MusiPy: A Personality-based Music Recommendation System

Jennifer M. Wang, Ph.D.
Insight Data Science

INTRODUCTION

- Music is central to human life
- Music is a $15.5 billion industry in U.S.
- Personality traits are associated with music preferences

Big Five Personality

Openness: open to new experiences

Conscientiousness: reliable, disciplined

Extraversion: sociable, outgoing, energetic

Agreeableness: cooperative, helpful, forgiving

Neuroticism: reactive, negative

intense & rebellious

energetic & rhythmic

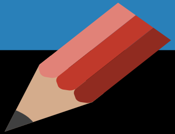
<u>MusiPy</u>: a music recommender based on personality

Either…

- <u>Collaborative filtering</u>
- <u>Content-based filtering</u>
- Cold-start problem

DATA AND METHOD: How was MusiPy created?

Take Big Five Inventory Test



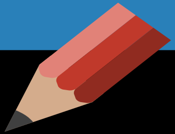
Listen and rate snippets of 50 songs

Personality Project (Michal Koskinski, Stanford University)

n = ~200,000 participants

kNN Regression + Cosine Similarity

Take Big Five Inventory Test



Listen and rate snippets of 50 songs

Content-Based Filtering

+

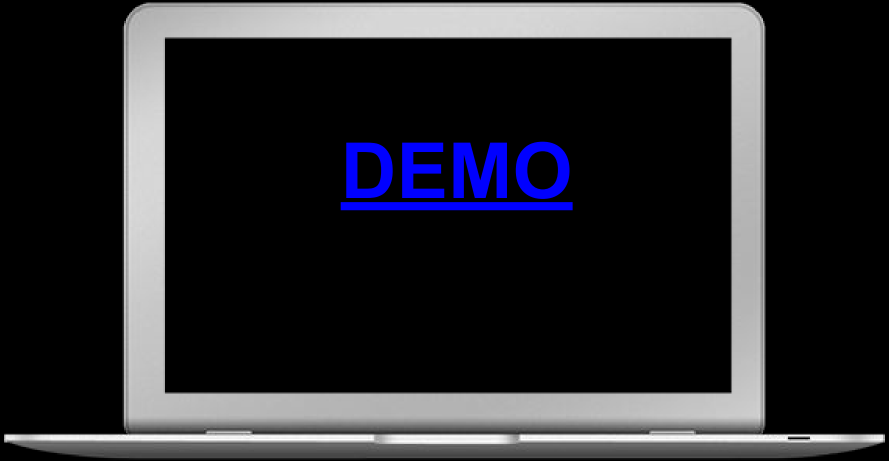
Collaborative Filtering

myPersonality Project (Michal Koskinski, Stanford University)

n = ~200,000

People with these personality traits like these songs

These songs are most like these other songs


DEMO

The Impossible Task of Validating Recommenders:

- Recommendation systems are notoriously hard to validate
- BUT have data on how each user rated each song
    - Can look at how accurate a model is at predicting users' actual ratings

Model:	Predicted Rating:	Actual Rating:
Model A (Baseline)	4	6
Model B	5	6

Split data into training and testing

Baseline

A song's **mean rating** was used to predict users' song ratings

Current

kNN was used to predict users' song ratings

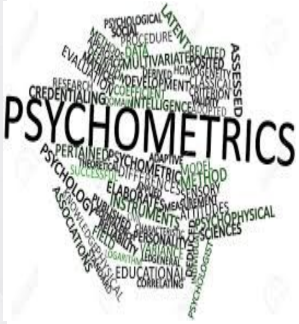
Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE)

Current Model reduced the RMSE score by **~15%**

SUMMARY

- Hybrid models that combine personality and user behavior may be a promising way to build effective and successful recommenders



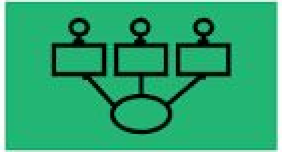
Quantitative Methodology

JENNIFERMADISONWANG@GMAIL.COM

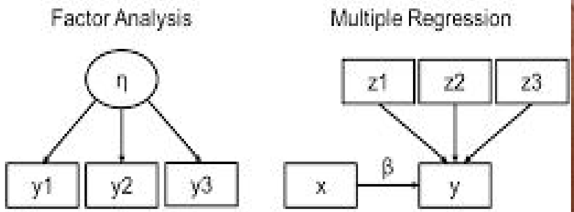
JENNIFERMWANG.COM

IN/JENNIFERMWANGPHD

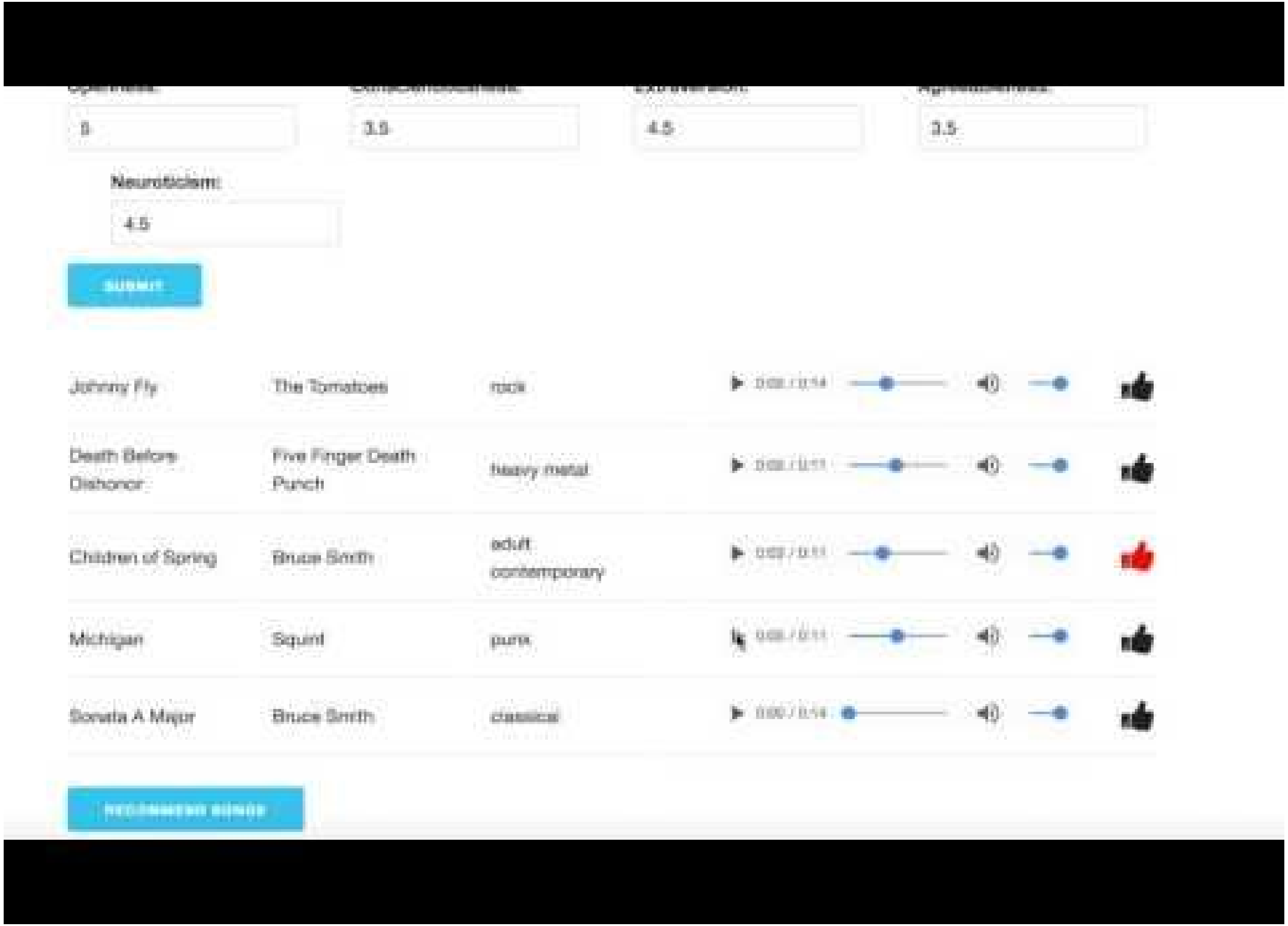
Personality and Relationship Science



Structural Equation Modeling



SUPPLEMENT SLIDES



5
3.5
4.5
3.5
Neuroticism:
4.5
SUBMIT
Johnny Fly
The Tomatoes
rock
Death Before Dishonor
Five Finger Death Punch
heavy metal
Children of Spring
Bruce Smith
adult contemporary
Michigan
Squint
punk
Sonata A Major
Bruce Smith
classical
RECOMMEND SONGS

Item	strongly disagree	disagree	somewhat disagree	neither agree nor disagree	somewhat agree	agree	strongly agree
worries a lot							
gets nervous easily							
remains calm in tense situations							
is talkative							
is outgoing, sociable							
is reserved							
is original, comes up with new ideas							
values artistic, aesthetic experiences							
has an active imagination							
is sometimes rude to others							
has a forgiving nature							
is considerate and kind to almost everyone							
does a thorough job							
tends to be lazy							
does things efficiently							

Your BFI-S scores are as follows (each score can be between 1 and 7):
Write these numbers down for later use.

- **O**penness: 7.0
- **C**onscientiousness: 4.67
- **E**xtroversion: 6.33
- **A**greeableness: 5.0
- **N**euroticism: 6.0

Descriptives:

- Pretty evenly split between genders (male and female)
  - ANOVA: no significant gender differences in relations
- **Personality scores** were normally distributed
  - Skewness and kurtosis were within normal range
  - Women slightly higher on extraversion, neuroticism, and agreeableness
- Song ratings were normally distributed
- Missing data: 7- 10% (FIML)

Descriptives:

Trait	Young adults	Middle Aged adults	Older adults
Openness	4.54	4.49	4.40
Conscientiousness	5.70	5.96	5.83
Extroversion	4.92	4.78	4.61
Agreeableness	5.30	5.37	5.36
Neuroticism	3.98	4.07	4.15

PROCEDURE (“Under the Hood”)

Data from Stanford's **Personality Project** (200,000 participants, Big 5 scores, 50 song ratings)

Content-Based Filtering
(Supervised Learning)

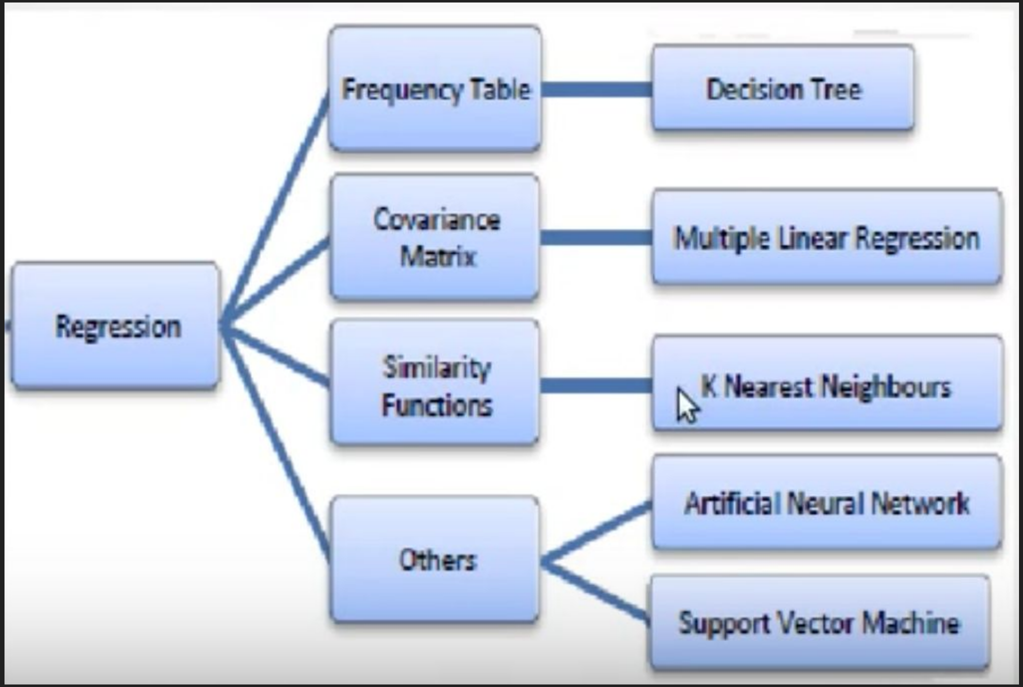
- Distance in personality scores (cosine similarity)
- kNN regression (weights = distance)

Collaborative Filtering
(Unsupervised Learning)

- Distance in **song ratings** (cosine similarity)
- Sort

python
Flask
web development, one drop at a time

kNN Regression



Content-Based Filtering:

- Using SciPy's DISTANCE (cosine similarity) to calculate distance in Personality (see Slide 22)
- **kNN Regression** (Weights = DISTANCE)
    - sklearn's KNeighborsRegressor
    - Using kNN Regression to predict SONG RATINGS
        - Number of k = square root of 200,000 = 445
- Sort

Why k = square root of 200,000?

- Performance of the kNN becomes more stable when using large k. Good rule of thumb is square root.
  - Y. Yang, "An evaluation of statistical approaches to text categorization," Information Retrieval, vol. 1, pp. 69-90, 1999.
  - Y. Yang and X. Liu, "A re-examination of text categorization methods," in Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, 1999, pp. 42-49.
  - M. Jirina and M. J. Jirina, "Classifier Based on Inverted Indexes of Neighbors," Institute of Computer Science, Technical Report No. V-1034, 2008.
  - M. Jirina and M. J. Jirina, "Using Singularity Exponent in Distance Based Classifier," in Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA2010), Cairo, 2010, pp. 220-224.
  - M. Jirina and M. J. Jirina, "Classifiers Based on Inverted Distances," in New Fundamental Technologies in Data Mining, K. Funatsu, Ed. InTech, 2011, vol. 1, ch. 19, pp. 369-387.

DISTANCE Example:

```
# get distances from scipy distance function (ds)
# v is the input value (what the user is putting in)
def get_distances(row, new_row):
    u = [row['ope'], row['agr'], row['neu'], row['con'], row['ext']]
    v = [new_row['ope'], new_row['agr'], new_row['neu'], new_row['con'], new_row['ext']]
    return ds.cosine(u,v)
```

	ope	agr	neu	con	ext	distance
0	4.20	3.15	2.05	4.50	4.25	0.040997
1	4.60	3.10	3.10	2.95	3.95	0.045795
2	4.00	4.45	2.05	3.85	3.10	0.036661
3	2.80	2.60	2.50	3.15	3.25	0.041063
4	4.30	4.65	1.80	3.85	3.10	0.042373
5	4.95	4.10	2.45	3.20	3.95	0.047472
6	4.50	4.00	2.15	2.35	4.05	0.080723
7	4.50	4.30	2.40	2.60	3.20	0.055160

Computes the Cosine distance between 1-D arrays.

The Cosine distance between *u* and *v*, is defined as

$$1 - \frac{u \cdot v}{||u||_2 ||v||_2}.$$

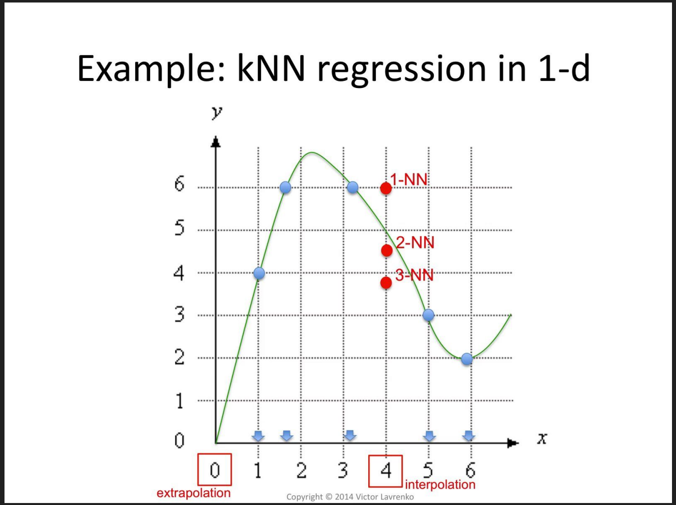
where $u \cdot v$ is the dot product of u and v.

Parameters: **u** : *(N,) array_like*
Input array.
v : *(N,) array_like*
Input array.

Returns: **cosine** : *double*
The Cosine distance between vectors *u* and *v*.

Terminology: kNN Regression

- Calculate the AVERAGE of the continuous/numerical outcomes (Y/dependent variables/ target) of the K nearest neighbors



Collaborative Filtering:

```python
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import manhattan_distances
from sklearn.metrics.pairwise import euclidean_distances
```

	0	1	2	3	4	5	6	7	8	9	...	23929	23930	23931	23932	23933	23934	23935	23936	23937	23938
q1	7	0	9	7	9	6	8	5	9	4	...	0	0	3	9	0	5	6	0	9	0
q2	6	0	2	2	4	1	4	3	4	1	...	0	0	4	9	0	6	6	0	7	0
q3	8	0	1	4	6	1	7	3	9	6	...	0	0	8	8	0	6	5	0	2	0
q4	5	0	1	7	6	1	6	4	3	2	...	0	0	3	6	0	4	8	0	3	0
q5	5	0	4	9	7	6	7	6	3	1	...	0	0	7	8	0	7	5	0	5	0
q6	6	0	1	1	1	1	3	3	9	2	...	0	0	3	3	0	4	5	0	2	0
q7	9	0	7	7	2	2	3	1	6	8	...	0	0	6	1	0	7	4	0	1	0
q8	7	0	1	7	2	1	4	4	2	1	...	0	0	7	3	0	3	5	0	6	0
q9	7	0	1	5	3	5	4	5	4	3	...	0	0	6	5	0	4	5	0	2	0
q10	7	0	1	2	3	7	3	4	2	5	...	0	0	7	1	0	2	1	0	3	0
q11	2	0	1	5	8	5	6	3	8	6	...	0	0	2	7	0	7	6	0	2	0
q12	4	0	9	5	9	2	6	6	8	6	...	0	0	5	9	0	4	6	0	9	0
q13	4	0	1	6	6	1	5	3	7	5	...	0	0	8	8	0	5	7	0	4	0
q14	4	0	1	6	5	6	5	4	5	1	...	0	0	6	8	0	7	6	0	8	0
q15	7	0	4	5	5	5	5	3	2	4	...	0	0	7	2	0	4	7	0	5	0
q16	5	0	9	5	7	5	9	3	2	9	...	0	0	8	2	0	6	7	0	6	0
q17	4	0	9	1	4	8	8	2	2	9	...	0	0	7	3	0	7	7	0	6	0
q18	6	0	9	2	4	9	9	2	4	7	...	4	0	6	3	0	6	7	0	1	0

← Output of the transpose

```python
# cosine similarity
dists_tran = cosine_similarity(transpose1)
dists_tran
dists_tran.shape
```

```
(50, 50)
```

```python
# dataframe for cosine similarity
dists_trans = pd.DataFrame(dists_tran, columns=transpose1.index)
dists_trans
```

	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	...	q41	q42	q43	q44
0	1.000000	0.913570	0.888949	0.889291	0.871537	0.831625	0.784107	0.791378	0.820888	0.767071	...	0.667245	0.650427	0.568199	0.656521
1	0.913570	1.000000	0.882582	0.904338	0.859773	0.830523	0.760426	0.786442	0.809508	0.758269	...	0.641898	0.624437	0.539420	0.628919
2	0.888949	0.882582	1.000000	0.875368	0.881286	0.875570	0.824709	0.829061	0.879841	0.817790	...	0.678303	0.656310	0.584160	0.666670
3	0.889291	0.904338	0.875368	1.000000	0.850975	0.836388	0.756962	0.787442	0.799888	0.757132	...	0.633121	0.613381	0.530197	0.616530
4	0.871537	0.859773	0.881286	0.850975	1.000000	0.819565	0.781702	0.807748	0.832987	0.800810	...	0.660133	0.650011	0.576336	0.656405
5	0.831625	0.830523	0.875570	0.836388	0.819565	1.000000	0.854287	0.829209	0.835619	0.780155	...	0.658704	0.640354	0.570777	0.644231
6	0.784107	0.760426	0.824709	0.756962	0.781702	0.854287	1.000000	0.799392	0.818968	0.769186	...	0.664196	0.655486	0.617649	0.656901
7	0.791378	0.786442	0.829061	0.787442	0.807748	0.829209	0.799392	1.000000	0.839320	0.830543	...	0.596766	0.600225	0.522989	0.594938
8	0.820888	0.809508	0.879841	0.799888	0.832987	0.835619	0.818968	0.839320	1.000000	0.873841	...	0.637585	0.632870	0.569353	0.634982
9	0.767071	0.758269	0.817790	0.757132	0.800810	0.780155	0.769186	0.830543	0.873841	1.000000	...	0.585631	0.588449	0.533745	0.582971
10	0.845260	0.807174	0.800290	0.782563	0.774114	0.783621	0.770559	0.745604	0.764870	0.691408	...	0.639338	0.637218	0.563997	0.633572
11	0.914662	0.876030	0.862134	0.847572	0.839181	0.822778	0.792274	0.782000	0.809973	0.743870	...	0.668409	0.662148	0.579443	0.659829
12	0.890731	0.852424	0.840547	0.829945	0.815978	0.813165	0.787671	0.765590	0.790797	0.724448	...	0.664533	0.658207	0.578156	0.654265
13	0.852649	0.827217	0.833663	0.808067	0.812944	0.813527	0.784704	0.784430	0.799526	0.733558	...	0.647319	0.649712	0.561947	0.639551

```python
# Yes, this one!!!
# If you like 'q4', these are the top 7 you will like
trans_df = dists_trans
new_trans_df = trans_df.sort(['q4'], ascending = False).head(7)
```

	q1	q2	q3	q4
q4	0.889291	0.904338	0.87536	1.000000
q2	0.913570	1.000000	0.88258	0.904338
q1	1.000000	0.913570	0.88894	0.889291
q3	0.888949	0.882582	1.00000	0.875368
q5	0.871537	0.859773	0.88128	0.850975
q23	0.848723	0.852328	0.83841	0.850182
q12	0.914662	0.876030	0.86213	0.847572

Collaborative Filtering:

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import manhattan_distances
from sklearn.metrics.pairwise import euclidean_distances
```

- Cosine Similarity to determine song rating similarity

Validation: Content-based Filtering

- sklearn's mean squared error→ converted to RMSE

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Validation:

	Baseline Model (mean score)	**Current Model** (kNN)
RMSE	0.41	0.35

Future Directions:

- Look at PROPERTIES of songs (tone, tempo, rhythm)
- Adding more songs: only 50 so far
- Sample diverse populations
- **VALIDATION**: user research (can look at novelty, serendipity in addition to accuracy)
    - **Mapping implicit feedback:** length of listening to songs; number of times listen to music; number of downloads
- Adding DISLIKE button

SUPPLEMENTAL READINGS:

- Chamorro-Premuzic et al. (2009). Big Five Personality Traits and Uses of Music
- Delsing et al. (2008). Adolescents' Music Preferences and Personality Characteristics.

Link to FAQ

Link to Code