Assignment 1 - Probability, Linear Algebra, & Computational Programming

Emma Mavis
NetID: egm28

Probability and Statistics Theory

1. Probability Density Function

$f(x)$ is a valid probability density function when $\alpha = \frac{1}{4}$

2. Cumulative Distribution Function

The CDF that corresponds to the PDF $f(x)$ is

$\int_{-\infty}^{0} 0\,dx + \int_{0}^{3} \frac{1}{3}\,dx + \int_{3}^{\infty} 0\,dx = \frac{1}{3}x$ for $0 < x < 3$ and 0 otherwise.

3. Expected Value and Variance

- $E(X) = \int_{-\infty}^{\infty} xf(x)\,dx = \int_{-\infty}^{0} 0 \times x\,dx + \int_{0}^{3} \frac{1}{3}x\,dx + \int_{3}^{\infty} 0 \times x\,dx = \frac{1}{6}x^2\Big|_0^3 = \frac{3}{2}$
- $Var(X) = \int_{-\infty}^{\infty} [x - E(X)]^2 f(x)\,dx = \int_{-\infty}^{\infty} x^2 - 3x + \frac{9}{4}\,dx = \frac{1}{3}x^3 - \frac{3}{2}x^2 + \frac{9}{4}x\Big|_0^3 =$

4. Mean and Variance

- $E(X) = \sum xf(x) = 2(\frac{1}{5}) + 3(\frac{1}{5}) + 10(\frac{1}{5}) - 1(\frac{1}{5}) - 1(\frac{1}{5}) = \frac{13}{5}$
- $Var(X) = \sum [x - E(X)]^2 f(x) = \frac{2022}{25} \sim 16.2$

Linear Algebra

In [ ]:
```
import numpy as np
# importing libraries for matrix display purposes
from sympy import Matrix, init_printing
init_printing()
```

In [ ]:
```
# initializing matrices
A = np.array([[1, 2, 3], [2, 4, 5], [3, 5, 6]])
b = np.array([[-1], [3], [8]])
c = np.array([[4], [-3], [6]])
I = np.eye(3)
```

5. Matrix Manipulation and Multiplication

- **AA**

In [ ]:
```
display(Matrix(np.matmul(A, A)))
```

$$\begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$$

- **AA**^{T}

In [ ]:
```
display(Matrix(np.matmul(A, A.transpose())))
```

$$\begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$$

- **Ab**

In [ ]:
```
display(Matrix(np.matmul(A, b)))
```

$$\begin{bmatrix} 29 \\ 50 \\ 60 \end{bmatrix}$$

- **Ab**^{T} cannot be computed. A $3x3$ matrix and a $1x3$ matrix cannot be multiplied.

- **bA** cannot be computed. A $3x1$ matrix and a $3x3$ matrix cannot be multiplied.

- **b**^{T}**A**

In [ ]:
```
display(Matrix(np.matmul(b.transpose(), A)))
```

$$\begin{bmatrix} 29 & 50 & 60 \end{bmatrix}$$

- **bb** cannot be computed. A $3x1$ matrix and a $3x1$ matrix cannot be multiplied.

- **b**^{T}**b**

In [ ]:
```
display(Matrix(np.matmul(b.transpose(), b)))
```

[74]

- **bb**^{T}

In [ ]:
```
display(Matrix(np.matmul(b, b.transpose())))
```

$$\begin{bmatrix} 1 & -3 & -8 \\ -3 & 9 & 24 \\ -8 & 24 & 64 \end{bmatrix}$$

- **b** + **c**^{T} cannot be computed. Matrices with mismatched dimensions cannot be added.

- **b**^{T}**b**^{T} cannot be computed. $1x3$ and $1x3$ matrices cannot be multiplied.

- **A**$^{-1}$**b**

In [ ]:
```
display(Matrix(np.matmul(np.linalg.inv(A), b)))
```

$$\begin{bmatrix} 6.0 \\ 4.0 \\ -5.0 \end{bmatrix}$$

- **A** ∘ **A**

In [ ]:
```
display(Matrix(np.multiply(A, A)))
```

$$\begin{bmatrix} 1 & 4 & 9 \\ 4 & 16 & 25 \\ 9 & 25 & 36 \end{bmatrix}$$

- **b** ∘ **c**

In [ ]:
```
display(Matrix(np.multiply(b, c)))
```

$$\begin{bmatrix} -4 \\ -9 \\ 48 \end{bmatrix}$$

6. Eigenvalues and Eigenvectors

- Eigenvectors and eigenvalues for matrix A

In [ ]:
```
w = np.linalg.eig(A)[0]
v = np.linalg.eig(A)[1]
print(f"The eigenvalues of matrix A are {w[0]}, {w[1]}, and {w[2]}. The corre
```

```
The eigenvalues of matrix A are 11.344814282762083, -0.5157294715892572, and 0
.17091518882717865. The corresponding eigenvectors are [-0.32798528 -0.5910090
5 -0.73697623], [-0.73697623 -0.32798528  0.59100905], and [ 0.59100905 -0.736
97623  0.32798528].
```

- Show that $Av = \gamma v$ for eigenvector v and eigenvalue γ.

$$Av = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} -0.33 \\ -0.59 \\ -0.74 \end{bmatrix} = \begin{bmatrix} -3.72 \\ -6.70 \\ -8.36 \end{bmatrix} = 11.34 \times \begin{bmatrix} -0.33 \\ -0.59 \\ -0.74 \end{bmatrix} = \gamma v$$

In [ ]:
```
# computation for above problem
np.matmul(A, v[:,0])
```

Out[ ]:
```
array([-3.72093206, -6.70488789, -8.36085845])
```

In [ ]:
```
# computation for above problem
np.matmul(A, v[:,0]) / w[0]
```

Out[ ]:
```
array([-0.32798528, -0.59100905, -0.73697623])
```

In [ ]:
```
# computation for above problem
w[0]*v[:,0]
```

Out[ ]:
```
array([-3.72093206, -6.70488789, -8.36085845])
```

- Show that the eigenvectors are orthogonal to each other. We can do this by taking the three separate inner products and confirm that they all are zero.

In [ ]:
```
print(f"For eigenvectors v1, v2, and v3, we have the following:")
print(f"v1 * v2 = {np.round(np.inner(v[:,0], v[:,1]), 4)}")
print(f"v1 * v3 = {np.round(np.inner(v[:,0], v[:,2]), 4)}")
print(f"v2 * v3 = {np.round(np.inner(v[:,1], v[:,2]), 4)}")
```

```
For eigenvectors v1, v2, and v3, we have the following:
v1 * v2 = -0.0
v1 * v3 = -0.0
v2 * v3 = -0.0
```

Therefore, the vectors are orthogonal.

Numerical Programming

7. Loading data and gathering insights from a real dataset.

In [ ]:
```
# load in data
import pandas as pd
data = pd.read_excel("~/Downloads/a1_egrid2016.xlsx", skiprows=0, header=1)
```

In [ ]:
```
# (a)
data.sort_values(by="PLNGENAN", ascending=False)
```

Out[ ]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPC
390	391	AZ	Palo Verde	33.388100	-112.861700	NUC	0.87801	420
164	165	AL	Browns Ferry	34.704200	-87.118900	NUC	0.85648	349
7966	7967	PA	Peach Bottom	39.758936	-76.268742	NUC	0.89679	278
8764	8765	TX	South Texas Project	28.795000	-96.048100	NUC	0.91432	270
8157	8158	SC	Oconee	34.793900	-82.898600	NUC	0.90654	266
...	...	...	...	...	...	...	...	
9683	9684	WY	Osage (WY)	43.970300	-104.410600	SUB	NaN	3
9684	9685	WY	Pilot Butte	43.219017	-108.787681	WAT	NaN	
9694	9695	WY	Simpson Ridge Wind Farm LLC	41.879444	-106.369722	WND	NaN	5
9698	9699	WY	Sweetwater Solar	41.629056	-109.683472	SUN	NaN	8
9703	9704	WY	Two Elk Generating Station	43.717500	-105.497500	WC	NaN	32

9709 rows × 10 columns

In [ ]:
```
# (b) (c)
data.sort_values(by="LAT", ascending=False)
```

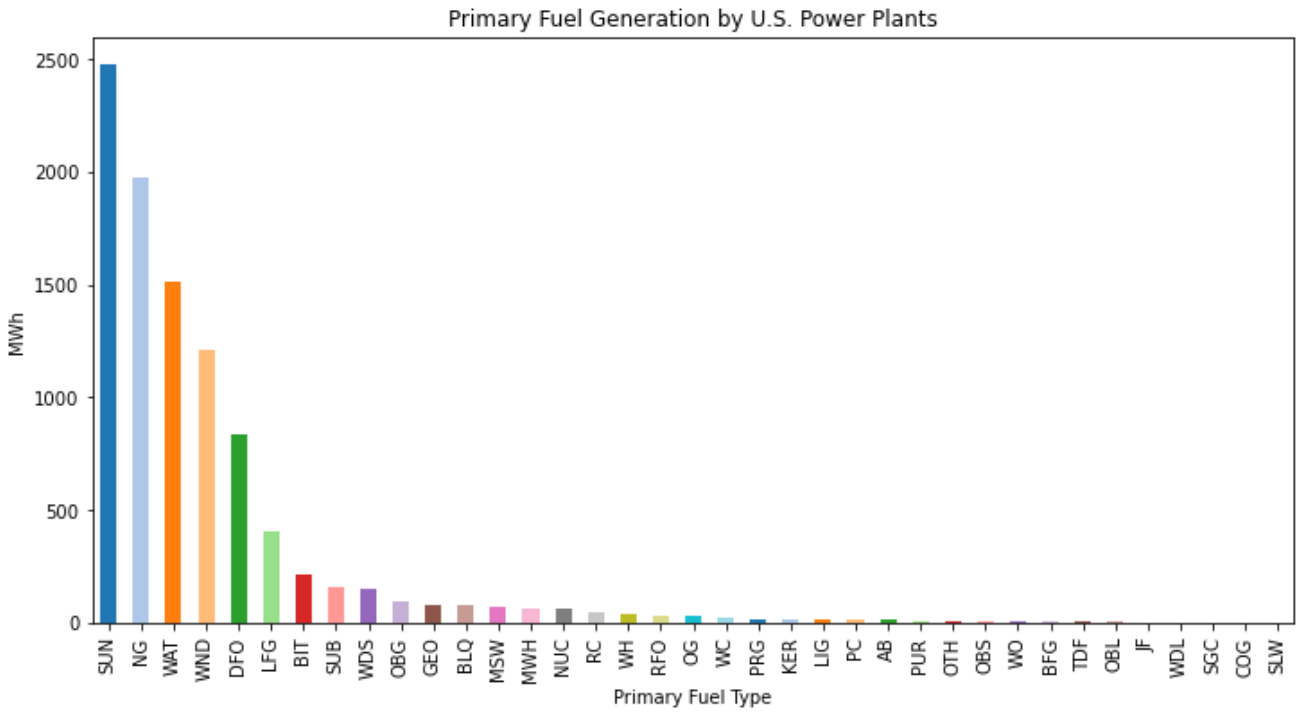
Out[ ]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPC
11	12	AK	Barrow	71.292000	-156.778600	NG	0.28208	2
93	94	AK	NSB Wainwright Utility	70.642877	-160.020461	DFO	0.23509	
88	89	AK	NSB Atqasuk Utility	70.482600	-157.425200	DFO	0.09599	
131	132	AK	TNSG North Plant	70.235278	-148.383611	NG	0.32293	2
90	91	AK	NSB Nuiqsut Utility	70.220565	-150.993492	DFO	0.15099	
...	...	...	...	...	...	...	...	
8809	8810	TX	Turbine	NaN	NaN	NG	NaN	12
9349	9350	WA	SPI - Everett	NaN	NaN	WDS	NaN	2
9574	9575	WI	Wheaton Solar	NaN	NaN	SUN	NaN	
9596	9597	WV	Chemours Belle Plant	NaN	NaN	NG	NaN	N
9631	9632	WV	UCC South Charleston Plant	NaN	NaN	NG	NaN	N

9709 rows × 10 columns

In [ ]:

```
# (d) (e)
import matplotlib.pyplot as plt
cmap = plt.get_cmap("tab20")
ax = data["PLPRMFL"].value_counts().plot(kind = 'bar', figsize = (12,6), colo
ax.set_xlabel("Primary Fuel Type")
ax.set_ylabel("MWh")
ax.set_title("Primary Fuel Generation by U.S. Power Plants")
plt.show()
```



a. The Palo Verde Plant in Arizona generated the most energy: 32,377,480 MWh.

b. The Northern-most power plant in the U.S. is named Barrow.

c. Barrow is located in Alaska.

d. See above plot.

e. The sun produces the most energy in the U.S., followed by natural gas and water.

8. Vectorization

In [ ]:
```
# setup
import time
a = np.random.randn(10000000)
```

In [ ]:
```
# calculation using for loop
sum = 0
start1 = time.time()
for num in a:
    sum += num**2
end1 = time.time()
```

In [ ]:
```
# calculation using vectorization
start2 = time.time()
a.dot(a)
end2 = time.time()
```

In [ ]:
```
print(f"Time [sec] (non-vectorized): {np.round(end1 - start1, 4)}")
print(f"Time [sec] (vectorized):     {np.round(end2 - start2, 4)}")
print(f"The vectorized code is {np.round((end1 - start1)/(end2-start2), 4)} t
```

```
Time [sec] (non-vectorized): 5.4883
Time [sec] (vectorized):     0.015
The vectorized code is 366.7202 times faster than the nonvectorized code
```

9. Basic Numerical Programming and Probabilistic Thinking

In [ ]:
```
# import libraries and set seed for reproducibility
from numpy.random import seed
from numpy.random import normal
seed(1)
```

1. Create vector of normally distributed points

In [ ]:
```
x = normal(loc=2, scale=1, size=10**4)
```

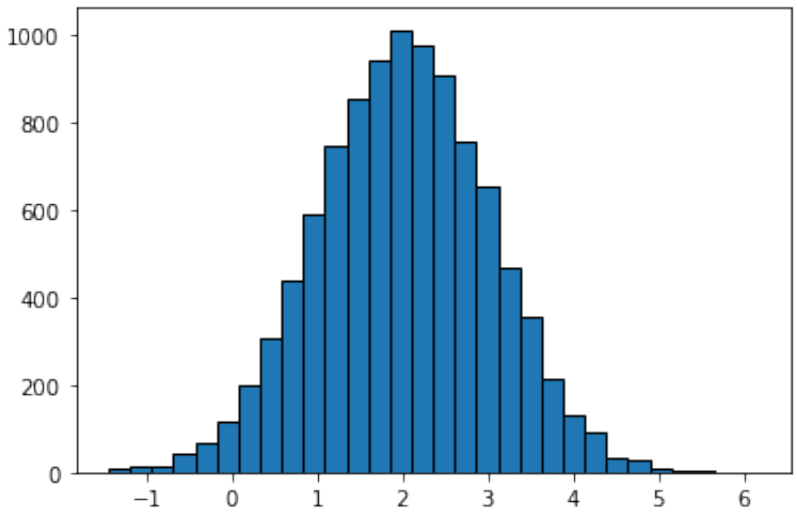
1. Validate mean and standard deviation

In [ ]:
```
print(f"The mean of x is {np.round(x.mean(),3)} and the standard deviation is
```

```
The mean of x is 2.009 and the standard deviation is 1.001.
```

1. Histogram of points

In [ ]:
```
plt.hist(x, bins=30, edgecolor="black")
plt.show()
```



1. 90th percentile

In [ ]:
```
print(f"The 90th percentile of x is {np.round(np.percentile(x, 90), 3)}.")
```

```
The 90th percentile of x is 3.298.
```

1. 99th percentile

In [ ]:
```
print(f"The 99th percentile of x is {np.round(np.percentile(x, 99), 3)}.")
```

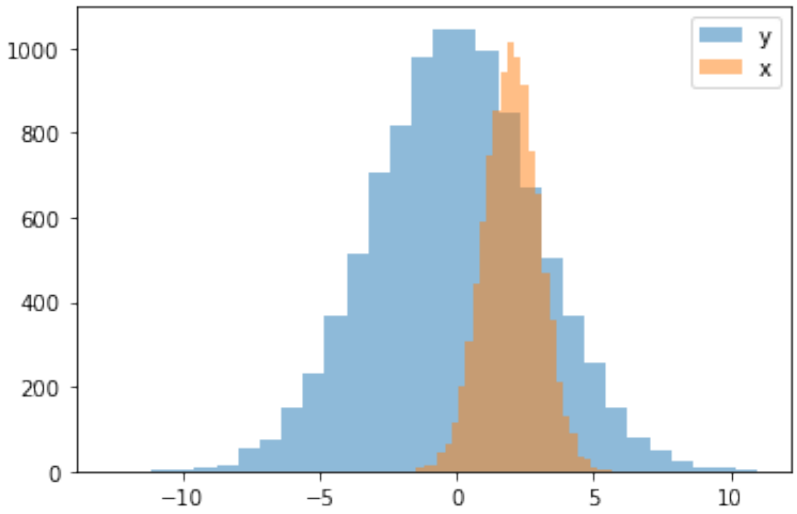
```
The 99th percentile of x is 4.295.
```

1. Create second vector of normally distributed points

In [ ]:
```
y = normal(loc=0, scale=3, size=10**4)
```

1. Overlaid histograms of x and y

In [ ]:
```
plt.hist(y, bins=30, alpha=0.5, label='y')
plt.hist(x, bins=30, alpha=0.5, label='x')
plt.legend()
plt.show()
```



1. Theoretically, $E(\mathbf{XY}) = E(\mathbf{X})E(\mathbf{Y}) = 0$. Based on the samples of the distribution though, $E(\mathbf{x})E(\mathbf{y}) = -0.088$.

In [ ]:
```
# calculation for above problem
x.mean()*y.mean()
```

Out[ ]: -0.0880344959434694

Version Control via Git

10. Atlassian Git Tutorial

I, Emma Mavis, affirm that I have completed the above tutorial.

Exploratory Data Analysis

11. Analysis of Spotify Song Data

Introduction

The style of popular music has changed drastically over time. As technology and culture change (as frequently as every decade), so does our taste in music. There is a huge difference in the features of music that was commonly listened to in the 1960's versus what we listen to now in the 2020's. I have two research questions that I would like to analyze in order to investigate this phenomenon. First, what features of a song make it popular, regardless of release date? Second, what are the primary features that stand out in music for each decade since the 1960s?

I will be using a dataset found on Kaggle that contains records of songs from 1921 until 2020. This data contains the release date, artist and song title, as well as features like energy, instrumentalness, and tempo - features that are measured on a 0-1 scale that help describe the different aspects of music. To investigate my first question, the response variable will be "popularity," and I will find the set of variables that appear to best have a relationship with the response. For the second question,

After initial inspection of the data, I found that there are no missing values and no duplicate entries. Next I subsetted on only the columns I would be using in the analyses. I also started by using data going back to 1921, but initial plotting showed very little useful information - there are so many data points and they span such a wide range of time. I therefore decided to use only the points with a release date from 1960 on, which created a bit more uniformity among the points. This makes sense because 1960 does represent a sort of "turning point" in the style of music that was made. Pre-1960 is so starkly different from post-1960, so it is beneficial to keep those groups of songs separate from each other.

In [ ]:

```
# load and inspect data
spot = pd.read_csv("data.csv")
spot.isna().sum()
spot.duplicated().sum()
spot.dtypes
to_remove = ["duration_ms", "id", "explicit", "key", "loudness", "mode", "rel
data = spot.loc[:, ~spot.columns.isin(to_remove)]
data = data[data.year > 1959]
data
```

Out[ ]:

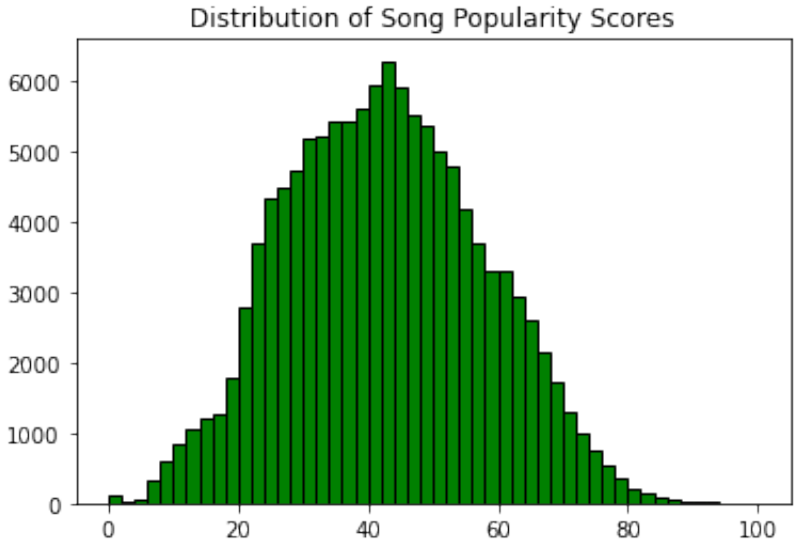
	acousticness	artists	danceability	energy	instrumentalness	liveness	nan
2054	0.7970	['Sam Cooke']	0.332	0.394	0.000021	0.1790	Swe Leila
2055	0.9660	['John Hughes', 'Frank Asper', 'Mormon Taberna...	0.311	0.105	0.000000	0.1200	Guide U O Th Gre Jehovah Voi
2056	0.7490	['Frankie Avalon']	0.474	0.220	0.000006	0.1540	A B Without G
2057	0.6580	['The Everly Brothers']	0.623	0.333	0.006890	0.2610	Memori Are Ma of This Remaster Versi
2058	0.9730	['John Bacchus Dykes', 'Mormon Tabernacle Choi...	0.191	0.138	0.002860	0.0854	Holy, Ho Holy Voi
...	...	...	...	...	...	...	
169904	0.1730	['DripReport', 'Tyga']	0.875	0.443	0.000032	0.0891	Skeche (feat. Tyg - Rem
169905	0.0167	['Leon Bridges', 'Terrace Martin']	0.719	0.385	0.031300	0.1110	Sweet (fea Terra Marti
169906	0.5380	['Kygo', 'Oh Wonder']	0.514	0.539	0.002330	0.1080	How Wou I Kno
169907	0.0714	['Cash Cash', 'Andy Grammer']	0.646	0.761	0.000000	0.2220	I Found Y
169908	0.1090	['Ingrid Andress']	0.512	0.428	0.000000	0.1050	Mo Hear Than Mi

121656 rows × 12 columns

Response Variable Distribution

In [ ]:

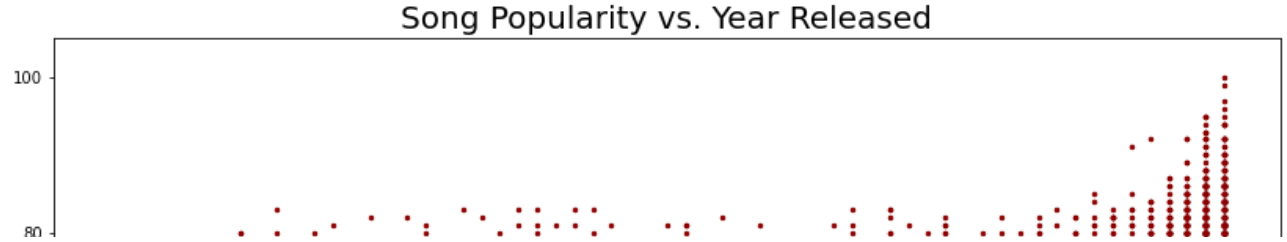
```
# using only songs from 1960+ allows response variable to have a more normal
plt.hist(data.popularity, bins=50, color="green", edgecolor= "black")
plt.title("Distribution of Song Popularity Scores")
plt.show()
```

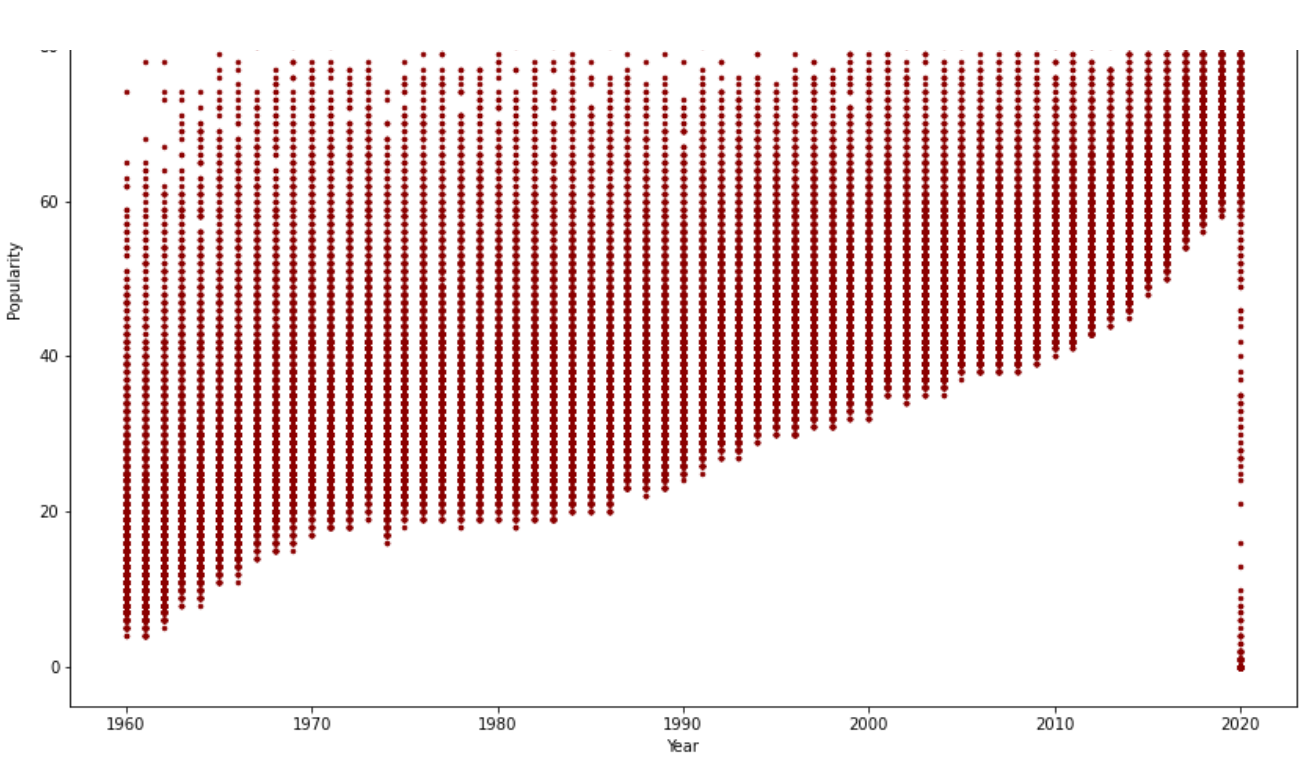


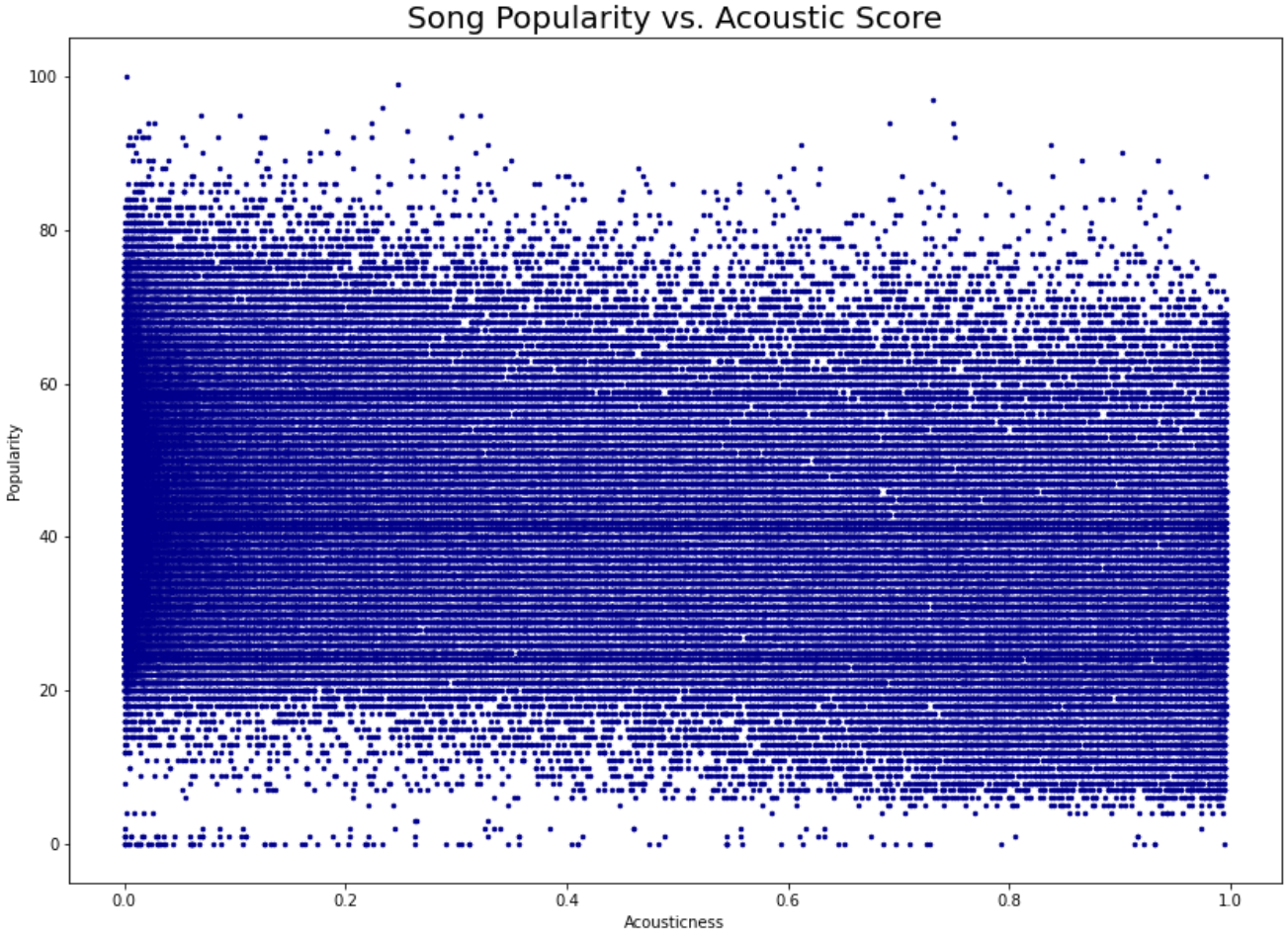
Features Correlated with Popularity

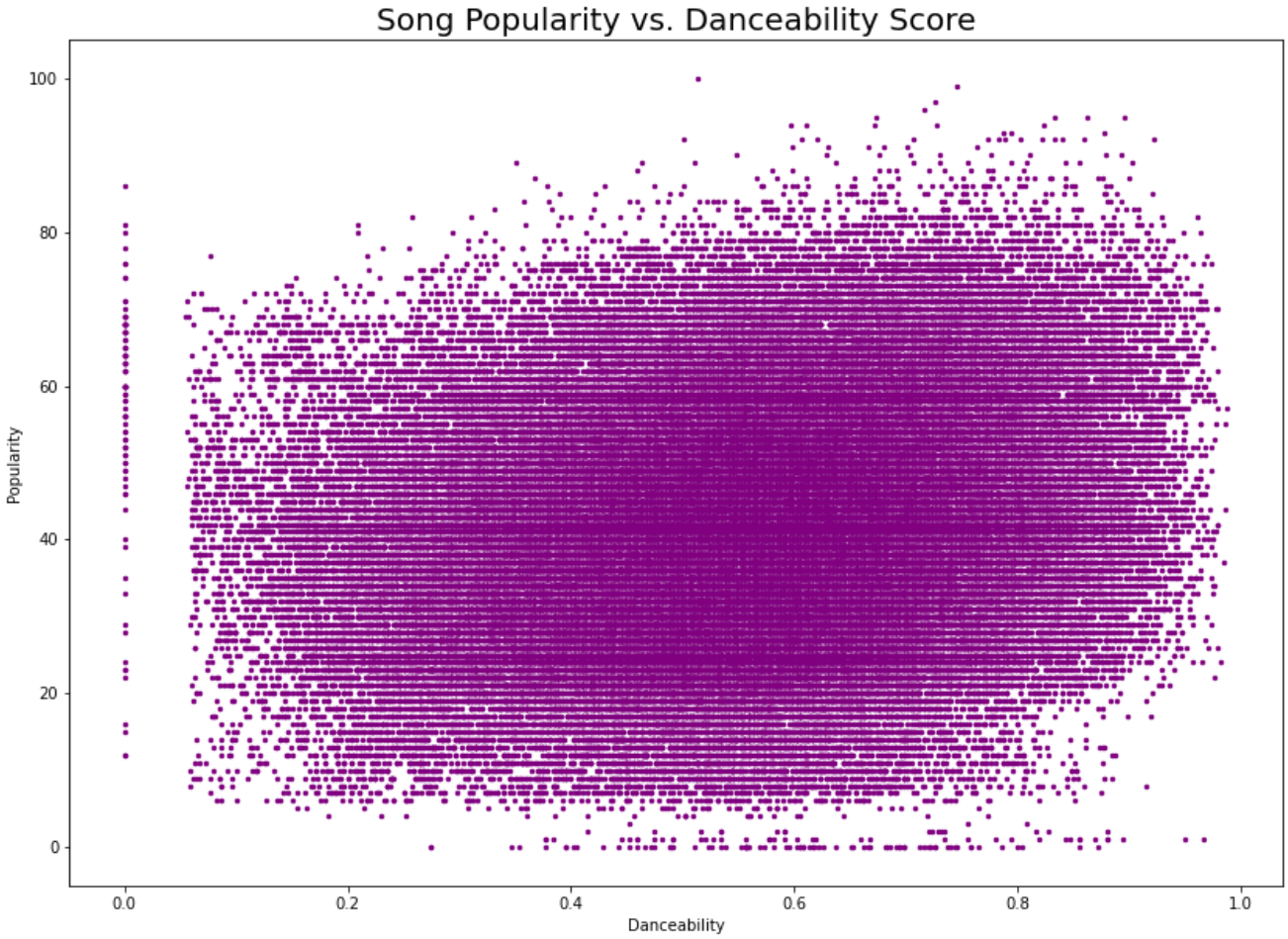
In [ ]:

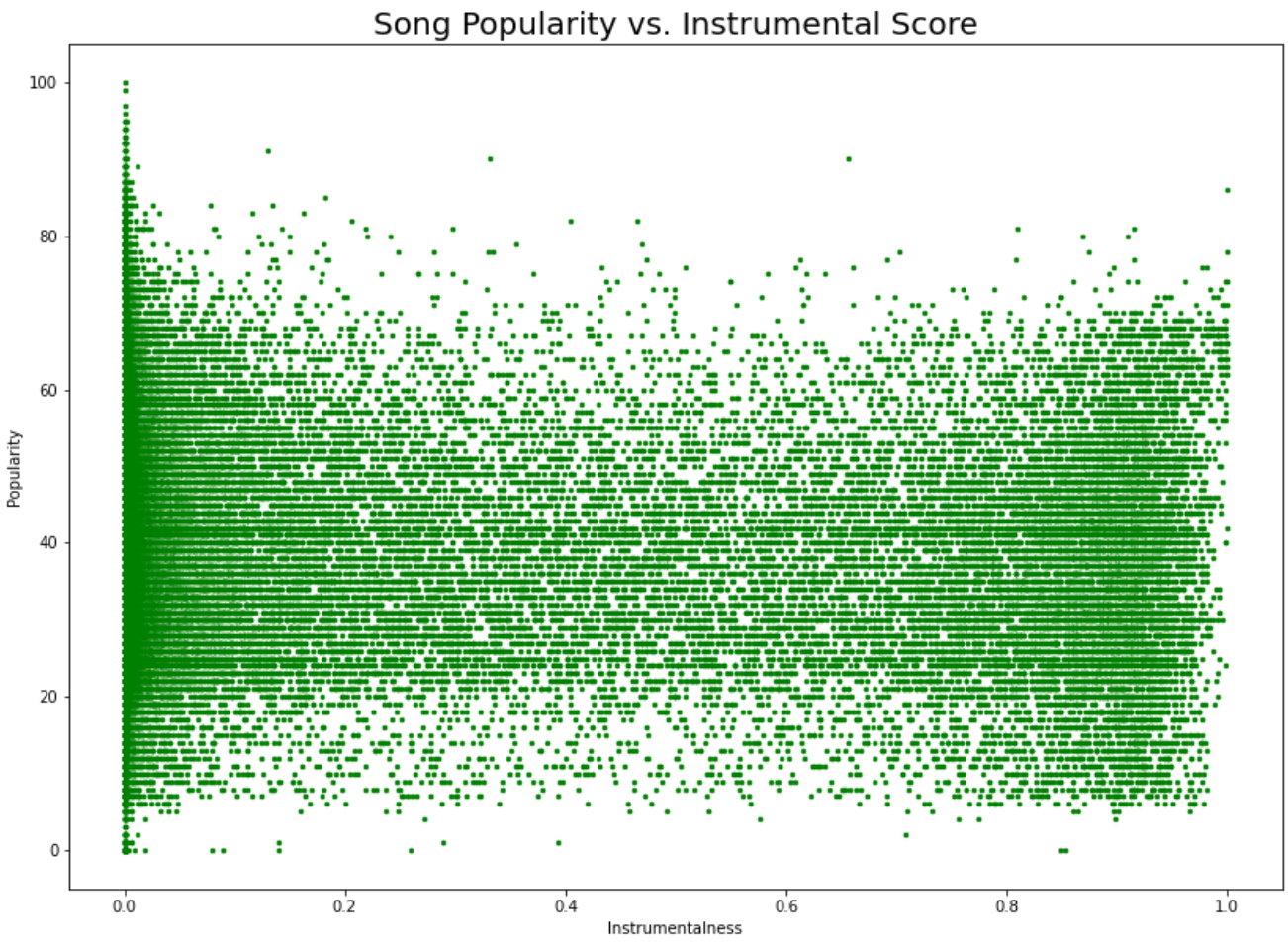
```
# plots to predict popularity
# predictors to explore: year, acousticness, danceability, energy, instrument
plt.figure(figsize=(14,10))
plt.scatter(data.year, data.popularity, s=6, color="darkred")
plt.xlabel("Year")
plt.ylabel("Popularity")
plt.title("Song Popularity vs. Year Released", size=20)
plt.show()

plt.figure(figsize=(14,10))
plt.scatter(data.acousticness, data.popularity, s=6, color="darkblue")
plt.xlabel("Acousticness")
plt.ylabel("Popularity")
plt.title("Song Popularity vs. Acoustic Score", size=20)
plt.show()

plt.figure(figsize=(14,10))
plt.scatter(data.danceability, data.popularity, s=6, color="purple")
plt.xlabel("Danceability")
plt.ylabel("Popularity")
plt.title("Song Popularity vs. Danceability Score", size=20)
plt.show()

plt.figure(figsize=(14,10))
plt.scatter(data.instrumentalness, data.popularity, s=6, color="green")
plt.xlabel("Instrumentalness")
plt.ylabel("Popularity")
plt.title("Song Popularity vs. Instrumental Score", size=20)
plt.show()
```




Popularity
0
20
40
60
1960
1970
1980
1990
2000
2010
2020
Year


Song Popularity vs. Acoustic Score
Popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Acousticness


Song Popularity vs. Danceability Score
Popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


Song Popularity vs. Instrumental Score
Popularity
Instrumentalness
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

The above plots show a song's popularity against the year it was released, its acousticness, danceability, and instrumentalness. The strongest relationship is between the popularity and the song' release year. The later a song was released, the more it appears people enjoy it. This makes sense, because this data was taken from Spotify, which is a fairly new music-listening service. There most likely is not a large population of people that would enjoy music from the 80's and earlier more than later releases.

There appears to be a weak linear relationship in the acoustic plot, where the more popular songs have a lower acoustic score. This opposes the danceability relationship, in which there is a weak, slightly positive relationship shown. People like to dance, unsurprisingly! The most interesting finding here is the instrumental score of a song: there appears to be a quadratic relationship here, where a song is most popular at lower and higher levels of instrumentation.

Due to the sheer volume of data points, it is hard to discern if there is any obvious clustering or outliers in these four plots. There does appear to be very many points on the low end of acousticness, around the 0.6 value of danceability, and the low end of instrumentalness. Notably, there is a set of points on the danceability chart that are exactly zero and have a wide range of popularity, and another set of points on the year-released chart that were all released in 2020 and span a wide range of popularity.

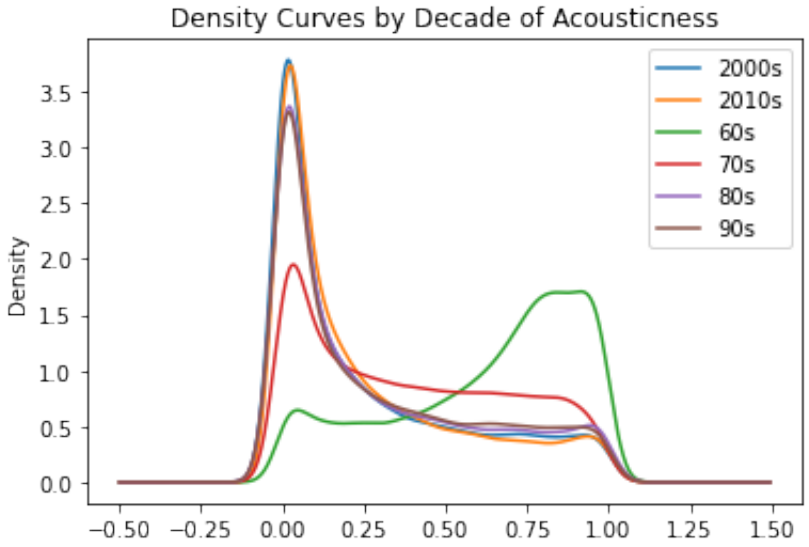
Features Unique to Decade

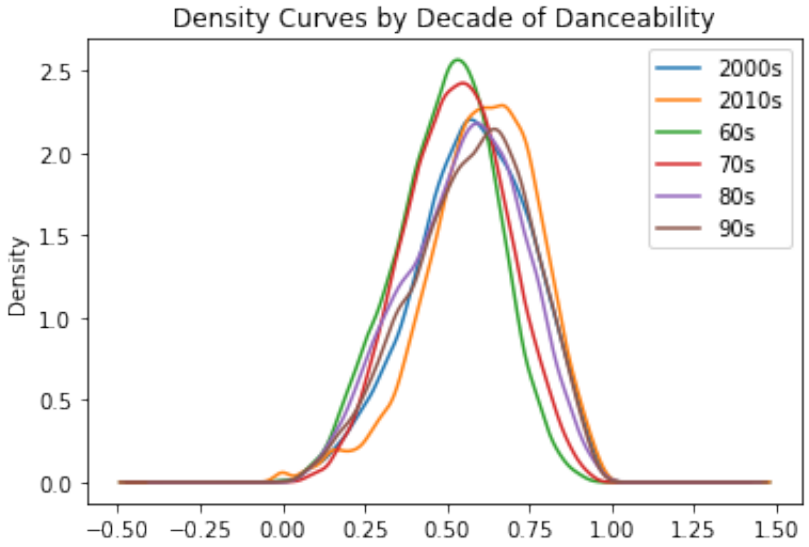
In [ ]:

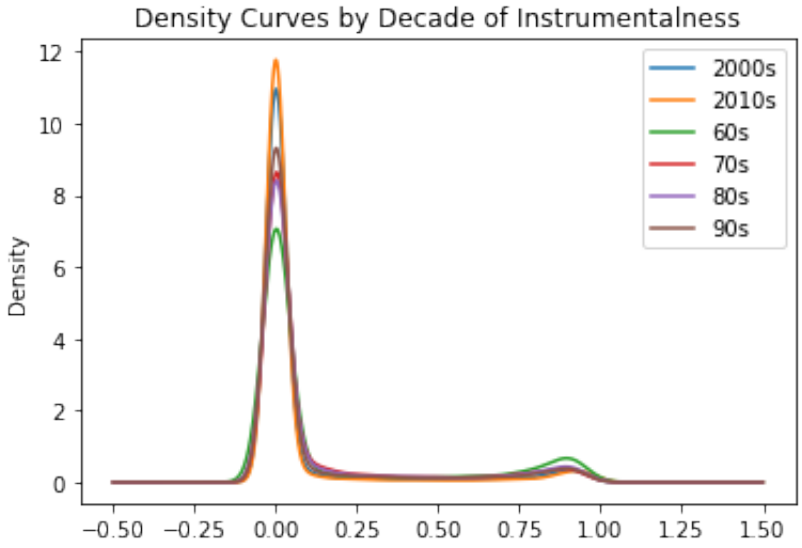
```
# create decade variable for additional year visualization and reshape data f
data["decade"] = np.nan
data.loc[data.year < 1970, "decade"] = "60s"
data.loc[(data.year > 1969) & (data.year < 1980), "decade"] = "70s"
data.loc[(data.year > 1979) & (data.year < 1990), "decade"] = "80s"
data.loc[(data.year > 1989) & (data.year < 2000), "decade"] = "90s"
data.loc[(data.year > 1999) & (data.year < 2010), "decade"] = "2000s"
data.loc[data.year > 2009, "decade"] = "2010s"

acoustic = data.pivot(columns="decade", values="acousticness")
dance = data.pivot(columns="decade", values="danceability")
instrument = data.pivot(columns="decade", values="instrumentalness")
```

In [ ]:

```
# plots of 3 chosen song features, colored by decade
acoustic.plot.density()
plt.title("Density Curves by Decade of Acousticness")
plt.legend()
plt.show()
dance.plot.density()
plt.title("Density Curves by Decade of Danceability")
plt.legend()
plt.show()
instrument.plot.density()
plt.title("Density Curves by Decade of Instrumentalness")
plt.legend()
plt.show()
```


Density Curves by Decade of Acousticness
Density
2000s
2010s
60s
70s
80s
90s


Density Curves by Decade of Danceability
Density
2000s
2010s
60s
70s
80s
90s


Density Curves by Decade of Instrumentalness
Density
2000s
2010s
60s
70s
80s
90s

Of the three feature plots above, the most stand-out observation is that songs from the 1960's have a much higher amount of acousticness than every other decade. This should not be surprising, since that was the decade in which Folk and Americana music became extremely prolific. All decades have similar amounts of danceability, and similar trends in instrumentalness. The instrumentalness plot is interesting to investigate, because it reflects the instrumental scatter plot above: the peak in popularity (and therefore, perhpas, amount) in both low and high scores, with a dip in the middle.

Overall, there unfortunately is not much to be discerned in terms of finding stand out features for each decade. This is potentially a limitation of this dataset because the popularity and other features are measured from the lens of the 21st century. If the songs could have been rated and measured in their own time and context, the results might reflect a more accurate reality.

Conclusion

Not too many strong insights were found in this analysis. Some limitations are the scope of the problem - actual statistical analysis could reveal things that can not be seen in plots - and others are in perhaps the reliability of the data, and the sheer number of observations tend to cloud any meaningful conclusions. I found that the most indicitive looking relationships to predict popularity are a song's release year, acousticness, danceability, and instrumentalness. Of the three musical features, the 1960's have the most marked distinction from the other decades in its acoustic levels. The remaining density plots reflect the patterns shown in the scatter plots and contextual knowledge of music over the decades.