Star Rating & Score, does it work well enough to supplement each other?

Evening

Goal

If Star Rating (SR) perfectly represents difficulty, we would have a perfect correlation between SR and Score achieved by players. However, this is not true, and we would like to find out what beatmaps are so-called "overrated" and "underrated".

Procedures

Processes	**Details**
Data Retrieval	Get all data via osu!API.
Thinning Data	Removal of "extreme and unnecessary" data points (Refer to **Thinning Data**).
Selection of Players and Maps	Stratified sampling of certain player groups. Then using INNER JOIN with available map data.
Removal of Invalid Records	Removal of: - Double Time Scores - Loved Beatmap Scores
Adding Variable Alpha	Alpha = Score/Star Rating.
Construction of Polynomial Regression (1)	Usage of Polynomial Regression Learner of Max Degree (3).
Adding Variable Regr_Delta	Regr_delta = Estimated Alpha – Current Alpha. *(Prediction Error but with polarity)*
Aggregation of Regr_delta	Using P^2 percentile to aggregate and calculate Regr_delta. Separate Graph fo
Top & Bottom 10 Limit	Limit results for ease of view graph.

Retrieval of players to analyze

Data Retrieval

We retrieve all maps starting from 2015 and retrieve scores from each of these maps for a total of **4621 maps**.

Within each of these maps, we gather results for the **top 100 scores** (limited by osu!API).

Thinning Data

As we are more interested in certain star rating groups, we ignore all data that have:

$$Star\ Rating < 3.0 \ || \ Star\ Rating > 10.0$$

We purposely ignored anything below 3.0 Star Rating as they do not provide interesting insight and doesn't affect results too much. All Top 100 Scores should be close to 1,000,000 in Score in that category.

There aren't any ranked maps that are above 10.0 Star Rating, it's just there to filter out extreme loved maps.

Selection of Players and Maps

We want to analyze are players that currently have scores in **ranked and loved maps**. This is to ensure that we don't analyze "dead accounts".

This has proven to be hard to get a **random sample** as it's hard to avoid "dead accounts" even with recent plays (the API only retrieves a 24h backlog), this method will trim out too many players.

To get the best possible **sample**, we will look at players that have been playing the recent maps, from there, we will generate a **purposive sample**.

By aggregating and counting the number of records the players have in the selected maps, we selected the **top 10%** of the players and **INNER JOIN** (SQL) with the current data.

Removal of Invalid Records

We remove Double Time and Loved Map Records from here:

Why I removed Double Time Records

osu! doesn't provide an updated Star Rating value when you grab it from the API, so it's impossible to calculate **alpha**, which is required later.

Why I removed Loved Records

Loved records greatly skew the data due to some of those maps having little to no player interaction. The regression learner works best if there are a lot of records to estimate to.

Construction of Polynomial Regression Learner

Using KNIME's Polynomial Regression Learner, we use a **maximum polynomial degree** of 3 as the data points seems to suit it quite well.

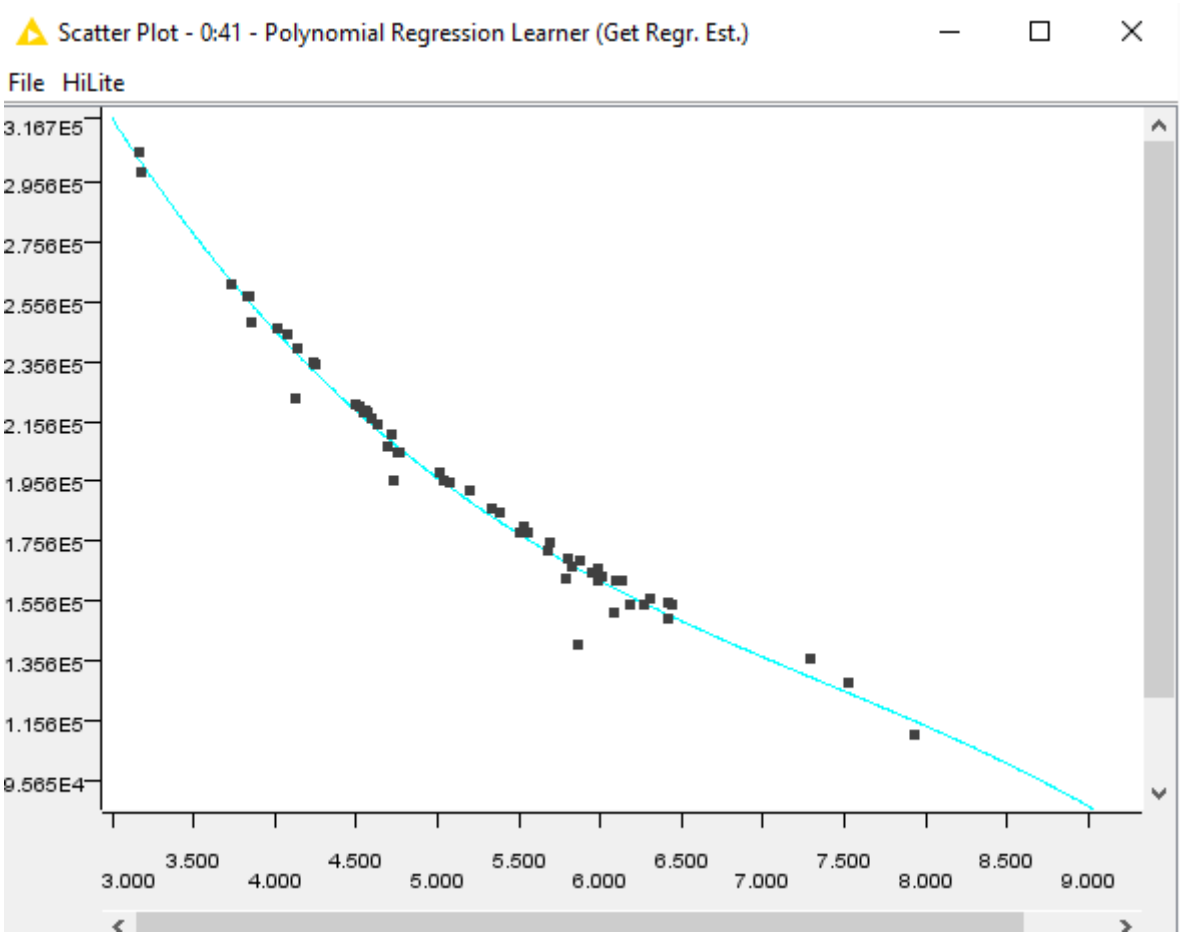

Fig. 1 – Polynomial Regression Learner Scatter Plot of a player.

x-Axis: Star Rating, y-Axis: Alpha

Adding Variable Regr_Delta

$$Regr_{Delta} = Alpha_{estimated} - Alpha_{actual}$$

This is like **PredictionError,** but that variable doesn't provide polarity, so we use this instead.

Aggregation of Regr_delta

We then use the P^2 Percentile Aggregation on all records of each alpha, grouped by each map.

Limit Top & Bottom 10

This is to make it easier to see the more interesting results from this experiment.

Limited Results

Most Underrated and Overrated maps by Polynomial Regression Analysis

keys	artist	title	Regr_delta
4	Quarks (kradness&Camellia)	**Intro + Dualive [Contemplate + Dive]**	84.57%
4	Hitori Tori	**perthed again (yambabom remix) [Extreme]**	89.70%
4	So Sus + Konka	**Acorn [Nuts]**	90.86%
4	Ekcle	**The Impulsive State [Entropy_]**	93.69%
4	Hitori Tori	**perthed again (yambabom remix) [Insane]**	94.65%
4	LunaticSounds	**Dement ~After Legend~ [Evening's EX]**	95.42%
4	KOAN Sound & Asa	**Tetsuo's Redemption [Searching]**	96.43%
4	sakuraburst	**skyshifter vip [us]**	96.83%
4	KOAN Sound & Asa	**fuego (sakuraburst remix) [MX]**	97.16%
4	Sisterz	**Inverse World [Supernova / Arzenvald's Insane]**	97.58%
4	Camellia as "Bang Riot"	**Blastix Riotz [Jinjin's INFINITE]**	101.01%
4	Helblinde	**Memoria (Original Mix) [Memories]**	101.09%
4	MAZARE	**Mazare Party [Ash's 4K Extra]**	101.13%
4	Yooh	**Road To The LegenD, [Thunder of Olympus]**	101.14%
4	LeaF	**NANO DEATH!!!!! [Extra]**	101.18%
4	Kobaryo	**Dotabata Animation [feat. t+pazolite] [Ultra]**	101.29%
4	kamome sano Electric Orchestra	**HE4VEN ~Tengoku e Youkoso~ [HEAVENLY]**	101.39%
4	Colorful Sounds Port	**ETERNAL DRAIN [Eternal]**	101.43%
4	Kaneko Chiharu	**iLLness LiLin [HEAVENLY]**	101.62%
4	Kurokotei	**Galaxy Collapse [Cataclysmic Hypernova]**	101.76%
5	Jaron	**Sonder [Severance]**	88.70%
5	LeaF	**LeaF Style Super*Shredder [SHD]**	94.60%
5	Renard vs Kitsune^2	**The Castle [Fortress]**	96.31%
5	CLIMAX of MAXX 360	**PARANOiA Revolution [Expert]**	97.16%
5	LeaF	**LeaF Style Super*Shredder [SC]**	98.28%
5	Renard vs Kitsune^2	**The Castle [Insane]**	98.44%
5	LeaF	**LeaF Style Super*Shredder [MX]**	98.49%

keys	artist	title	Regr_delta
5	CINDERELLA PROJECT	**Shine!!(you Remix) [Sparkle!!]**	98.82%
5	Camellia	**paroxysm [outburst]**	98.84%
5	TAG	**Theory of Eternity [EXTREME]**	98.89%
5	Gom (HoneyWorks)	**Zen Zen Zense [L.Tay's 5K Hard]**	100.25%
5	LiSA	**rapid life syndrome [Muu's Another]**	100.31%
5	Idol College	**Be My Zombie [5K HD 'Noire']**	100.36%
5	LiSA	**Brave Freak Out -TV ver.- [Kyou's MX Lv.14]**	100.37%
5	Nizikawa	**Isuzu no TRACK [Deep Sea Fleet]**	100.39%
5	Ayane	**Endless Tears... [MX]**	100.44%
5	S-C-U feat. Qrispy Joybox	**anemone [5K Another]**	100.45%
5	Sanaas & Asterisk	**Reliance (Original mix) [Faith]**	100.52%
5	CHiCO with HoneyWorks	**Heart no Shuchou [5K Julie's in Love]**	100.64%
5	Baby's breath	**Habataki no Birthday (TV Size) [HD]**	100.64%

keys	artist	title	Regr_delta
6	DragonForce	**The Warrior Inside [6K Collab Guardian]**	96.95%
6	Gareth Coker	**Restoring the Light, Facing the Dark [Darkness]**	97.14%
6	Ocelot	**TSUBAKI [Camellia]**	98.26%
6	DJ TOTTO VS TOTTO	**Vajra [6K SPECIAL]**	98.41%
6	loos	**Starlight Disco(feat. Meramipop) [Usagi's MASTER]**	98.74%
6	MAZARE	**Mazare Party [JAKARE's 6K Extra]**	98.77%
6	xi	**Happy End of the World [6K Hard]**	99.05%
6	DragonForce	**Symphony of the Night [Rhapsody of the Warriors // 6K]**	99.33%
6	Cranky	**La Campanella : Nu Rave [6K MX]**	99.61%
6	MAZARE	**Mazare Party [6K Hard]**	99.67%
6	Gom (HoneyWorks)	**Zen Zen Zense [6K Hard]**	100.36%
6	S3RL feat Krystal	**R4V3 B0Y [Bray's 6K Insane]**	100.37%
6	Ayumi.	**Hanagoyomi short version [Sakura]**	100.44%
6	S-C-U feat. Qrispy Joybox	**anemone [6K Another]**	100.46%
6	penoreri	**Lord=Crossight [Shin's 6K EXHAUST]**	100.55%
6	xi	**Happy End of the World [Fullerene's 6K Grand Finale]**	100.60%
6	FELT	**In my room [Tidek's 6K Insane]**	100.76%
6	Yooh	**MariannE [Shin's 6K Liberty]**	101.25%
6	WEAVER	**S.O.S. [6K Udon!]**	101.29%
6	MAZARE	**Mazare Party [Ash's 6K Extra]**	101.87%
7	HO-KAGO TEA TIME	**GO! GO! MANIAC (TV Size) [Another]**	91.98%
7	Doin	**Further [Lulu's Hyper]**	92.76%
7	EGOIST	**The Everlasting Guilty Crown [Emotionless Love]**	93.12%
7	m108	*** Crow Solace * [richard's 7K Soluis]**	93.47%
7	D(ABE3)	**MANIERA [Masterpiece]**	95.17%
7	HO-KAGO TEA TIME	**GO! GO! MANIAC (TV Size) [K-ON!!]**	95.24%
7	Camellia	**Bangin' Burst [Rage!!]**	95.35%
7	w_tre respect for AT&HU	**Schur's Theorem [Black Another]**	95.95%
7	Orange Heart(cv:Honda Mariko) Neptune(cv:Tanaka Rie)	**Mousou Katharsis [_UJ's MX]**	96.09%
7	LeaF	**Doppelganger [Zen's Insane]**	96.33%

keys	artist	title	Regr_delta
7	MiddleIsland	**Achromat [7K Black Another]**	101.81%
7	KRUX	**Illusion of Inflict [7K Kruxified]**	101.87%
7	technoplanet	**Inscape [HEAVENLY]**	101.88%
7	kamome sano	**archive::zip [GRAVITY]**	101.88%
7	LeaF	**Alice in Misanthrope -Ensei Alice- [Alice in Wonderland]**	102.03%
7	Chroma	**Hoshi ga Furanai Machi [Meteor Shower // pporse's 7K]**	102.19%
7	Yooh	**Ice Angel [Euphoria]**	102.31%
7	Umeboshi Chazuke	**Panic! Pop'n! Picnic! [Picnic!]**	102.46%
7	Yooh	**LiFE Garden (Extended Mix) [Eutopia]**	102.51%
7	LeaF	**Doppelganger [Alter Ego]**	105.84%

keys	artist	title	Regr_delta
8	xi	**F [I]**	87.53%
8	Ti7	**Love Maker [8K MX]**	89.09%
8	xi	**F [X]**	90.77%
8	Ryu*	**Yukizukiyo [victorica's 8K Lv.10]**	91.92%
8	xi	**F [H]**	92.49%
8	Ryu*	**Yukizukiyo [victorica's 8K Lv.12]**	93.33%
8	Hermit	**Dysnomia [Rayz' 8K Hyper]**	93.34%
8	RADWIMPS	**Yume Tourou [Shana's Hard]**	94.27%
8	Warak	**REANIMATE [Reanimated obj. Kamikaze]**	94.43%
8	Hermit	**Dysnomia [D's 8K Another]**	94.73%
8	ZUN	**Tsukidokei ~ Luna Dial [Lunatic]**	99.54%
8	P*Light feat. mow*2	**Homeneko*Sensation [8K Lv.9]**	99.81%
8	S-C-U feat. Qrispy Joybox	**anemone [8K Another]**	99.89%
8	XeoN	**Xeus [Lv.9]**	100.00%
8	sakuzyo	**Imprinting [8K Lv.12]**	100.09%
8	Ryu feat.Mayumi Morinaga	**Din Don Dan [8K Lv.10]**	100.14%
8	XeoN	**Xeus [Lv.12]**	100.17%
8	Chino(CV.Minase Inori)	**Shinsaku no Shiawase wa Kochira! [Lv.11]**	100.31%
8	XeoN	**Xeus [LV.12 Leggendaria]**	100.31%
8	S-C-U feat. Qrispy Joybox	**anemone [Kawa & Julie's 8K Extra]**	101.70%
9	xi	**Happy End of the World [9K Collab Insane]**	82.08%
9	xi	**Happy End of the World [9K Meteor Shower]**	82.89%
9	CLIFF EDGE	**The Distance feat. Nakamura Maiko [Insane]**	83.76%
9	DJ Mashiro	**Prismatic Lollipops [Lv.20]**	84.62%
9	xi	**Happy End of the World [9K Collab Hard]**	87.85%
9	FELT	**In my room [SitekX's 9K Insane]**	88.55%
9	Tamura Yukari	**MERRY MERRY MERRY MENU... Ne! [Insane]**	89.03%
9	Takamiya Nasuno (CV:Narumi Kyoko)	**MeniMeni ManiMani [EX]**	90.27%
9	Simon Felix	**Trailer! [Monika [MX]]**	92.83%
9	Horie Yui	**PRESENTER (TV Size) [yoshilove's EX]**	96.30%

Personal Thoughts

Is Star Rating a good indication of difficulty?

I think Star Rating does an alright job at handling most maps. However, it starts to produce very inconsistent **alpha** (inversely proportionate to star rating) at maps above 5.0 Star Rating. Since star rating is very closely related to maximum density of the chart itself, it's obvious that it will follow a similar trend to the actual difficulty of the chart.

Trends about the outliers

Lower Regr_delta maps

Usually these maps (data points) have characteristics that make it avoid this **"Star Rating Inflation"** by affecting difficulty increasing density. Here are the top 2 factors that contribute to this:

- Difficult SVs
    - Intro + Dualive [Contemplate + Dive]
    - perthed again (yambabom remix) [Extreme]
    - Acorn [Nuts]
    - and much more…
- Unorthodox Patterning
    - The Impulsive State [Entropy_]
    - Finger VIP [Difficult]
    - Ren Ren Ai Ai Cir Cir Cula Cula Tion Tion [Koi Koi]

Higher Regr_delta maps

On the contrary, maps that have high regr_delta because:

- Easier Patterning for high density sections (Jumptrills/Quads)
    - Galaxy Collapse [Cataclysmic Hypernova]
    - iLLness LiLin [HEAVENLY]
    - ETERNAL DRAIN [Eternal]
- Lack of difficult SVs in general

Fix Star Rating?

I think the best way to tackle star rating calculation is to generate a universal formula using deep learning algorithms, but it'll be a long way from now if no one does it. If it does succeed, this doesn't mean we throw away the idea of the current one, it has proven to be "alright", and if anything goes wrong, at least we can still count on what we have right now.

Annex

Figure 1. Grabs Recent maps from osu!API

```
# Retrieves Maps from osu API
import requests
import json

parameters = {"k": "REDACTED",
              "since": "2018-06-30 11:00:12",
              "m": "3",
              "limit": "500"}

response = requests.get("https://osu.ppy.sh/api/get_beatmaps", params=parameters)
print(response.status_code)

with open('api_responses/response10.json', 'w', encoding='utf-8') as outfile:
    json.dump(response.content.decode('utf-8'), outfile)
```

Figure 2. Grabs specific map data via osu!API

```
# Converts JSON to CSV
import json
import csv

with open('api_responses/out.csv', 'w', newline='') as respcsv:
    csv_file = csv.writer(respcsv)

    for x in range(0, 11):
        with open('api_responses/response' + str(x) + '.json', encoding='utf-8') as resp:
            respJson = json.loads(json.load(resp))

        for map in respJson:
            csv_file.writerow([map["approved_date"],
                               map["beatmap_id"],
                               map["beatmapset_id"],
                               map["difficultyrating"],
                               map["diff_size"],
                               map["hit_length"],
                               map["favourite_count"],
                               map["playcount"],
                               map["passcount"],
                               map["artist"],
                               map["title"],
                               map["version"]])
```

Figure 3. Grabs Top 100 Player Scores from each map

```
# Retrieves Maps from osu API
import requests
import json
import csv
import time

key = open('../../osu_api_key.txt', 'r').read()
ids = open('documents/Selected Maps.csv', 'r').read().split('\n')

for id in ids:
    parameters = {"k": key,
                  "b": id,
                  "m": "3",
                  "limit": "1"}

    response = requests.get("https://osu.ppy.sh/api/get_scores", params=parameters)
    print(response.status_code)

    # Writes to json
    with open('api_responses/map_scores/json/' + id + '.json', 'w', encoding='utf-8') as outfile:
        json.dump(response.content.decode('utf-8'), outfile)

    # Writes to csv
    with open('api_responses/map_scores/csv/' + id + '.csv', 'w', newline='') as respcsv:
        csv_file = csv.writer(respcsv)
        for score in json.loads(response.content):
            csv_file.writerow([score["score_id"],
                               score["score"],
                               score["username"],
                               score["count300"],
                               score["count100"],
                               score["count50"],
                               score["countmiss"],
                               score["maxcombo"],
                               score["enabled_mods"],
                               score["user_id"],
                               score["date"],
                               score["rank"],
                               score["pp"]])

    time.sleep(1) # Prevent spam requests
```

Figure 4. KNIME Data Mining Workflow

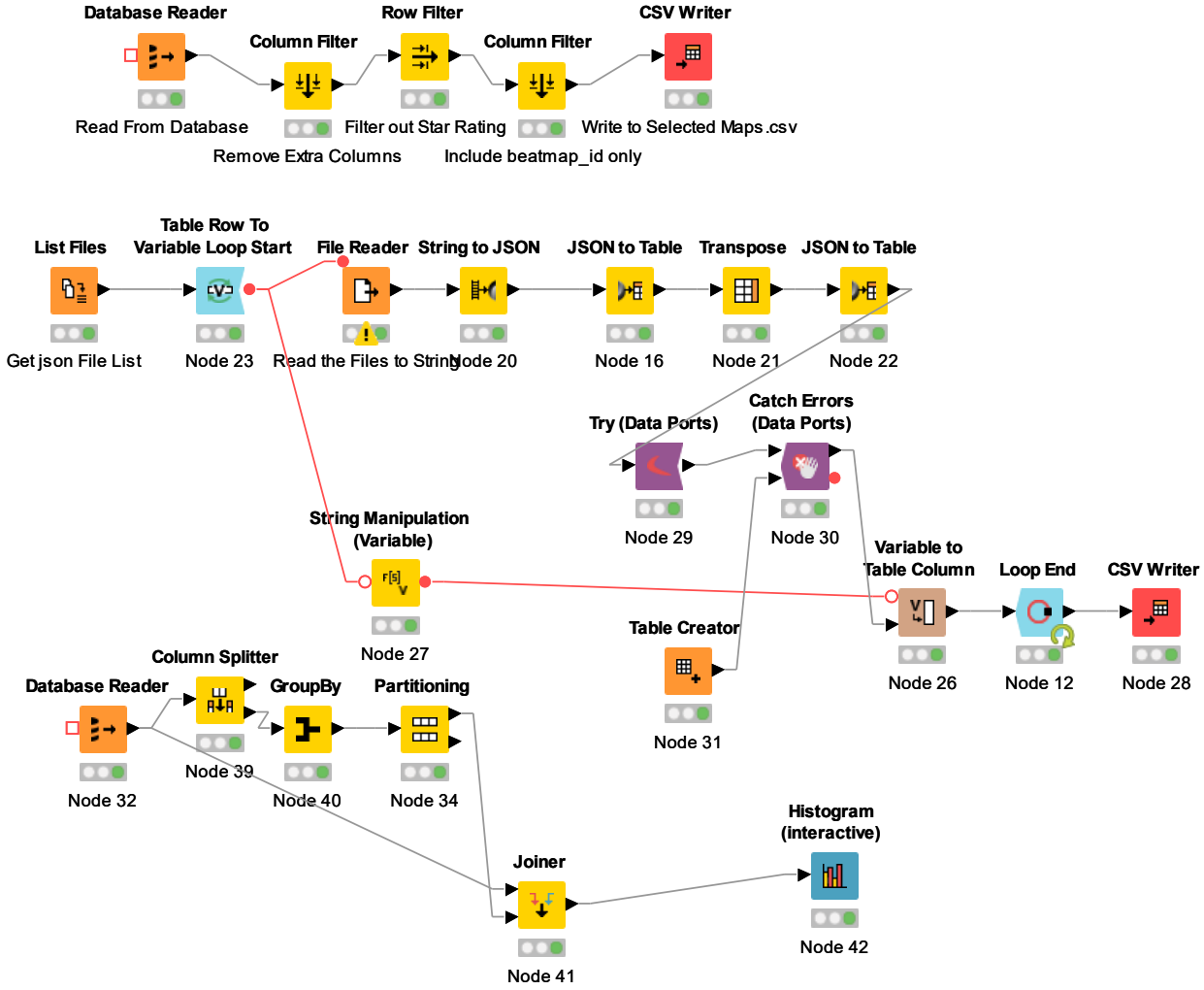

Figure 5. KNIME Data Analysis Workflow