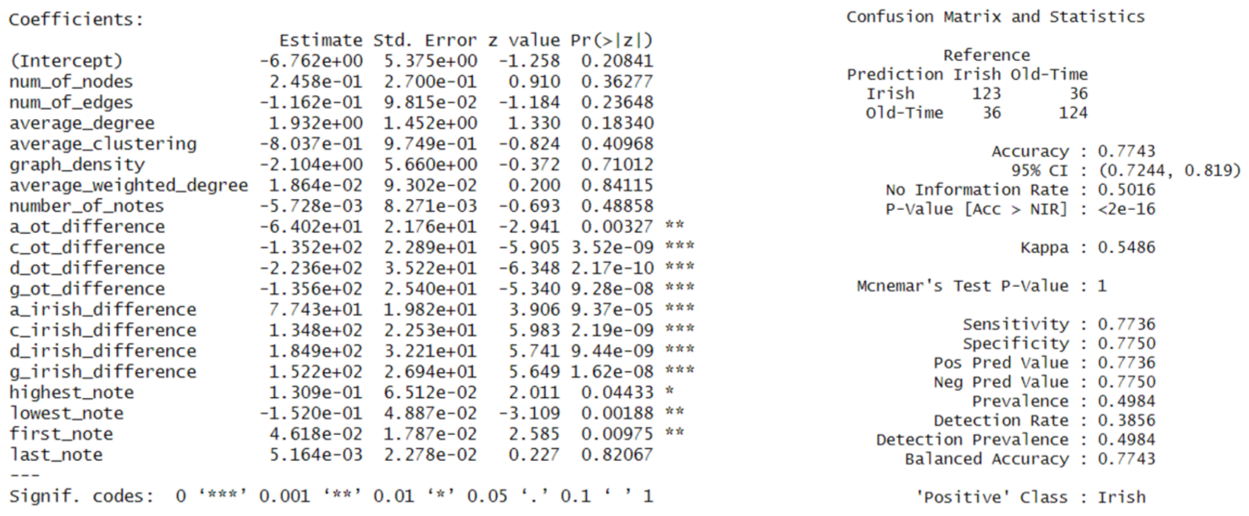
Average Clustering Coefficient – This value is above 0.6 for every type of aggregate graph, meaning most of each node's neighbors are connected to each other. This, combined with the average path length, point to the networks being small-worlds.

Average Path Length – Ranging from 1.632 to 2.064, the average path length for the aggregate graphs is low. This means any note can be reached from any other note within seconds in a tune, since only a note or two may be in between them. This, coupled with the average clustering coefficient, points to the aggregate networks being small-worlds.

For all the above statistics, there wasn't anything that jumped out as a distinguishing factor between the various keys and genres, they all had roughly the same values across the board. This led to the next step, applying machine learning methods to hopefully discover patterns among the tunes that would differentiate them by key and genre.

3.2 Binary and Multinomial Logistic Regression

Two main machine learning methods were used, normal logistic regression (or binary) for predicting the genre of the tunes, and multinomial logistic regression for predicting the key a tune is in. The two spreadsheets for Irish and Old-Time tunes that contained all the data for each song (such as number of nodes, average degree, difference of tune from each key's aggregate graph, etc.) were combined into one and then imported into R Studio. The data was then randomly separated into a training set with 70% of the data and a testing set with the other 30% of the data. The first test was predicting the genre, utilizing binary logistic regression with two classes (Irish and Old-Time). A model was created, and then predictions were made using that model and summarized using a confusion matrix. Those results can be seen in Figure 10. The most significant coefficients for predicting the genre were the calculated differences of each song from the eight aggregate graphs created for each key/genre combination, followed up by the lowest note and first note of a song. Logistic regression was able to accurately predict the genre of a tune 77.43% of the time, with a balanced rate of successful predictions for both Irish and Old-Time.



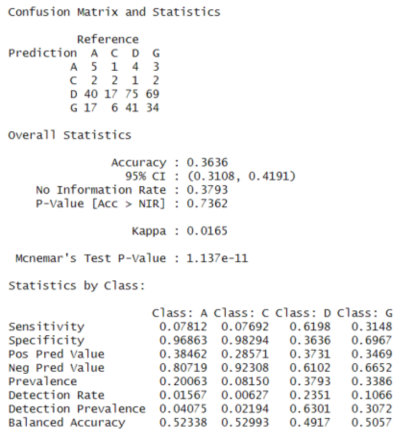
```
Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)            -6.762e+00  5.375e+00  -1.258  0.20841
num_of_nodes            2.458e-01  2.700e-01   0.910  0.36277
num_of_edges           -1.162e-01  9.815e-02  -1.184  0.23648
average_degree          1.932e+00  1.452e+00   1.330  0.18340
average_clustering     -8.037e-01  9.749e-01  -0.824  0.40968
graph_density          -2.104e+00  5.660e+00  -0.372  0.71012
average_weighted_degree 1.864e-02  9.302e-02   0.200  0.84115
number_of_notes        -5.728e-03  8.271e-03  -0.693  0.48858
a_ot_difference        -6.402e+01  2.176e+01  -2.941  0.00327 **
c_ot_difference        -1.352e+02  2.289e+01  -5.905 3.52e-09 ***
d_ot_difference        -2.236e+02  3.522e+01  -6.348 2.17e-10 ***
g_ot_difference        -1.356e+02  2.540e+01  -5.340 9.28e-08 ***
a_irish_difference      7.743e+01  1.982e+01   3.906 9.37e-05 ***
c_irish_difference      1.348e+02  2.253e+01   5.983 2.19e-09 ***
d_irish_difference      1.849e+02  3.221e+01   5.741 9.44e-09 ***
g_irish_difference      1.522e+02  2.694e+01   5.649 1.62e-08 ***
highest_note            1.309e-01  6.512e-02   2.011  0.04433 *
lowest_note            -1.520e-01  4.887e-02  -3.109  0.00188 **
first_note              4.618e-02  1.787e-02   2.585  0.00975 **
last_note               5.164e-03  2.278e-02   0.227  0.82067
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Confusion Matrix and Statistics

          Reference
Prediction Irish Old-Time
  Irish      123       36
  Old-Time    36      124

               Accuracy : 0.7743
                 95% CI : (0.7244, 0.819)
    No Information Rate : 0.5016
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5486

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7736
            Specificity : 0.7750
         Pos Pred Value : 0.7736
         Neg Pred Value : 0.7750
             Prevalence : 0.4984
         Detection Rate : 0.3856
   Detection Prevalence : 0.4984
      Balanced Accuracy : 0.7743

       'Positive' Class : Irish
```

Figure 10 - Coefficient and Confusion Matrix Information for Predicting Genre of a Tune

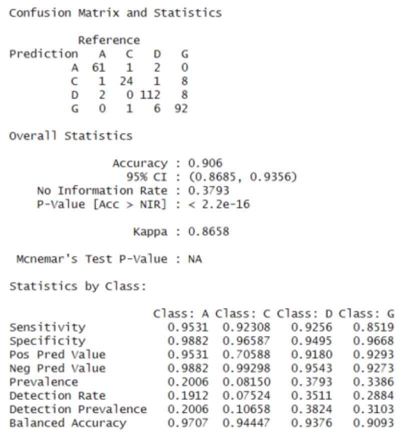
After predicting song genres, the next goal was to predict a song’s key. For this, Multinomial Logistic Regression had to be utilized due to there now being 4 classes (A Major, C Major, D Major, and G Major). Three models were trained with different combinations of predictor variables. The results of these different combinations can be seen in Figure 11. One model was trained using all predictor variables except the aggregate graph differences explained and calculated in Section 2.5. This model was only able to successfully predict the key 36.36% of the time. This was too low to be useful. Another model was trained using all the predictor variables, and was successful at predicting the key 90.6% of the time. On a whim, another model was trained using only the predictor variables that were calculated aggregate graph differences, and this model had the highest successful predictions with 93.73% of the predictions of a song’s key being correct.

Without Aggregate Graph Differences



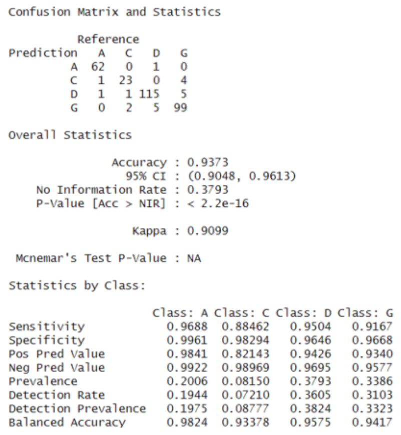
```
Confusion Matrix and Statistics

          Reference
Prediction  A  C  D  G
         A  5  1  4  3
         C  2  2  1  2
         D 40 17 75 69
         G 17  6 41 34

Overall Statistics

               Accuracy : 0.3636
                 95% CI : (0.3108, 0.4191)
    No Information Rate : 0.3793
    P-Value [Acc > NIR] : 0.7362

                  Kappa : 0.0165

 Mcnemar's Test P-Value : 1.137e-11

Statistics by Class:

                     Class: A Class: C Class: D Class: G
Sensitivity           0.07812  0.07692   0.6198   0.3148
Specificity           0.96863  0.98294   0.3636   0.6967
Pos Pred Value        0.38462  0.28571   0.3731   0.3469
Neg Pred Value        0.80719  0.92308   0.6102   0.6652
Prevalence            0.20063  0.08150   0.3793   0.3386
Detection Rate        0.01567  0.00627   0.2351   0.1066
Detection Prevalence  0.04075  0.02194   0.6301   0.3072
Balanced Accuracy     0.52338  0.52993   0.4917   0.5057
```

With Everything



```
Confusion Matrix and Statistics

          Reference
Prediction   A   C   D   G
         A  61   1   2   0
         C   1  24   1   8
         D   2   0 112   8
         G   0   1   6  92

Overall Statistics

               Accuracy : 0.906
                 95% CI : (0.8685, 0.9356)
    No Information Rate : 0.3793
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8658

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: C Class: D Class: G
Sensitivity            0.9531  0.92308   0.9256   0.8519
Specificity            0.9882  0.96587   0.9495   0.9668
Pos Pred Value         0.9531  0.70588   0.9180   0.9293
Neg Pred Value         0.9882  0.99298   0.9543   0.9273
Prevalence             0.2006  0.08150   0.3793   0.3386
Detection Rate         0.1912  0.07524   0.3511   0.2884
Detection Prevalence   0.2006  0.10658   0.3824   0.3103
Balanced Accuracy      0.9707  0.94447   0.9376   0.9093
```

With Only Aggregate Graph Differences

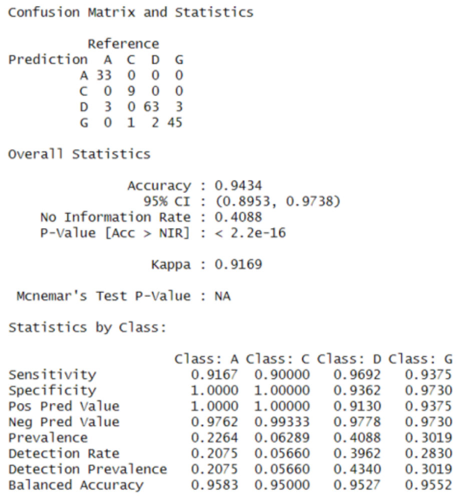


```
Confusion Matrix and Statistics

          Reference
Prediction   A   C   D   G
         A  62   0   1   0
         C   1  23   0   4
         D   1   1 115   5
         G   0   2   5  99

Overall Statistics

               Accuracy : 0.9373
                 95% CI : (0.9048, 0.9613)
    No Information Rate : 0.3793
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9099

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: C Class: D Class: G
Sensitivity            0.9688  0.88462   0.9504   0.9167
Specificity            0.9961  0.98294   0.9646   0.9668
Pos Pred Value         0.9841  0.82143   0.9426   0.9340
Neg Pred Value         0.9922  0.98969   0.9695   0.9577
Prevalence             0.2006  0.08150   0.3793   0.3386
Detection Rate         0.1944  0.07210   0.3605   0.3103
Detection Prevalence   0.1975  0.08777   0.3824   0.3323
Balanced Accuracy      0.9824  0.93378   0.9575   0.9417
```

Figure 11 - Confusion Matrices and Statistics for Various Combinations of Predictor Variables to Predict Song Keys

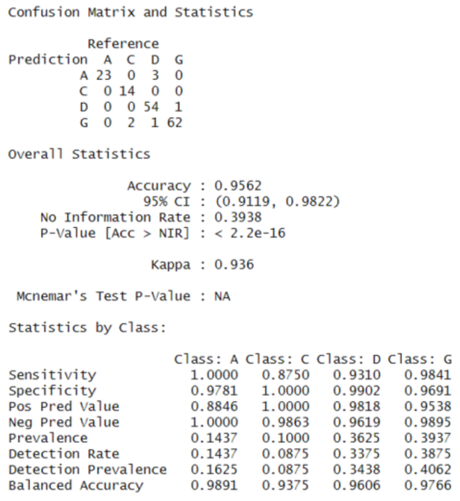
To guarantee that the success rate of predictions wasn't high simply because elements of the song being predicted were in the created aggregate graph, two more tests were performed. The Irish tunes keys were predicted, but only using their differences from the Old-Time aggregate graphs as predictor variables (which there would be no trace of the Irish tunes contained within since these aggregate graphs were trained purely on Old-Time tunes), and vice versa with the Old-Time tunes keys being predicted using only their differences from the Irish aggregate graphs. The results of these tests can be found in Figure 12. The Irish tunes' keys were successfully predicted 94.34% of the time and the Old-Time tunes' keys were 95.62% successfully predicted. This helped to confirm that the keys were being accurately predicted because of the aggregate graphs overall, and not because trace bits of each tune lay in those aggregate graphs.

Irish Tunes with only Old-Time Aggregate Graph Differences



```
Confusion Matrix and Statistics

          Reference
Prediction  A  C  D  G
         A 33  0  0  0
         C  0  9  0  0
         D  3  0 63  3
         G  0  1  2 45

Overall Statistics

               Accuracy : 0.9434
                 95% CI : (0.8953, 0.9738)
    No Information Rate : 0.4088
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9169

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: C Class: D Class: G
Sensitivity            0.9167  0.90000   0.9692   0.9375
Specificity            1.0000  1.00000   0.9362   0.9730
Pos Pred Value         1.0000  1.00000   0.9130   0.9375
Neg Pred Value         0.9762  0.99333   0.9778   0.9730
Prevalence             0.2264  0.06289   0.4088   0.3019
Detection Rate         0.2075  0.05660   0.3962   0.2830
Detection Prevalence   0.2075  0.05660   0.4340   0.3019
Balanced Accuracy      0.9583  0.95000   0.9527   0.9552
```

Old-Time Tunes with only Irish Aggregate Graph Differences



```
Confusion Matrix and Statistics

          Reference
Prediction  A  C  D  G
         A 23  0  3  0
         C  0 14  0  0
         D  0  0 54  1
         G  0  2  1 62

Overall Statistics

               Accuracy : 0.9562
                 95% CI : (0.9119, 0.9822)
    No Information Rate : 0.3938
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.936

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: C Class: D Class: G
Sensitivity            1.0000   0.8750   0.9310   0.9841
Specificity            0.9781   1.0000   0.9902   0.9691
Pos Pred Value         0.8846   1.0000   0.9818   0.9538
Neg Pred Value         1.0000   0.9863   0.9619   0.9895
Prevalence             0.1437   0.1000   0.3625   0.3937
Detection Rate         0.1437   0.0875   0.3375   0.3875
Detection Prevalence   0.1625   0.0875   0.3438   0.4062
Balanced Accuracy      0.9891   0.9375   0.9606   0.9766
```

Figure 12 - Confusion Matrices and Statistics to Test if Song Key Predictions were Biased

4 Song Generation

A program was created called SongGeneration.py to create new tunes based on previous ones. The program takes as input one of the eight aggregate graphs created in earlier sections. This gives the generator both the key that the tune will be in and also the genre, since both traits are merged in every aggregate graph. Once the seed aggregate graph has been selected, the generator begins looking at all previous tunes in that key and genre and what note they started with. Counting how many times each unique note started a song, and then dividing by the total number of songs for that key and genre, gives the probability each unique note will begin one of the songs generated by the program. For instance, 23 Old-Time tunes in the key of D Major began with the MIDI note number 62, or D4 (an open note on the D string of a violin) out of 210 D Major tunes, leading to a probability of 10.95% that a D Major Old-Time tune will start on an open D string. For an example run, the aggregate graph for Old-Time in D Major was selected, and the computer chose MIDI note number 64 (E4) as the starting note (This can be visualized in Figure 13).

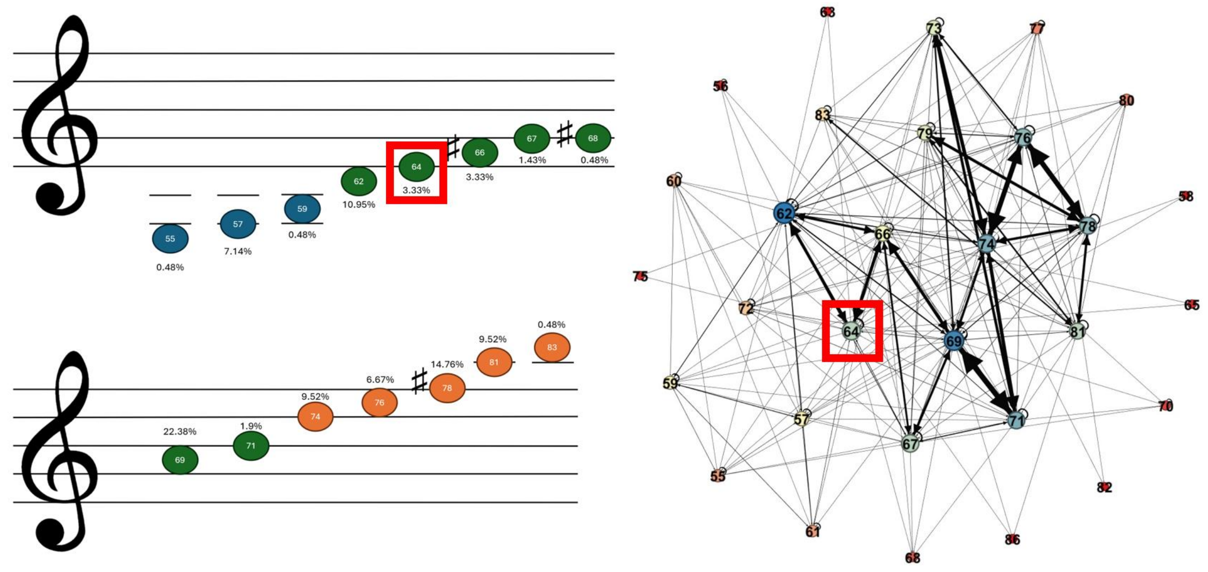
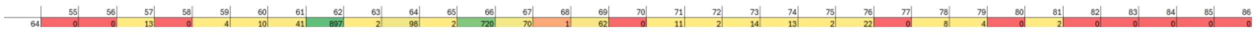


Figure 13 - Selecting the First Note of a Computer Generated Song (Old-Time in D Major)

From there, if the user likes the starting note, they can then select how long each section of the musical piece will be, how many sections, and how many times each section is repeated. For a typical Old-Time or Irish tune, each section is about 50 notes long, with there being two sections that are repeated twice (an A part and a B part, no relation to the notes by the same name). The user can also select how far notes will stray from the beginning pitch if they don't want the song to be too adventurous, or they can the let the tune freely go wherever it may take it. The user can also select how the outputted audio file will sound, whether they want each note to be played the same length or if they want the song to be played with shuffle bowing (one long note, two short notes, repeated over and over). Once all preferences have been selected, the program precedes to generate the tune. Beginning with the first note chosen, it looks at all possible following notes for that note in the adjacency matrix in that MIDI note number's row. For MIDI note number 64, it's row in the aggregate graph adjacency matrix can be viewed in Figure 14.



	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86
64	0	0	13	0	4	10	41	897	2	98	2	720	70	1	62	0	11	2	14	13	2	22	0	8	4	0	2	0	0	0	0	0

Figure 14 - All Notes and the Number of Times Those Notes Followed MIDI Note Number 64

Taking this distribution of following notes, probabilities are assigned based on the number in each cell divided by the overall number of notes in that row. To visualize that in graph form, see Figure 15. In Figure 15, MIDI note number 66 was chosen by the program as the note to follow MIDI note number 64 (the starting note). MIDI note number 66 becomes the new "starting note" and the process repeats until the specified number of notes is obtained for the musical piece. Inspiration was taken from (Xiao Fan Liu, 2009) for the process of choosing the next note based on edge weight.

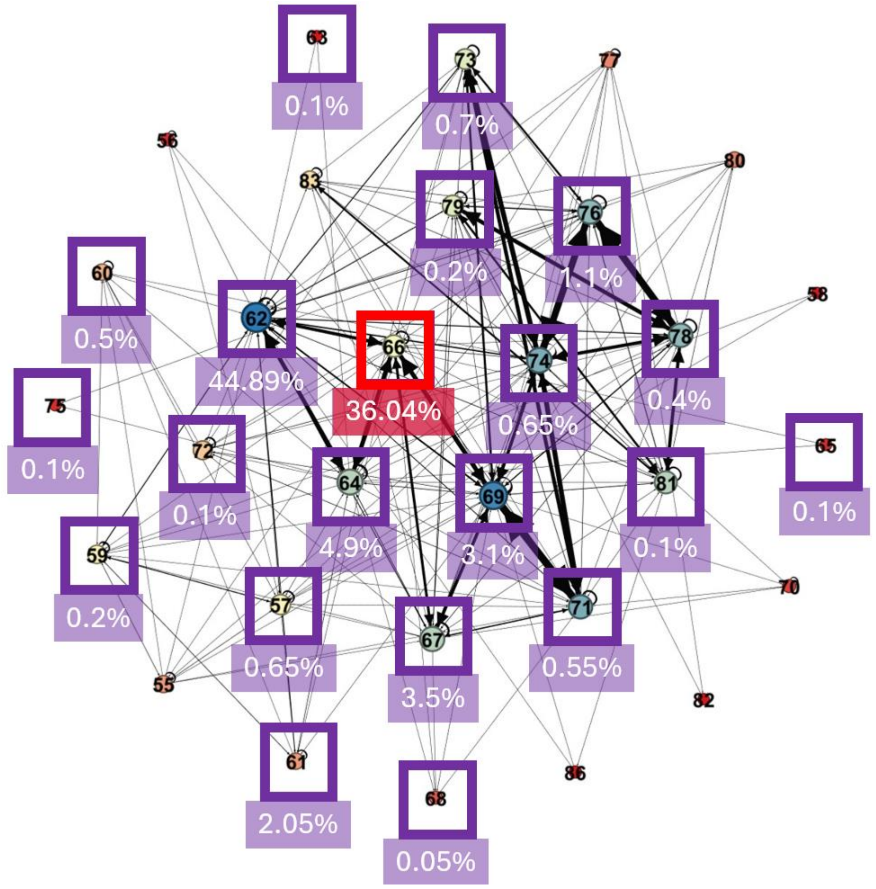


Figure 15 - Aggregate Graph for D Major Old-Time with the Probabilities of Next Note Visualized, as Well as the Note Chosen by the Program Marked in Red

Once the song is completed, sheet music and a MIDI file are outputted for the user to listen to and learn how to play the song. The library MIDIUtil was utilized for the creation of MIDI files. (Wirt, 2018) An example of the outputted sheet music can be seen in Figure 16.

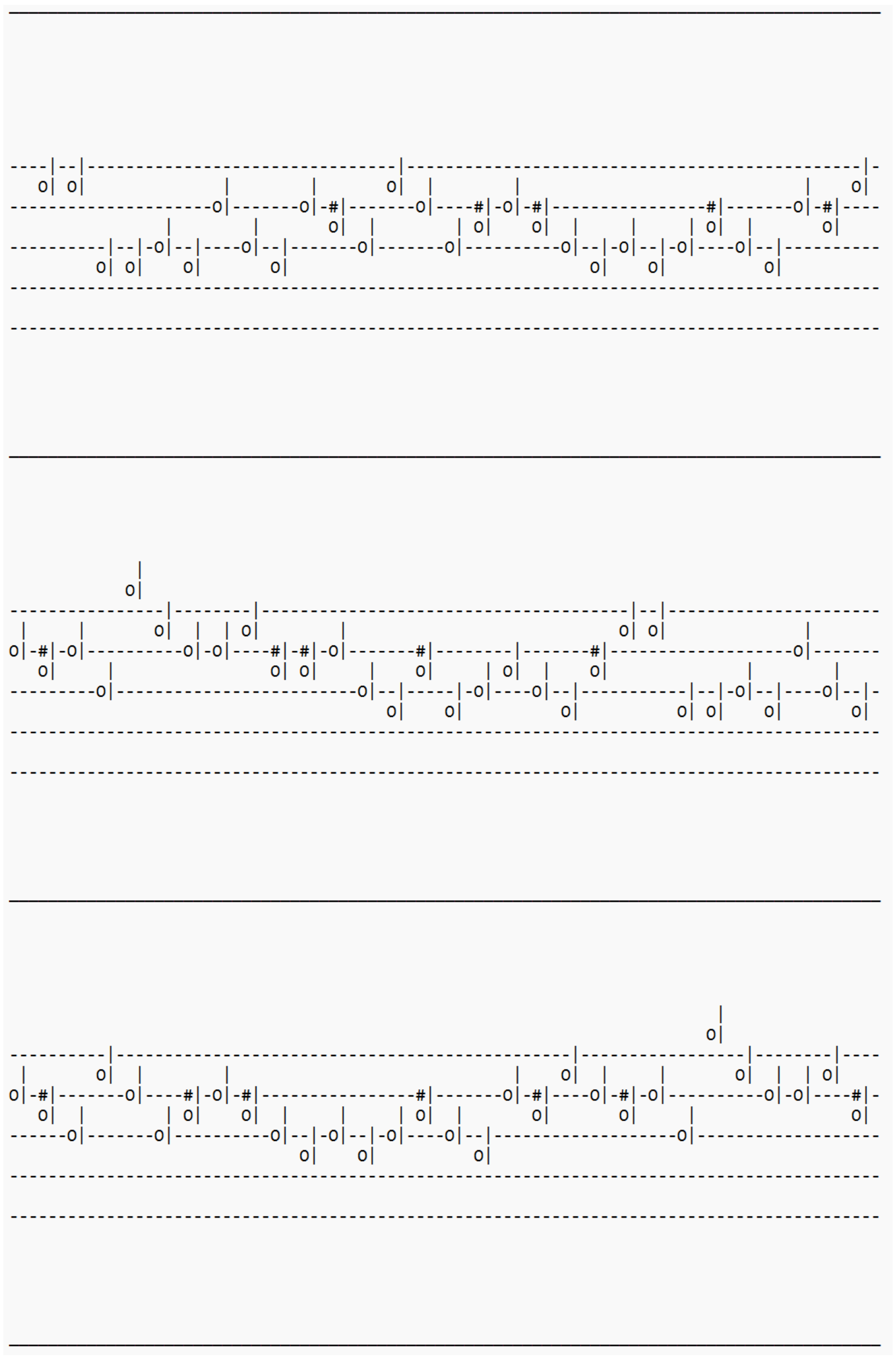


Figure 16 - Randomly Generated Old-Time Song in the Key of D Major

5 Conclusion

Overall, the main goals set out for this project were achieved (predicting tunes' keys and genres and generating new tunes). Predicting the genre of a tune could be improved from its 77.43% success rate, though with how similar the two genres were, this rate was higher than expected. An accuracy rate of 93.73% for predicting tune's keys was found to be acceptable for most purposes. The songs generated generally had a pleasing sound (which is more subjective than scientific). Having only ~500 songs to work with for Old-Time and Irish definitely placed some restrictions on what could be done, especially for the key of C Major that only had ~50 songs for creating its aggregate graph. It was thought the various properties of complex networks (such as average degree, diameter, etc.) would have more of a role in determining a song's genre and key, but based on the logarithmic regressions run on the data, these values played very little part in predictions. The difference between a song's adjacency matrix and all aggregate adjacency matrices however played a major role in determination of a song's genre and key. Using this knowledge, aggregate graphs created with more songs in that key and genre could create a set of fingerprints per se that could be used to identify and create more unique songs in the future.

Also, the author had hoped to delve more into creating aggregate graphs for specific Old-Time composers, but a lack of large amounts of tunes per composer and too many unique composers in the OTFT database (317 composers to be exact out of 655 tunes) prevented this particular avenue of study.

6 Future Work

There are several routes of study to pursue as future work from this project. One of the main ones is the analysis of more unique genres. While Irish and Old-Time tunes had just enough differences in their structures to allow mostly successful predictions of song's genres, they also shared similar roots. Comparing these songs to aggregate graphs created from Classical, Jazz, or other songs would give more insight into whether using the equation for differences between a song's adjacency matrix and an aggregate graph's adjacency matrix would work even more successfully with songs and genres that have a lot less in common with each other. Also, no songs in Irish or Old-Time went above MIDI note number 90, when the max possible MIDI note number on the violin is 105. Having an aggregate graph that covers the higher notes in a key would be useful for future predictions for songs that are higher pitched in those keys. Creating more aggregate graphs for different keys would be another avenue for the future, such as E Major, F Major, B Major, D Minor, A Minor, etc.

Another possibility would be to incorporate rhythm into the aggregate graphs for every key. Currently the adjacency matrices were 51 by 51 (2601 possibilities), but incorporating each note's duration (whole note, half note, quarter note, eighth note, sixteenth note) could expand that to a 255 by 255 adjacency matrix (65,025 possibilities), and that's not even including the various rests or triplets. This wasn't done for this project because it would have diluted the aggregate adjacency matrices made, especially since the max number of songs for a key was no more than 220. If a source with more songs could be found, or if a good web scraper could be built to take advantage of the numerous songs on The Session's website, this would then be a good possibility for research. This could then allow the created song generator to incorporate rhythm into it's generation of tunes (though only for Irish, since no massive source of Old-Time MIDI files has been found as of yet).

Works Cited

(n.d.). Retrieved from The Session: https://thesession.org/

Chen Xin, H. Z. (2016). Complex network approach to classifying classical piano compositions. *EPL*.

Lamancusa, J. (2024, April 4). Retrieved from Old Time Fiddle Tunes: https://www.oldtimefiddletunes.net/

Walker, J. (2008, January). *midiCSV*. Retrieved from fourmilab.ch: https://www.fourmilab.ch/webtools/midicsv/

Wirt, M. C. (2018, March 4). *MIDIUtil 1.2.1*. Retrieved from PyPi.org: https://pypi.org/project/MIDIUtil/

Xiao Fan Liu, C. K. (2009). Composing Music with Complex Networks. *Complex Sciences: First International Conference*.