卒業研究報告書

題目

Python を用いたアバタースキンに対する「いいね」数の予測

指 導 教 員 石水 隆 講師

報告者

19-1-037-0142

藤本幹也

近畿大学理工学部情報学科

令和 5 年 1 月 30 日提出

概要

「マインクラフト」などの一部のゲームでは，プレイヤーが自ら作成したアバタースキンを使用してプレイすることができる．また，各自が制作したアバタースキンを投稿する「NameMC」などのサイトも存在し，そこではアバタースキンに対する「いいね」を閲覧することができる．このアバタースキンへの「いいね」は，閲覧者がそのスキンに対して好印象を持つことで付与されるものなので，どのスキンが多くの「いいね」を集めるのかは人間の感覚によるところが大きい．そのためスキン投稿者にとっても，投稿するアバタースキンがどれほどの「いいね」を獲得できるのか予測することは難しい．

そこで本研究では，Python を用いて任意のアバタースキンがどれほどの「いいね」を獲得できるのか予測する AI の開発を行う．十分な精度を持つ AI を作成することができれば，どのようなアバタースキンが閲覧者から好印象を受けることができるのか予測することが可能となり，人間がどのような印象を受けるのか機械で予測できたことになる．

目次

1. 序論

1.1. マインクラフトとは

「マインクラフト(Minecraft)」はマルクス・ペルソン氏と Mojang Studios 社の社員によって開発され 2011 年に発売されたサンドボックスビデオゲームである[1]. 当初は Windows, macOS, Linux 向けに発売されたが, 現在では PlayStation や Nintendo Switch, スマートフォンなどさまざまなプラットフォームに移植され, 全世界で 2 億 3800 万本以上, 世界一の売り上げを誇る大人気ゲームとなっている.

マインクラフトの世界は全てがブロックで構成されており, プレイヤーはブロックを採取・配置または組み合わせることで, マインクラフトの世界を自由に冒険することができる[2]. また, 昨今は Microsoft 社から追加プログラムである「Code connection for Minecraft」[3]も提供されており, マインクラフト内でプログラミングの基礎を学ぶこともでき, 子供教材としての利用にも注目が集まっている.

1.2. アバタースキンの重要性

マインクラフトのプレイヤーは, 1 人称視点と 3 人称視点を自分の好みで切り替えながらゲームをプレイすることができ, 3 人称視点ではプレイヤーの全身を見ながらプレイすることになる. そのため自分が気に入っているアバタースキンを使用することで, より楽しくマインクラフトの世界を冒険することができる.

また, マインクラフトは一人でも遊ぶことができるが, 最大数十人規模のマルチプレイにも対応している. そのためそれぞれのアバタースキン[4]こそがプレイヤーの個性となり, アイデンティティにもなりうるので, いかに自分らしいスキン, 目立つスキンでゲームに参加するかなどプレイヤーの好みが現れやすい部分にもなっている.

1.3. NameMC とは

本研究では, Web サイト「NameMC」[5]から得たデータを機械学習に用いる. 2015 年から運用されている NameMC では, 世界中のマインクラフトプレイヤーが投稿した様々なアバタースキンを閲覧することができ, 気に入ったアバタースキンがあればダウンロードしてマインクラフト上で使用することができる. マインクラフトのアバタースキンは誰でも作成することが可能だが, 色使いや立体的なデザインなど, 満足なスキンを自力で作成することができる人は決して多くない. そのため, デザイン能力に長けた人のスキンの中から自分好みのものを見つけ, Web サイト経由で入手したいという需要があり, NameMC は人気を博している.

また, NameMC には本研究の題材となっている「いいね」機能がある. NameMC の閲覧者はサイト上の無数のアバタースキンを閲覧し, 自身が好印象に感じたものや, マーキングしておきたいスキンを見つけた際に「いいね」ボタンを押す. この「いいね」機能を利用することで閲覧者は, 以前気に入っていたスキンを再発見しやすくなり, また投稿者に対してお礼の気持ちを伝え, スキンを賛美することができる.

またこの「いいね」は, アバタースキン投稿者にとっても非常に大きな意味を持つ. 多くの「いいね」を獲得できるということは, 多くの人から好印象を持たれるアバタースキンを作成できているということの証明であり, 閲覧者から投稿者への評価と直結している. また, NameMC には「トレンド機能」があり, 一定期間で多くの「いいね」を獲得しているアバタースキンのみを閲覧することができる. 閲覧者はこの機能を利用することで, 人気のあるアバタースキンのみを厳選して閲覧することができ, 効率的に良質なスキンを発見することができる. そのため, 投稿者にとっては多くの「いいね」を獲得してトレンド機能に採用されることで, より多くの閲覧者に自分のスキンを閲覧・利用してもらうことが一つの目標となっている.

2. 本研究の目標

2.1. 本研究の背景

前節でも述べた通り, NameMC におけるアバタースキンに対する「いいね」は閲覧者の主観や感覚によって付与されるため, 投稿者が事前にその数を予測することは困難である. しかし, 機械学習を用いてこれを予測することができれば, より多くの「いいね」を獲得できるようにアバタースキンを投稿前に改善できることや, どのようなスキンを投稿すべきなのか判断材料に用いることができると考えられる.

「いいね」の数を予測する研究はすでに行われており，特に SNS を題材にした研究・開発が盛んである．例えば, Twitter 広告における「いいね」数の予測に関する研究[6]や，ファッション系 SNS として有名な WEAR での「いいね」数を予測するアプリの開発[7]などが行われてきた. [6]では，予測対象とするインフルエンサーの過去のツイートを学習データとした深層学習モデルを構築することで「いいね」数の予測を行う手法が提案されている．また，[7]では ResNet[8]の転移学習を行い画像の左右反転などを行うことで学習モデルの汎化性能を高める手法が採用されている．

これら研究・開発の成果として，「いいね」数を学習データとして用いる AI の開発手法はある程度示されたように思えるがその領域は未だ狭く，開拓の余地があるように思える．現状では，アバタースキンに対する「いいね」数を予測する AI は見受けられないため，本研究で取り扱いたいと考えた．

2.2. 本研究の目的

本研究では，マインクラフトのアバタースキン投稿 Web サイトである NameMC を題材として，アバタースキンに対する「いいね」数を予測する AI の開発を行う．アバタースキンに対する人間の持つ感覚を AI で予測することを可能にすることで，投稿者が客観的データに基づいて投稿行為を行えるようにする．

2.3. 本報告書の構成

本報告書の構成は以下の通りである．まず第 3 章で AI 開発の準備となるデータの収集について述べる．その後，第 4 章で AI 開発について述べ，第 5 章で本研究の結論及び今後の課題についてまとめる．

3. データの収集

本章では, Python の機械学習を用いた AI を作成するにあたって行った, NameMC からのデータ収集について述べる．

3.1. 収集すべきデータ

本研究では，教師あり学習を用いて AI の開発を目指す．そのため，アバタースキンとそれに対する「いいね」の数が学習データとして必須である．

また, AI の精度向上を目指すにあたって，ノイズとなるデータは可能な限り排除したい．NameMC には，運用開始の 5 年前から今日に至るまで投稿されてきた様々なアバタースキンが保存されている．図 1 に NameMC に投稿されているアバタースキンの例を示す．図 1 において，数分前に投稿されたアバタースキンと，数年前に投稿されたアバタースキンを比較すると，前者の方が「いいね」の数が少ないことがわかる．この理由を察するに，「いいね」が十分に集まるまでには一定の時間を要するため，投稿されてからの経過時間が格段に短いアバタースキンのデータは特異なデータとなり，機械学習を行う上でノイズとなる可能性が考えられる．そのため，学習に用いるにあたってこれらのデータを排除できるよう，投稿されてからの日数もデータとして収集する．

図 1 経過時間による「いいね」数の差

また, NameMC にアバタースキンを投稿している投稿者にはそれぞれ人気度があり, より良質なアバタースキンを何度も投稿してきた者ほど人気度が高くなっている. 人気の高い投稿者に関しては, その投稿者が作成したスキンであるという理由だけで他のアバタースキンよりも閲覧数が増え「いいね」を獲得しやすくなるため, ノイズ除去のため投稿者の人気度もデータとして活用したいと考えたのだが, NameMC の規約により投稿者のプロフィール情報に関して収集が禁止されていたため断念した.

3.2. データの収集

本研究では Web サイト NameMC から学習データを収集するにあたって, Python の Selenium[9]を用いた. Selenium とは, ブラウザを自動的に操作するためのライブラリであり, 人間がブラウザを経由して操作しているのと同じ動きを実現することができ, 主に Web アプリケーションのテストや Web スクレイピングに利用されている.

本研究での Web スクレイピング(データ収集)の流れは以下の通りである.

1. ランダムにアバタースキンを表示するページにアクセスする.
2. アバタースキン画像が保管されているページの URL と, 「いいね」の数, 投稿されてからの日数をそれぞれセットで入手. CSV ファイルにまとめる.
3. 先ほど入手した URL にアクセスし, アバタースキン画像を保存する.

手順 1 で, ランダムにアバタースキンを表示するページ(https://ja.namemc.com/minecraft-skins/random)にアクセスした際の画面を図 2 に示す. また, このページの HTML の一部を図 3 に示す.

図 2 /random の画面

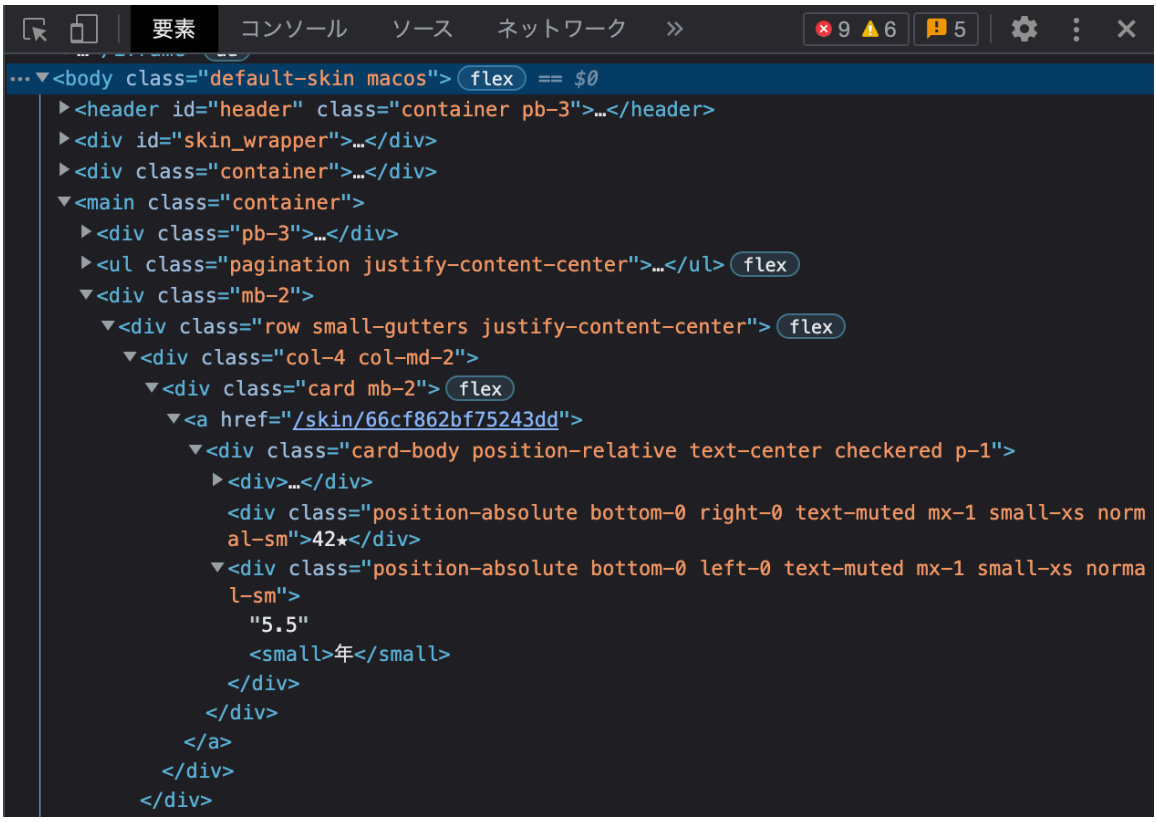

図 3 /random の HTML

図 2 で横に 6 つ並んでいるアバタースキンの内，最も左のアバタースキンに関する情報が図 3 の上から 12 行目，a 要素に格納されている．また href タグで指定されている”/skin/66cf862bf75243dd”が，このアバタースキンの画像が保管されている Web ページの URL である．

図 3 から分かるようにこの a 要素の Xpath は”//main/div/div/div/div/a”であり，「いいね」の数と投稿されてからの経過時間の Xpath はそれぞれ”//main/div/div/div/div/a/div/div[2]”と”//main/div/div/div/div/a/div/div[3]”である．手順 2 ではこれらの Xpath を用いてデータの収集，必要デー

タを CSV ファイルへ書きまとめる.

また，実際のアバタースキン画像が保管されているページは図 2 のページとは異なり，“https://s.namemc.com/i/hoge.png”の形式のページである．ここで,“hoge”の部分は手順 2 で入手した URL の一部であり，今回例示しているアバタースキンの場合”https://s.namemc.com/i/66cf862bf75243dd.png”となる．そのため手順 3 ではこのページにアクセスし，該当のアバタースキン画像をダウンロードする．実際にダウンロードされた画像を図 4 の中央に示す．この画像はアバタースキンを平面で表したものであり，実際のゲーム内や NameMC 上では図 4 の左のような状態で表示される．しかし，NameMC の閲覧者がアバタースキンを前から見た状態のみで「いいね」を押すか決定しているのかは自明でなく，あらゆる角度から見た上で「いいね」を押す判断をしている可能性も踏まえ，全てのアバタースキン情報が含まれる形で学習に利用するため，図 4 の中央の形式で保存・利用した.

図 4 アバタースキン画像

3.3. グレースケール化

本研究ではデータ量を削減するためそれぞれのアバタースキン画像にはグレースケール化[10]を施した上で保存・保管し，学習に利用した．前節で例示したアバタースキン画像をグレースケール化したものが図 4 の右である.

グレースケールとは，画像を色味のない明るさの度合いだけで表現したものである．グレースケールの利点は前述の通りデータ容量の削減であり，カラー画像の 1/3 のデータ量で画像を格納することができる．本研究で取り扱う機械学習では大量の学習データを必要とするため，より多くのアバタースキン画像データを用意できるよう，グレースケール化を採用した.

しかしグレースケール画像には，カラー画像に比べて情報量が減るというデメリットもある．カラー画像では Red(R), Green(G), Blue(B)のカラー情報が格納されているが，グレースケールでは色味を排除し明るさのみで画像を表現することから R, G, B 3 つの値を一つにまとめて格納する．そのためグレースケール化に伴い，カラー画像が本来持っていた情報量の多くが損なわれてしまう.

デメリットはあるものの本研究では限られたリソースの中で AI の開発を実現するため，グレースケール化の採用を決断した．後述する本研究の結果により,AI がある程度の精度を獲得できていることからグレースケール化の影響は限定的であったと考えられるが，カラー画像を用いて学習を行なった方が高い精度の AI を実現できる可能性も高い.

4. AI の開発

本章では，本研究で行った AI の開発について述べる.

4.1. Python による機械学習を用いた AI 開発の手法

Python は機械学習の開発の先駆けとなった言語であり， 機械学習に活用できるライブラリが豊富に用意されている．本研究では，機械学習ライブラリとして定番である scikit-learn（サイキット・ラーン）[11]を用い

て AI の開発を行なった．この scikit-learn には機械学習全般のアルゴリズムが実装されており，適切なアルゴリズムを利用して機械学習を行えるかによって AI の精度が大きく異なる．そのため scikit-learn では，ライブラリの利用者がどの機械学習アルゴリズムを選択するべきかのガイドラインとして，図 5 のようなチートシートが用意されている．

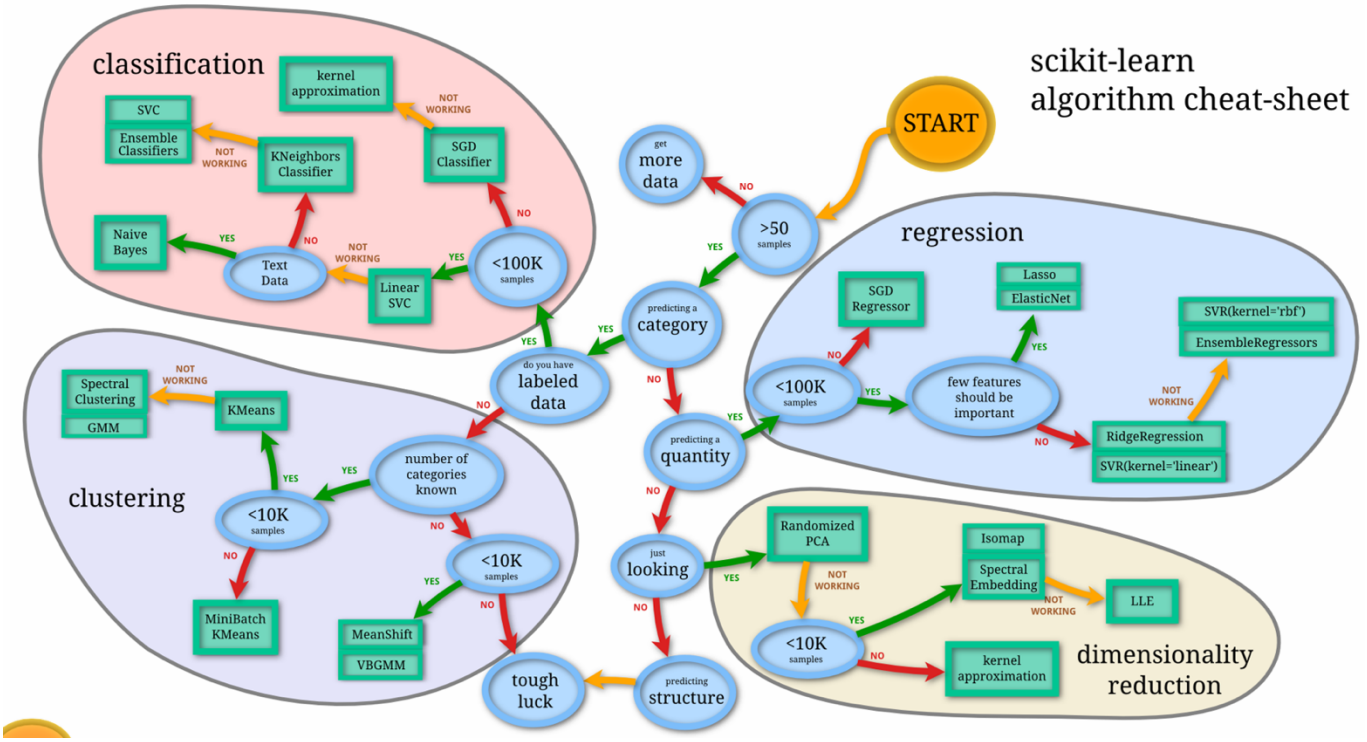

図 5 scikit-learn のチートシート[28]

図 5 の通り， scikit-learn に用意された機械学習アルゴリズムは 4 つのカテゴリに分類される．そこで，それぞれのカテゴリの特徴を表 1 に示す．

表 1 チートシートのカテゴリ

カテゴリ	説明
Classification(分類)	ラベルとデータを学習し，データに対してのラベルを予測する．
Regression(回帰)	実数値をデータで学習して，実数値を予測する．
Clustering(クラスタリング)	データの似ているもの同士をまとめて，データの構造を発見する．
Dimensionality reduction(次元削減)	データの次元を削減して要因を発見(主成分分析など)するため，または他の手法の入力に使う．

詳細は後述するが，本研究では「いいね」の数に応じて学習データにラベリングを施し機械学習に利用する．そのため，図 5，表 1 による Classification(分類)カテゴリの機械学習アルゴリズムを選択すべきであることがわかる．

4.2. サポートベクトルマシン(SVM)

本節では， 分類を行う機械学習アルゴリズムの一つであるサポートベクトルマシン(SVM)について述べる．SVM を説明するためには，ハードマージン SVM とソフトマージン SVM について触れる必要があるため，ま

ずはハードマージン SVM について説明する．ハードマージン SVM は識別平面で学習データが完全に分類できる問題で使用される．識別平面の作成方法には損失関数などは使用せず，代わりに識別平面からそれぞれのクラスの最も近い点までの距離が最大となるように識別平面を設定する．これをマージン最大化と呼ぶ．

しかし識別平面で学習データを完全に分類できるとは限らず，その場合は識別が上手くいかないデータに対して損失関数を定義し，その損失を考慮して SVM を行う必要がある．これがソフトマージン SVM である．損失関数には hinge 関数や log 関数があり，この各データの損失の大きさを求めた変数をスラック変数と呼ぶ．また，損失関数をどれほど考慮するのかを変数 C でコントロールすることができる．変数 C が小さければ損失を許し，C が大きいほど損失を大きく計上するため，変数 C が無限であればハードマージン SVM となる．

同じ Classification(分類)カテゴリの機械学習アルゴリズムに SGD Classifier があるが，SGD は外れ値があったときにその点が大きな誤差になり多大な影響を及ぼす．一方で SVM はマージン最大化が主であり識別平面近傍の点で決定されるため，外れ値に影響されにくいという違いを持つ．

4.3. 機械学習アルゴリズムの検証

前述の通り， 精度の良い AI を開発するためには適切な機械学習アルゴリズムの選択が重要となる．そのため本節では，Classification(分類)を行う機械学習アルゴリズムをそれぞれ試行し，各アルゴリズムでの AI の精度を検証する．

また，検証の前提条件は以下の通りである．

- ✓ データは全ての検証で同じ，グレースケール化したアバタースキン画像データ 18,540 個を用いる．
- ✓ アバタースキン画像データには，後述するラベリングを施した状態で検証する．
- ✓ 特異データに関しては，後述するデータクレンジングを行なった後の状態で検証する．
- ✓ 学習データとテストデータの分け方は全検証で同じく，データの 75%を学習データ，25%をテストデータに用いる．また，学習データ・テストデータに割り振るデータの仕分け方も同一のものとする．

表 2 アルゴリズム検証の結果

アルゴリズム	精度
Linear SVC[12]	89.15%
SGD Classifier(hinge 関数)[13]	89.56%
SGD Classifier(log 関数)	78.94%
Kernel approximation(RBFSampler)[14]	92.38%
Kernel approximation(AdditiveChi2Sampler)	92.38%
KNeighbors Classifier[15]	91.39%
SVC[16]	92.38%

検証の結果は表 2 の通りである．Linear SVC や SGD Classifier では約 89%の精度であり，SGD Classifier では使用した損失関数によっても精度が大きく異なる結果となった．それに対し Kernel approximation や KNeighbors Classifier, SVC では 92%近い精度を得ることができた．Kernel approximation と SVC の精度の結果が全く同じ数値となったのは，検証に用いたテストデータが 4,635 個と少ないため偶然発生した現象である．データを学習データとテストデータに割り振る際の乱数を変更して何度か検証した結果，SVC で 0.9264%，Kernel approximation で 0.9266%など，Kernel approximation が僅かに上回るケースも存在したが，誤差の範囲内であった．

また図 5 の scikit-learn のチートシートに照らし合わせて，SVC の精度が高くなることは想定していたが，データ数が 10 万個以上要求される Kernel approximation でも高い精度を得ることができたことは想定外であった．このことからチートシートは非常に参考になるが，その他の機械学習アルゴリズムも検証する価値があると分かった．

またこの結果とチートシートの適性から，本研究では SVC を用いて AI の開発を進めた．

4.4. データのラベリング

本研究では，アバタースキン画像データに対し「いいね」の数に応じたラベリング[17]を行なった．ラベリ

ングとはデータに正解となる分類情報を付与することで，いわばAIモデルの「お手本」となる振る舞いを用意する重要な作業である．機械学習によるAI開発を成功に導く上では，重要な要素に高品質なデータ収集と適切なラベル付けが挙げられるほど，ラベリングの影響は大きい．データラベリングアノテーションとも呼ばれ，例えば以下のような作業が該当する．

- ✓ 画像中から「自動車」「人」「自転車」の領域に外接矩形を割り当てそれぞれに相当するラベル付けを行う．
- ✓ 道路情報の画像の中から背景，道路，動くオブジェクトを塗りつぶしてラベル付けを行う．
- ✓ 動画の各フレーム単位で，人物の関節およびそれらを結んだ線分(ボーンデータ)をポインティングする．
- ✓ 人物の顔の画像をN種類の感情タイプに分類する．

人間と同様，間違った「お手本」で学習したAIは期待と異なる振る舞いをするようになるため，いい加減なラベリングを施されたデータを「お手本」として学習させれば回数を重ねても期待した精度に達しないことが考えられる．そのため本研究では，どのようなラベリングを行えば十分精度の良いAIを実現することができるのか調査した．この調査の結果を表 3に示す．この調査にはアルゴリズム検証の際に使用したアバタースキン画像データを使用し，機械学習アルゴリズムにはSVCを用いた．また，理由は後述するが本調査では901以上の「いいね」を持つアバタースキン画像データは排除しているため，調査における「いいね」数の上限は900である．

表 3 ラベリング調査の結果

ラベルの付け方	ラベル数	精度
「いいね」数をラベルにそのまま採用.	900	1.19%
0~72, 73~900. (中央値)	2	54.96%
0~55, 56~104, 105~900.(三分位範囲)	3	40.08%
0~48, 49~72, 73~135, 136~900.(四分位範囲)	4	32.18%
0~449, 450~900. (値域を 2 分割)	2	96.40%
0~299, 300~599, 600~900. (値域を 3 分割)	3	92.38%
0~224, 225~449, 450~674, 675~900.(値域を 4 分割)	4	88.35%

表 3の結果から，同じデータ，同じアルゴリズムを用いてもラベルの付け方によってAIの精度が大きく異なることがわかる．また，同様の形式でラベルを付与する場合，ラベル数が少ないほど精度が高く，ラベルの数が増えれば精度が落ちることがわかった．これは単純に分類するべきラベルが増えることでデータの細かい差異を検出する必要が増し，分類の難易度が上がることが原因だと考えられる．

また，三分位範囲などを用いたラベル方法に比べて値域を均等に分割する方法が高い精度のAIを実現できていることが見て取れる．これは第1三分位範囲に分類されるか第2三分位範囲に分類されるのかの差が数十の「いいね」によって左右されるため，その細かな差をAIが識別できない，もしくはデータ自体にあまり差がないことが考えられる．先述の通り，本研究では投稿者の人気度などをAIに反映できておらず，例えば全く同じアバタースキンを人気のある投稿者と無名な投稿者が投稿した場合，スキン画像は同じであるにも関わらず「いいね」の数が数十程度変化することも考えられる．また，NameMCへのアクセス数が多い期間に投稿されたアバタースキンとそうでない場合でも，数十程度であれば「いいね」の数が変動する可能性もある．本研究で作成したAIではこれらの外的要因を考慮できないため，細かな「いいね」の差を捉えることが難しいと考えられる．

また，純粋に精度の高いAIを目指すのであれば「0~899, 900(ラベル数 2)」などでラベリングをすれば，100%に極めて近い精度のAIを作成することも可能である．しかし，こういったAIを使用することで得られる効果は無く，全く意味のないAIとなってしまう．そのため本研究では，四分位範囲や値を均等に分割した場合など，AIとして用いた際のある程度の効果，実用性を持つラベルでのみ調査を行なった．この観点から「0~299, 300~599, 600~900」のラベルは，マインクラフトのアバタースキンを「可・良・優」の三段階で評価できるという効果と，90%以上の精度を持つ実用性を併せ持つことから，本研究で発見できた最良のラベリング方法であると私は考える．そのため本報告書では，前述の機械学習アルゴリズムの検証や後述するデータクレンジン

グに関しても「0~299, 300~599, 600~900」でラベリングしたデータを用いた結果を記載している.

4.5. データクレンジング

AI 開発を行う上でデータの分析や機械学習を進めていくためにはデータを綺麗な形に加工していく必要がある. データを綺麗な形に加工する一連の流れを「前処理」と言い, その処理の一つにデータクレンジング[18]がある. ここで, データクレンジングの対象となるのは以下のようなデータである.

- ✓ 欠損

  「なくてはならない箇所でデータが抜けている」状態. 欠損のあるデータごと削除する, あるいは代表値で埋めることで処理する. データを削除する対処方法ではデータが揃っているデータだけを残して次に進むことが可能だが, 欠損に伴い削除するデータが多くなると今後の学習や分析に支障が出る危険性がある. それに対して代表値で埋める対処法ではデータの平均値や中央値, あるいは最頻値で欠損部を補填することで, データごと削除し学習や分析に支障が出るリスクを抑えることができるが, あくまで代表値であり完璧なデータとは言えない.

- ✓ 表記ゆれ

  「同じものを意味しているがデータの書き方が異なる」状態. 一方で「株式会社〇〇」と表記し, 他方で「(株) 〇〇」と表記している状態などが表記ゆれに当てはまる. これに対しては, 名称の統一や表記ゆれに対する対応表の作成などの対処方法がある. また, 複数の単語が複合してしまっている場合には単語レベルに分割する必要がある.

- ✓ 外れ値

  値の中でも他のものと大きくかけ離れてしまっているもののこと. 他のデータと比較して明らかに不自然であるものは異常値と考え削除してよい. しかし異常値と断言できない場合にはまずは外れ値の検出やヒストグラムなどの手法を使い数値が離れている理由を考察する. 考察の結果異常値と考えられる場合には削除し, そうでなければ対数変換などでデータを修正する対処法をとる.

本研究においては欠損や表記ゆれのあるデータは認められず, またデータ形式の異なるアバタースキン画像データに関してはデータ収集の時点で排除していたため, 外れ値に対するデータクレンジングのみが求められた. 本研究で取り扱う値のデータはアバタースキンが投稿されてからの経過時間と, 「いいね」の数である.

4.5.1. 投稿からの経過時間

第 3 章 1 節でも述べた通り NameMC においては, 数分前に投稿されたアバタースキンと数年前に投稿されたアバタースキンを比較すると「いいね」の数が大きく異なり, 前者の「いいね」が少なくなる傾向がある. この理由としては, アバタースキンが投稿されてからの経過時間が短すぎると十分な回数閲覧される機会を得られず, 「いいね」を獲得しづらいことが考えられる. しかし一方で, 投稿からの経過時間が長ければ長いほど「いいね」の数が比例して増えていくかと考えれば, そうではない. NameMC の閲覧者は, 新規に投稿されたアバタースキン, もしくは直近で多くの「いいね」を集めているトレンド性のあるアバタースキンを好んで閲覧するため, 投稿されてからの経過時間が長すぎるものに関してはむしろ閲覧される機会を失っていくものがほとんどである. そのため投稿されてから一定期間が過ぎれば「いいね」の数が増える機会を失い, 一定の値に収束していくことが考えられる.

このことから, 投稿されてからの経過時間が明らかに不足しているデータに関しては排除する一方で, 経過時間が長いものに関しては制限をかけない方針をとった. また閲覧者が好んで使用する NameMC の機能の一つ「トレンド」[19]では図 6 のように, 経過時間が 30 日未満のアバタースキンが並んでいる. そのため, 経過時間が 30 日に達していないアバタースキン画像に関してはこれらの閲覧される機会を十分に確保しきれていないと考え, 30 日未満のデータを排除して検証を行なった. アルゴリズムに SVC, ラベルには前節で決定したものを使用して検証した結果, データ排除前は精度 90.42%であったのに対して, 排除の精度は 91.04%であった. その差はわずか約 0.63%ではあるが, AI の精度が高まった.

この結果により, 投稿からの経過時間が極端に短いデータを排除する手法は有効であると考えられる.

図 6 NameMC のトレンド機能[5]

4.5.2. 「いいね」の数

NameMC において，多くの「いいね」を集めているアバタースキンには，以下のような理由が考えられる．

理由1.　投稿者のデザインセンスが評価されている
理由2.　投稿者がすでに多くのファンを抱えており，特別多く閲覧される機会を持っている
理由3.　アバタースキンが，既存の人気キャラクターなどに擬えて作られている

理由 1 に属する場合はアバタースキンのデザインでのみ「いいね」の数が決定しているため，収集したアバタースキン画像データのみで評価可能であり，本研究で開発する AI で対応可能だと考えられる．しかし，理由 2 や理由 3 に属する場合は異なる．

理由 2 に属する場合，投稿者の人気度に関するデータがあれば人気度と「いいね」数の相関関係から，一般的な投稿者が同じアバタースキンを投稿した場合の「いいね」数にデータを修正することもできると考えられるが，前述の通り NameMC では投稿者のプロフィール情報に関しては収集が禁止されており入手することができなかった．そのため，どのスキンが理由 2 に属するのか判定することもできず，データを狙い撃って削除することや修正することができない．

また，NameMC には理由 3 に属するアバタースキンも多く見受けられる．図 7 の左は NameMC で最も多くの「いいね」を集めているアバタースキンだが，これはマインクラフトのデフォルトスキンである「Steve」を模したものである．また，図 7 の中央は人気漫画「ONE PIECE」[20]の主人公ルフィを模したスキン，右は同じく人気漫画「NARUTO」[21]の登場人物うちはイタチを模したスキンである．これらはアバタースキンのデザインの良し悪しではなく，元となっている作品やキャラクターの人気によって莫大な「いいね」を集めている．この理由 3 に属するスキンは，特に並外れた多さの「いいね」を集めていることもあり，大きな外れ値となっている．そのため理由 3 に属するスキンのデータは全て排除したいのだが，これは実は非常に困難である．図 7 に示したような，日本でも人気のあるゲームや漫画，アニメのキャラクターであれば私でも判別可能だが，NameMC には海外の利用者も多く，海外作品のキャラクタースキンも多く投稿されている．それらのキャラクターは日本人である筆者には馴染みがなく，判別することができない．また，ゲームやアニメのキャラクターだけでなく，マインクラフトのプレイ動画を YouTube[22]などに投稿している実況者のアバタースキンを模したものも理由 3 に含まれるため，ゲームやアニメの作品名を集めたリストなどを用いたとしても排除は困難である．

図 7 人気キャラクターを模したスキン

これらの理由から，理由 2 及び理由 3 に属するスキンを特定し排除することは非常に困難であり，本研究では実現することができなかった．そのため，「いいね」数の外れ値に関しては，一定以上多くの「いいね」を集めているスキンデータは一律に排除する手法を試行することにした．経過時間に基づくデータ排除を行なった後のデータセットに対して，「いいね」数の閾値を 900 としてそれ以上のスキンデータを排除して学習，検証を行なった．その結果として，「いいね」数 900 以上を排除する前の精度が 91.04%であったのに対して，排除後の精度は 92.38%となった．その差は 1.34%で 1%を超えており，期待以上の精度上昇を得ることができた．

この結果から，極端に多くの「いいね」を集めているアバタースキンデータは，AI に対して大きなノイズとなっておりこれを排除することは有効であると考えられる．しかし，この手法では理由 1 に属する明らかに異常値であるとは認められないデータも排除してしまっている観点から，悪影響も出ている可能性がある．

4.6. 結果のまとめ

様々な試行の結果 92%を超える精度を持つ AI を作成することができたが，この AI の精度はデータへのラベリング方法によって大きく左右されることがわかった．数パターンのラベリング方法では 90%を超える精度を確保できた反面，3 割や 4 割程度の精度に留まったラベルもあった．これは，限定された数パターンのラベルであれば有効な AI を作成することができるが，任意のラベルで十分な精度の AI を実現することはできなかったことを意味する．もし，ある「いいね」の数の閾値で AI を作成するという条件があった場合，本研究で行った手法では実現できない可能性が高い．

またデータクレンジングの結果から，外れ値などを検証し綺麗な形にデータを整える前処理が非常に重要であることがわかった．教師あり学習の場合，どのようなデータを機械学習に用いるかによって AI が期待通りの動作をしてくれるかが決定されるため，良質で大量のデータを用意することが AI 開発を成功させるためには必要不可欠であることを再確認させられる結果となった．

5. 結論・今後の課題

本研究では，Python を用いてアバタースキンに対する「いいね」の数を AI で予測するために，NameMC を題材とした AI 開発を行なった．そして Selenium を用いたデータ収集，アルゴリズムの検証やデータのクレンジングを行うことで，92%以上の精度を持ち，且つ実用性のある AI を開発することができた．また本研究を

通して，最適な機械学習アルゴリズムの選定やデータの前処理が AI 開発を成功に導くために非常に重要であることを再確認することができた．

今後の課題としては，より良質で大量のデータを集めることが第一に挙げられる．本研究では，データ収集への制限やデータクレンジング手法の未熟さにより，データの質が非常に良かったとは考えられない．データの質をより高めるため，異常値は徹底して排除すると共に，明らかに異常値とは言えない外れ値は削除せずに修正するなどすることで，より多くの有効データを残して学習に活用することが求められる．また，本研究では 18,540 個のアバタースキン画像データを使用したが，より多くのデータを収集できるようになれば AI のさらなる精度上昇も見込める．さらにこれらの画像データはグレースケール化を施していたためにカラー画像に比べ情報が少なくなっているため，カラー画像データ情報を保持したまま機械学習に活用することができれば，より高精度で有用性の高い AI を作成できることも考えられる．

また，インターネットへの投稿物に対する「いいね」数の予測は今後も研究が進められていく分野だと考えられ，他のアバタースキンやより多くの SNS 媒体などで AI の開発，または「いいね」数予測 AI を活用した活動などが広がっていくことが期待できると考える．

謝辞

本研究を行うにあたって，また本報告書を作成するにあたって，石水隆講師から多大な御助力，ご指導をいただきました．ここに感謝の意を表します．

参考文献

[1] MojangAB : MINECRAFT 公式ガイド クリエイティブ MOJANG 公式本, 技術評論社 (2018)

[2] 小春 :（連載）ママ・パパおすすめ！「マイクラとは？」がざっくりわかる！これでバッチリ！基礎編, コテエコマガジン, GMO メディア株式会社, 2022 年 12 月 29 日, https://coeteco.jp/articles/10403

[3] Code Connection for Minecraft, Microsoft Studios, (2017) https://apps.microsoft.com/store/detail/code-connection-for-minecraft/9PPFPG2FG2QB?hl=ja-jp&gl=jp

[4] 清水千夏, 渡邊慎二 : アバターの外見と動きとその印象に関する研究, 日本デザイン学会研究発表大会概要集 No.68, pp.214-215 (2021) https://www.jstage.jst.go.jp/article/jssd/68/0/68_214/_pdf

[5] NameMC, https://ja.namemc.com/

[6] 山崎康之介, 牛尼剛聡 : SNS 広告におけるインフルエンサー推薦のための「いいね」数予測, 日本データベース学会, DEIM Forum 2021 I31-3, pp.1-5 (2021)
https://proceedings-of-deim.github.io/DEIM2021/papers/I31-3.pdf

[7] NakaokaRei :【ファッション×テック】WEAR のいいね数を予測するアプリを開発した, Qiita, Qiita 株式会社, 2018 年 12 月 16 日, https://qiita.com/NakaokaRei/items/03dd5587babcc5f772d3

[8] Residual Network(ResNet)の理解とチューニングのベストプラクティス, DeepAge, Spot Inc. , 2016 年 11 月 30 日, https://deepage.net/deep_learning/2016/11/30/resnet.html

[9] selenium-python, GitHub, Inc, https://github.com/baijum/selenium-python

[10] デジタル画像（カラー画像とグレースケール画像）, 有限会社イグノス, (2006) http://www.igunoss.co.jp/imageproc/imageproc1-2.html

[11] Python で機械学習 (scikit-learn), Qiita, Qiita 株式会社, (2019) https://qiita.com/y-tksk/items/1720ca4f8cbc7219148e

[12] Linear SVC(クラス分類) (SVM Classification) 【Python と scikit-learn で機械学習：第３回】, 「機械学習、AI、ディープラーニングのプログラミングを解説」, すぐる, (2017) http://neuro-educator.com/ml3/

[13] SGD(クラス分類)【Python と scikit-learn で機械学習：第１回】, 「機械学習、AI、ディープラーニングのプログラミングを解説」,すぐる, (2017) http://neuro-educator.com/mlearn1/

[14] カーネル近似(クラス分類)【Python と scikit-learn で機械学習：第２回】, 「機械学習、AI、ディープラーニングのプログラミングを解説」, すぐる, (2017) http://neuro-educator.com/ml2/

[15] K 近傍法(クラス分類)(KNeighbors Classifier【Python と scikit-learn で機械学習：第４回】, 「機械学習、AI、ディープラーニングのプログラミングを解説」, すぐる, (2017) http://neuro-educator.com/ml4/

[16] Sklearn.svm.SVC, scikit-learn, scikit-learn 開発者 (BSD ライセンス), (2023) https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[17] AI プロジェクトの成功を左右する「データラベリング」とは, AOS DATA AI データ, (2021) https://aidata.jp/blog/data-labelling/

[18] データクレンジング/データクリーニングとは？重要ワードを楽々理解, AIZINE(エーアイジン), 株式会社お多福 lab, 2021 年 1 月 28 日, https://aizine.ai/data-cleansing-0518/#toc2

[19] マインクラフトのスキン(トレンド), NameMC, https://ja.namemc.com/minecraft-skins/trending

[20] ONE PIECE.com, 集英社, (2023) https://one-piece.com/

[21] 【公式】NARUTO OFFICIAL SITE, 集英社, (2023) https://naruto-official.com/

[22] YouTube, Google, (2023), https://www.youtube.com/

[23] 宇野毅明，武富有香，小林亮太，橋本隆子，久保山哲二，申吉浩：多様性の解析を用いたニュース記事に対するコメント集合の分析，じんもんこん 2022 論文集，pp.207-212，情報処理学会，(2022) http://id.nii.ac.jp/1001/00223185/

[24] 神野悦太郎，北栄輔：画像投稿 SNS におけるハッシュタグの投稿関連度予測，情報処理学会論文誌 数理モデル化と応用（TOM）, Vol.15, No.3, pp.97-105, (2022) http://id.nii.ac.jp/1001/00218969/

[25] 平尾礼央，小町守，岡照晃：美容品レビューのクリック予測に向けたマルチモーダルデータの利用，情報処理学会 研究報告 音声言語情報処理（SLP），Vol.2021-SLP-139, No.3, pp.1-7, (2021) http://id.nii.ac.jp/1001/00213989/

[26] 佐久間唯太，前川知行，柴田遼一，植田有咲，杉浦孔明，今井倫太：対話的画像修正による好み推定に向けたインターフェースの検討, 情報処理学会 研究報告 ヒューマンコンピュータインタラクション（HCI）, Vol. 2022-HCI-200, No.17, pp.1-5, (2022) http://id.nii.ac.jp/1001/00221954/
[27] 津田奏，清水洸希，市野順子：アバターの外見的性別とユーザーの実性別がスキンシップに及ぼす影響:ソーシャル VR におけるフィールドスタディ，情報処理学会 研究報告 ヒューマンコンピュータインタラクション（HCI）, Vol. 2022-HCI-197, No.48, pp.1-7, (2022) http://id.nii.ac.jp/1001/00217359/
[28] Choosing the right estimator, scikit-learn, https://scikit-learn.org/stable/tutorial/machine_learning_map/

付録A データ収集プログラムのソースコード

- **scrap_data.py**

NameMC のスキンランダム表示ページにアクセスし，スキン画像 URL，「いいね」数，経過時間を収集し CSV ファイルにまとめるためのプログラム.

```
from selenium                          import webdriver
from selenium.webdriver.common.by import By
from time                              import sleep
import csv
import datetime

# 保存先ファイル名
csv_file_name = 'new_list.csv'

# 保存先ファイル(オープン)
file              = open(csv_file_name, 'w', encoding='cp932', errors='ignore')

# 保存先ファイルへの書き込みオブジェクト
writer            = csv.writer(file, lineterminator='¥n')

skin_no = 1
while True:
    sleep(1)
    # Chrome を起動
    driver = webdriver.Chrome()
    # URL 取得
    driver.get('https://ja.namemc.com/minecraft-skins/random')
    # ページ内のスキン情報を一つずつ取得
    for elem_a in driver.find_elements(By.XPATH, '//main/div/div/div/div/a'):
        elem_like = elem_a.find_element(By.XPATH, './div/div[2]')
        elem_time = elem_a.find_element(By.XPATH, './div/div[3]')
        url       = elem_a.get_attribute('href').split('/')[4]
        like      = float(elem_like.text.split('★')[0])
        time      = elem_time.text
        # 経過期間は日数で取得, 1 日に満たないものは 0 日とする
        if '日' in time:
            time = float(time.split('日')[0])
        elif '年' in time:
            time = float(time.split('年')[0]) * 365.0
        else:
            time = 0.0
        # CSV ファイルへ書き込み
        csv_line  = [skin_no, url, like, time]
        writer.writerow(csv_line)
        skin_no += 1
        print(skin_no)
    # Chrome を終了(セキュリティ回避のため)
```

```
    del driver
    # 目標数に達したら終了
    if skin_no > 10000:
        break

file.close()
```

- **dawnload_png.py**

scrap_data.py で入手した URL を元に, アバタースキン画像をダウンロードするプログラム

```
from time                                  import sleep
from selenium                              import webdriver
from selenium.webdriver.common.by import By
from webdriver_manager.chrome        import ChromeDriverManager
import csv
import os

# 読み込み先ファイル名
src_file = 'dataset.csv'

# ファイルを'f'としてオープン
with open(src_file, 'r', encoding='cp932') as f:
    for src_data in csv.reader(f):
        sleep(1)
        # ドライバを起動
        driver = webdriver.Chrome(ChromeDriverManager().install())

        # ダウンロードするスキンの URL を取得・アクセス
        url = 'https://s.namemc.com/i/' + src_data[1] + '.png'
        driver.get(url)

        # html からスキン画像を抽出
        img = driver.find_element(By.TAG_NAME, 'img')
        src = img.get_attribute('src')
        # 保存時のファイル名
        png_name = url_tag + '.png'

        # ソースファイルが確認できれば保存
        if src:
            with open(png_name, 'wb') as f2:
                f2.write(img.screenshot_as_png)

        s = 'now dawnloaded: ' + src_data[0]
        print(s)

        # ドライバを終了
        del driver
```

- **gray.py**

グレースケール化を行うプログラム

```
import cv2
import os
import csv

# 読み込み先ファイル名
src_file = 'dataset.csv'

with open(src_file, 'r', encoding='cp932') as f:
    for src_data in csv.reader(f):
        # ファイルパス
        path = 'skin_png_data/' + src_data[1] + '.png'
        if os.path.isfile(path):
            # グレースケール化した状態で読み込む
            png = cv2.imread(path, 0)
            # カラー画像を削除
            os.remove(path)
            # グレースケール化したものを書き込む
            cv2.imwrite(path, png)
```

付録B 機械学習プログラムのソースコード

- **ai.py**

機械学習を行うプログラム. SVC を学習アルゴリズムとして選択した場合を例示. その他のアルゴリズムで実行する場合には変数 model への代入式などが変わるが, ほとんど類似したソースコードなので割愛する.

```
from sklearn.model_selection import train_test_split
from sklearn                  import svm, linear_model
from sklearn                  import metrics
from sklearn.metrics          import accuracy_score
from sklearn.decomposition    import PCA
import matplotlib.pyplot as plt
import csv
import cv2
import os
import itertools

# ラベル済みソースファイル名
src_file = 'label_6.csv'
# アバタースキン画像データを格納するリスト
png_data_list = []
# 「いいね」数データ(ラベル)を格納するリスト
like_data_list = []

with open(src_file, 'r', encoding='cp932') as f:
    for src_data in csv.reader(f):
```

```
            #スキン画像のファイルパス
            path = 'skin_png_data/' + src_data[1] + '.png'
            if os.path.isfile(path):
                # 画像を読み込んでリストに追加
                png_data = cv2.imread(path, 0)
                png_data = list(itertools.chain.from_iterable(png_data))
                png_data_list.append(png_data)

                # 「いいね」数のラベルをリストに追加
                like_data_list.append(src_data[2])

# 画像データ，ラベルデータを学習データ，テストデータに分割
# random_state は乱数のシード
png_train, png_test, like_train, like_test = train_test_split(png_data_list, like_data_list, random_state=1)

# 機械学習モデルの選択
model = svm.SVC() # SVC

# 学習
model.fit(png_train, like_train)

# 性能評価
pred = model.predict(png_test)
print(accuracy_score(like_test, pred))
```