Summary report on Machine Learning

Rafał Górniak & Piotr Hynasiński

December 19, 2024

1 Introduction

This report provides an overview of machine learning, its methods, and applications. Machine learning is a branch of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed. It is widely used in various fields such as healthcare, finance, and technology, playing a critical role in data-driven decision-making.

The field of machine learning is mainly divided into three categories: supervised, unsupervised, and reinforcement learning. Each category addresses different types of problems and leverages certain approaches and techniques to solve them. In this report, we focus on some of the most commonly used algorithms within the area of mentioned types, highlighting their features and practical usage.

The paper is divided into six sections. The first five describe, one by one, each of the approaches and functions taken into account, including classification and recommendation techniques. The last section presents results and conclusions as well as our opinions concerning the appropriateness and effectiveness of the described methods.

All the data used in this report comes from the TMDB movie database. As an input for training, we took train.csv containing exactly 90 rows with rating for particular movie for 358 users. The goal was to assess 30 new films for users, being stored in test.csv. Using the TMDB API, we fetched features for each movie, which are discussed in detail later.

2 K-Nearest neighbours

2.1 Overview

The K-NN algorithm is primarily used for classification and regression tasks. The key idea behind it is that similar data points are likely to have similar labels, which is determined based on a distance metric or similarity measure. The parameter k represents the number of nearest neighbors.

2.2 Movie Features

In the implementation of the K-NN algorithm there are two types of features considered: numerical and categorical. Two sets of movie features were tested. In the first set numerical features include release date, runtime, budget, revenue, popularity, vote average and vote count. Genres are the only feature that is categorical. The second set adds more categorical features such as information whether the movie is for adults, lists of spoken languages, production companies and production countries but does not utilize genres.

Numerical features are normalized using Min-Max Scaling to scale their values between 0 and 1. As a result, features with larger values such as budget or revenue are prevented from dominating the calculations of distance when compared to smaller values.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

Min-Max Scaling formula.

Genres are encoded using one-hot encoding.

$$G = \{g_1, g_2, \ldots, g_n\}, \quad v_i = \begin{cases} 1 & \text{if } g_i \text{ is present,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

One-Hot Encoding formula, where the binary vector v represents the presence or absence of genres g_i.

2.3 Feature similarities

For numerical features, the similarity is computed using a normalized Euclidean-based similarity measure. In the case of categorical features, the similarity is calculated using the Jaccard similarity.

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}, \tag{3}$$

Euclidean distance formula, where a and b are feature vectors.

$$\text{Similarity}_{\text{numerical}} = 1 - \frac{d(a, b)}{\text{max\_range}}, \tag{4}$$

Normalized similarity formula for numerical features, where max_range is the max normalized range.

$$\text{Similarity}_{\text{categorical}} = \frac{|A \cap B|}{|A \cup B|} \tag{5}$$

Jaccard similarity, A and B are the binary vectors

2.4 Performance and Evaluation

The performance of the K-NN algorithm was evaluated using cross-validation for different k values. For each k, the accuracy of the model was determined and the k-value that achieved the highest accuracy was selected as the optimal k for each user. The predictions were then made using the optimal k value. The average accuracy was approximately 41% for both feature sets. The prediction was made by aggregating the labels of the k-most similar neighbors, and the label that appears the most frequently among the neighbors was selected as the final prediction.

3 Decision Tree

3.1 Overview

The Decision Tree algorithm is a supervised learning method that recursively partitions data by evaluating features and selecting thresholds that maximize the separation of target labels. They consist of internal nodes that represent conditions on features and leaf nodes that contain predictions.

3.2 Feature Selection and Splitting

The movie features used for the decision tree are the same as in the case of K-NN. The data is split in the decision tree by a measure of Information Gain. Impurity is calculated using Entropy. At each step, the algorithm evaluates possible splits and selects the one that achieves the highest Information Gain.

$$H(Y) = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{6}$$

Entropy formula, where p_i is the proportion of samples in class i.

$$IG = H(Y) - \left(\frac{n_l}{n}H(Y_l) + \frac{n_r}{n}H(Y_r)\right) \tag{7}$$

Information Gain formula, where n_l, n_r are the sizes of the left and right subsets relative to n.

3.3 Stopping Conditions

The tree induction process ends when one of the following criteria is met:
- maximum depth: the tree reaches a predefined depth d.
- minimum samples: the number of samples in a node falls below the threshold min_samples_split.
- homogenity: all samples in a node belong to the same class, resulting in zero entropy $H(Y) = 0$.

3.4 Tree Visualization

The decision tree generates a structure, where splits are based on features. Each internal node represents a decision condition, while leaf nodes indicate the predicted ratings. Figure below presents two trees. The first tree used the first set of movie features (the one with genres), and the latter - second set.

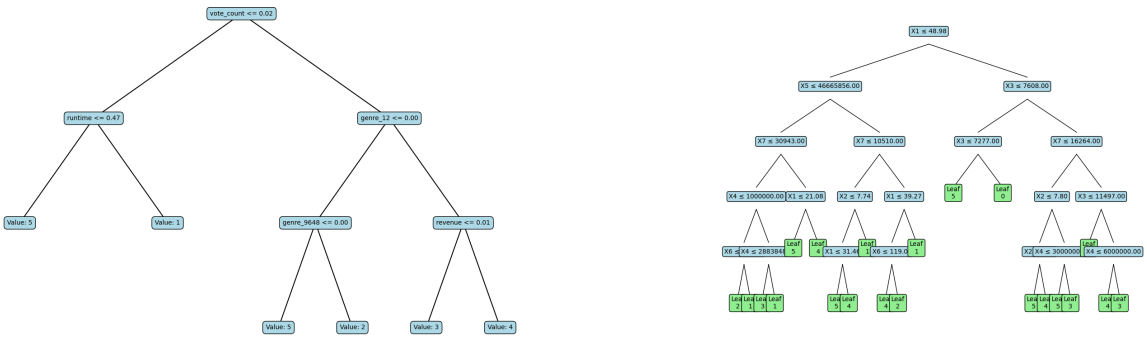

Figure 1: Decision trees made with two different movie feature sets

3.5 Performance and Evaluation

To evaluate the model's performance, hyperparameters such as max depth and min samples split were optimized using cross-validation. The max depth was tested within a range of 2 to 7, and the same range was applied to min samples split. Similar to the K-NN algorithm, the accuracies were consistent across both sets of movie features. However, the decision tree model achieved a lower accuracy of approximately 37%. This lower performance could be caused by the main drawbacks of decision trees: their tendency to overfit the training data, resulting in high variance and reduced generalization.

4 Random forest

4.1 Overview

The Random Forest algorithm is a supervised ensemble learning method that builds multiple decision trees during training and aggregates their output to improve predictive accuracy and robustness. Each tree is constructed using a randomly sampled subset of data and features, introducing diversity in the model. Predictions are made by majority voting for classification or averaging for regression, leveraging the collective decision-making of the forest to reduce overfitting and enhance generalization.

4.2 Building the tree

In the Build the Forest section, Random Forest constructs multiple decision trees using two key techniques: bootstrap sampling and random feature selection. Bootstrapping involves creating a random subset of the training data for each tree by sampling with replacement, allowing each tree to be trained on different data points. In addition, at each split in a tree, a random subset of features is chosen. The number of features selected is typically set as the square root of the total number of features for classification or one-third for regression.

The standardization of the features is performed before training, where each feature is normalized by subtracting the mean and dividing by the standard deviation. This ensures that all features are on a comparable scale, preventing features with larger ranges from dominating the model.

$$z_{ij} = \frac{x_{ij} - \frac{1}{N-1}\sum_{k \neq i} x_{kj}}{\sqrt{\frac{1}{N-1}\sum_{k \neq i}\left(x_{kj} - \bar{x}_{-i,j}\right)^2}} \tag{8}$$

Formula standardize feature j for point i by subtracting the mean and dividing by the standard deviation, excluding i.

$$m = \begin{cases} \sqrt{M}, & \text{for classification} \\ \frac{M}{3}, & \text{for regression} \end{cases} \tag{9}$$

Formula calculates the number of features m randomly selected for each tree split, where M is the total number of features.

4.3 The Random Forest algorithm

Since the decision tree has already been described, along with its loss function, gradient, and entropy, this section focuses on the tree's overall structure and extracting a unified solution from the model.

User data is randomly selected from various movies, and the hyperparameters are tuned to determine the best possible configuration. First, each number of trees, second, the maximum depth, the maximum number of features considered, and the minimum number of samples in the leaf data for the decision to terminate the algorithm. A differentiated choice for each user defines the accuracy.

First of all, bootstrapping and aggregation should be distinguished, one separates the data randomly with insertion and substitution, and the other decides the final label based on a specific formula, it is up to us which one we choose, whether counting the average for regression or the most frequent occurrence for checkout. We chose the second option.

$$\hat{y} = \arg\max_{c \in C} \sum_{i=1}^{n} I(y_i = c) \tag{10}$$

This formula calculates the predicted class $\hat{y}$ in classification. Select the class c that has the highest frequency in the samples reaching a given tree node. The function $I(y_i = c)$ is an indicator function that returns 1 if the true class$\hat{y}$ matches class c, and 0 otherwise.

4.4 Performance and Evaluation

Despite the improved method compared to the previous algorithm, the accuracy increased slightly to 39%, with *n_estimators* and *min_samples_split* proving to be the key parameters.

5 Person similarity

5.1 Overview

The similarity approach is one of the simplest classification solutions. It reduces time and memory complexity by avoiding the need to analyze vector features in detail, yet it can achieve better accuracy than some complex models. There are surely several different approaches within this area, but we would focus on the easiest, and the best one as well.

5.2 Measures and comparison

Probably the most important factor in this algorithm is choosing which two vectors are similar to each other, and which are not. In the case of users having 90 movies already graded it is not so hard. Ways can differ as much as ones ideas, but there are several well-known and regarded as most close.

For this case scenario, I took counting the number of similar or exact note in particular movie. However, the trouble occurs where there is lack of the same films in the dataset. That why the percentage content matters. Person with 10 exact same grades out of 30 same movies would be treated as more similar than a user with 20 same grades from 70 films.

$$S(U_1, U_2) = \frac{1}{|F_{1,2}|} \sum_{i \in F_1} \sum_{j \in F_2} I(r_{1li} = r_{2j}) \tag{11}$$

This formula calculates the similarity between two users, U_1 and U_2, based on their movie ratings. The double summation iterates over all movies rated by U_1 (F_1) and all movies rated by U_2 (F_2). The indicator function adds 1 when the ratings for the same movie i and j from both users match, otherwise, it adds 0. The final similarity score is the average of these matches, divided by the total number of common movies rated by both users.

5.3 Evaluation and testing

Because there was not used any complicated way of computing the similarity, there was no need to trying finding the best possible hyper-parameters. The algorithm sums only exact grading withing two movies in a single comparison. That way, the accuracy could be a little worse, but also the confidence of closer match and also the deviation would be smaller.

After looping over each user for each one, the array with similarities could be defined. It must have been sorted in the ascending order and cut, that the last element, having 100% accuracy (the same user) was removed. In such case, iterating over each users, we got sure that he got labels for not graded movie, from the test dataset. This solution seems to lack of disadvantages, because firstly, we are confident that labels for more similar users would overlap these of less ones. Secondly, every user is taken into consideration, and in hypothetic situation, all users does not have one peculiar movie graded, but "the worst" person have. So despite probably wrong label that would be associate with current user, he would get the mark.

5.4 Performance and outcome

To our surprise, the person similarity method achieved the highest accuracy among all the algorithms tested. It overpowered the model learning mechanisms. The cause of this can be found in true labeling and tendency, rather then complex calculations and anticipation. My prediction were evaluated for around 55%, what with six possible options to choose from is a good score.

6 Collaborative filtering

6.1 Overview

The collaborative filtering approach is a method used in recommendation systems that relies on the patterns of interactions between users and items. Instead of analyzing the content or attributes of items, it focuses on leveraging the collective preferences and behaviors of a group of users to make predictions. By identifying similarities in user behavior or item interactions, collaborative filtering aims to suggest items that a user might like based on the preferences of others with similar tastes.

This approach can be applied to various type of data. In this case, it focuses on vector presenting movie details and person parameters, that are adjusted in the runtime. There are mainly three utilized types of algorithm, for having no information of details and users, and for having one or another. To avoid over-complexity and bring up the core of formulas, we comply to finding user parameters with already vectored movies, that is having features in them.

6.2 Matrices and Methods

The typical way to represent the entire dataset is by building a multi-dimensional matrix. In our solution there are eight movie features, as mentioned in earlier chapters, and then there are also eight parameters for each user. Every movie is either graded or have a null value for rating. The algorithm for mark prediction is coherent, but simultaneously not comprehensive. Initially user parameters set with default value equal zero, are with the epochs iteration changing. For each loop block there is firstly gradient calculation and then update of parameters for current person. The size of step the parameter gets increased or decreased is determined by *etas* parameter.

$$\hat{y}^{r,m} = f(\mathbf{x}^m, \mathbf{p}^r) = \sum_{i=1}^{n} p_i^r x_i^m + p_0^r \qquad (12)$$

Prediction formula: It estimates the rating $\hat{y}^{r,m}$ that user r would give to movie m by combining the user's parameters and the movie's features.

$$Q(\mathbf{p}^r) = \frac{1}{2} \sum_{m:e^{r,m}=1} (f(\mathbf{x}^m, \mathbf{p}^r) - y^{r,m})^2 \qquad (13)$$

Cost function: calculates the squared error between the predicted ratings and actual ratings for all movies rated by the user.

6.3 Parameters and Tests

In order to obtain the best possible hyper-parameters for each user, matrix was not purely thrown into the algorithm, because in that way, one possible combination of parameters would be applied to everyone. In this solution every user can have different variables, that suits the most. As testing it via cross-validation concept and obtaining certain accuracy scores, each person got different values. But the average number of epochs oscillates around 500, so pretty huge number. And the values of etas around 0.005, so quite little learning step. The value calculated thank to formula above was float with enormous range, so to adjust it to output, formula must have been used.

$$\hat{r} = \max(0, \min(5, \text{round}(\hat{y}^{r,m}))) \qquad (14)$$

6.4 Performance

The outcome grades and the accuracy were in range of good. Comparing it to previous ones, it come out to be on average very similar. Despite fluctuations among user, where one could have accuracy over 90 and other below 20, the mean we got was 39%

7 Summary and results

This section summarises the results and observations associated with testing of various machine learning algorithms. Each approach has been evaluated through its predictive accuracy.

1. K-Nearest Neighbors (K-NN)

- **Accuracy:** 41%

K-NN achieved decent performance, with its accuracy reliant on the choice of similarity measures and the optimal number of neighbors (k). It handled both numerical and categorical features using normalized Euclidean and Jaccard similarities.

2. Decision Tree

- **Accuracy:** 37%

While simple and interpretable, the Decision Tree algorithm most likely suffered from overfitting and reduced generalization. Despite hyperparameter tuning (max depth, minimum samples per split), the model underperformed compared to K-NN.

3. Random Forest

- **Accuracy:** 39%

As an ensemble method, Random Forest improved its performance compared to Decision Trees by reducing variance and overfitting. Bootstrap sampling and random feature selection certainly added robustness. However, its accuracy remained still lower than K-NN.

4. Person Similarity

- **Accuracy:** 55%

The simplest approach that is based solely on user rating similarity outperformed all other machine learning models. By utilizing exact match counts and considering overlapping movie ratings, this straightforward approach, proved to be the most accurate.

5. Collaborative Filtering

- **Accuracy:** 39%

This technique struggled to achieve consistent performance across users. Variations in accuracy (ranging from 20% to over 90% for different users) highlighted its sensitivity to initialization and hyperparameter tuning. Despite a mean accuracy similar to Random Forest, its per-user inconsistency is definitely a drawback.

To sum up, the evaluation showed accuracies ranging from 37% to 55%. Person Similarity achieved the highest accuracy. K-NN performed moderately at 41%. Decision Tree and Random Forest achieved 37% and 39% accuracy. Collaborative Filtering demonstrated good results but lacked consistency across users.