

# Games classifier

Team name

Members: Julia Cygan, Borys Adamiak, Patryk Flama  
Supervisor: Marek Adamczyk

UWr

January 28, 2025

## Use case example:

Imagine that you run a online game store where users can add their games to your library. Instead of manually checking if user tagged correctly the game, you can use our model to do that job for you.

## Goal:

We want to be able to automatically assign tags (or genres) to games, based on their (text) description.

Additionally, in aspect of ML project, we want to make a small comparison of different models and methods for solving such multilabel classification problem.

# Info about the dataset

Steam has its own official API, from which we downloaded games, their descriptions, tags and genres. That resulted in a bit over *200'000* games.

To clean the data we:

- Converted descriptions to alphanumeric lowercase
- Removed html tags
- Removed empty descriptions or tags
- (optional) Removed tags/genres that occurred at most  $n$  times

After that we ended up with a dataset of size around *50'000* games and *400* unique tags or *100* unique genres.

# Data preprocessing

To represent the output we decided to use multi label binary vector.

# Data preprocessing

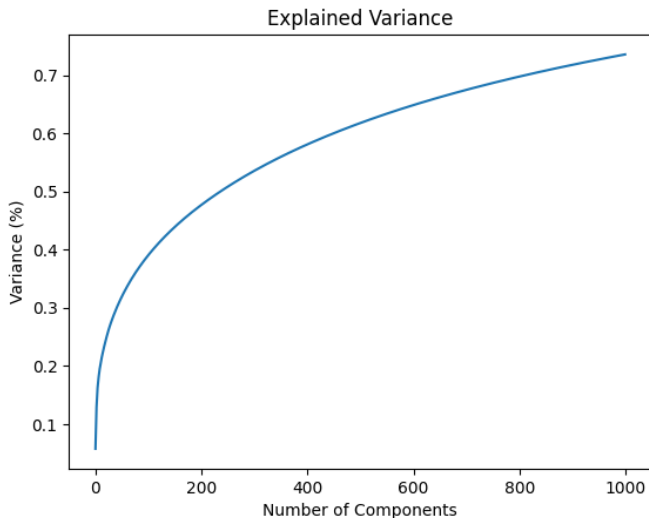
To represent the output we decided to use multi label binary vector.

For input preprocessing we tried:

- Bag of Words
- TF-IDF
- Hashing vectorizer

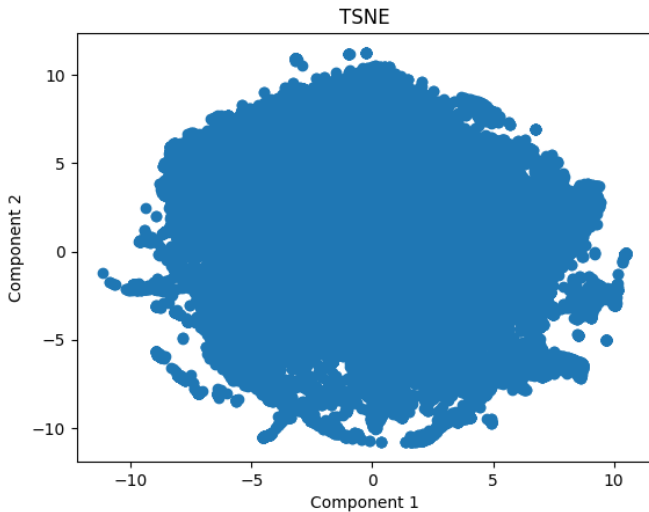
We decided to check if there are some patterns in the data that we can use to improve our model.

Figure: PCA analysis on Bag of Words



# Data preprocessing

Figure: t-SNE 300 iterations + PCA to 50 on Bag of Words



# Models

- KNN

Seemed to be a good baseline for our project, since its so simple



# Models

- KNN

Seemed to be a good baseline for our project, since its so simple

- Naive Bayes

Also quite simple in practice, but a bit more interesting in the context of text classification

# Models

- KNN  
Seemed to be a good baseline for our project, since its so simple
- Naive Bayes  
Also quite simple in practice, but a bit more interesting in the context of text classification
- Logistic Regression  
Classical approach

# Models

- KNN  
Seemed to be a good baseline for our project, since its so simple
- Naive Bayes  
Also quite simple in practice, but a bit more interesting in the context of text classification
- Logistic Regression  
Classical approach
- Decision Trees  
Do not have the best reputation, so we wanted to check how they perform

# Models

- KNN  
Seemed to be a good baseline for our project, since its so simple
- Naive Bayes  
Also quite simple in practice, but a bit more interesting in the context of text classification
- Logistic Regression  
Classical approach
- Decision Trees  
Do not have the best reputation, so we wanted to check how they perform
- Random Forest  
Natural extension of Decision Trees

# Models

- KNN  
Seemed to be a good baseline for our project, since its so simple
- Naive Bayes  
Also quite simple in practice, but a bit more interesting in the context of text classification
- Logistic Regression  
Classical approach
- Decision Trees  
Do not have the best reputation, so we wanted to check how they perform
- Random Forest  
Natural extension of Decision Trees
- Simple perceptron-based neural network  
It is a neural network, thus we could not skip it

# Models

- KNN  
Seemed to be a good baseline for our project, since its so simple
- Naive Bayes  
Also quite simple in practice, but a bit more interesting in the context of text classification
- Logistic Regression  
Classical approach
- Decision Trees  
Do not have the best reputation, so we wanted to check how they perform
- Random Forest  
Natural extension of Decision Trees
- Simple perceptron-based neural network  
It is a neural network, thus we could not skip it
- Support Vector Machine  
Interesting concept

# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

- Exact match  
typical approach for one label - the prediction must be exact



# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

- Exact match  
typical approach for one label - the prediction must be exact
- Precision  $TP/(TP+FP)$   
we focus on minimizing FP

# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

- Exact match  
typical approach for one label - the prediction must be exact
- Precision  $TP/(TP+FP)$   
we focus on minimizing FP
- Recall  $TP/(TP+FN)$   
we focus on minimizing FN

# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

- Exact match  
typical approach for one label - the prediction must be exact
- Precision  $TP/(TP+FP)$   
we focus on minimizing FP
- Recall  $TP/(TP+FN)$   
we focus on minimizing FN
- F1-score  $(2 * precision * recall) / (precision + recall)$   
combines precision and recall

# Evaluation

Since choosing proper evaluation function is a major part of our project (because we have specific multi-label classification) we decided to use following metrics:

- Exact match  
typical approach for one label - the prediction must be exact
- Precision  $TP/(TP+FP)$   
we focus on minimizing FP
- Recall  $TP/(TP+FN)$   
we focus on minimizing FN
- F1-score  $(2 * precision * recall) / (precision + recall)$   
combines precision and recall
- Intersection over union score  
this evaluation does not treat each label separately, but check ratio of intersection and union

Figure: Different number of neighbors

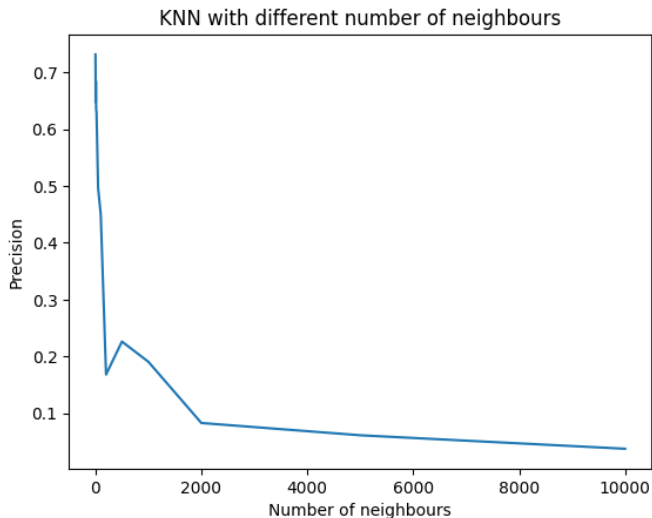
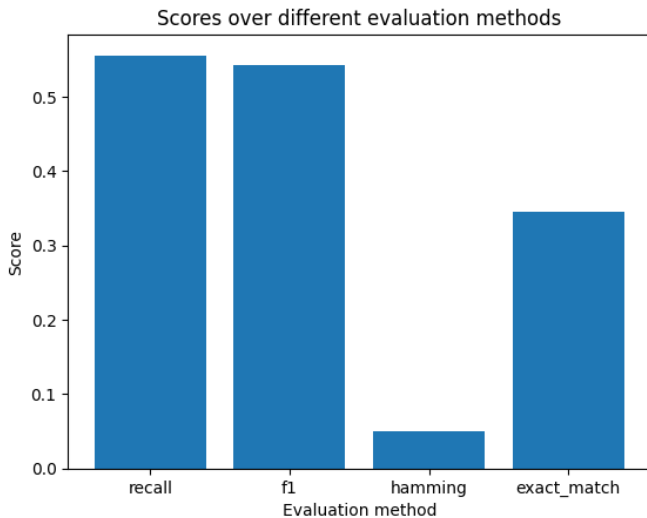
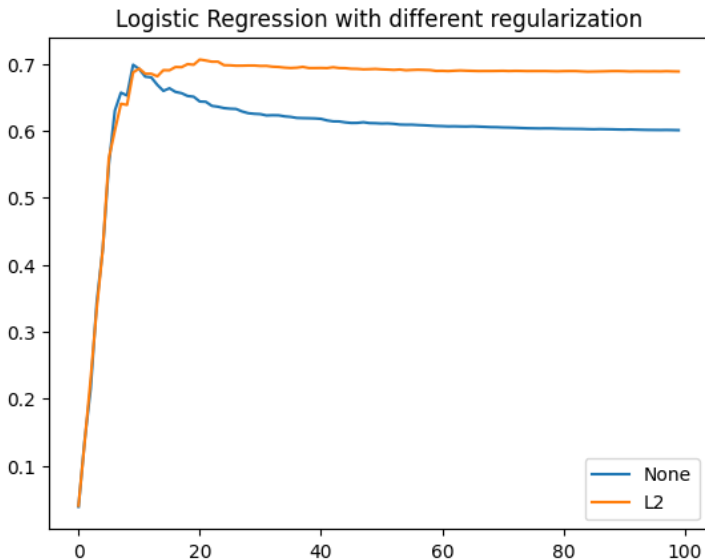


Figure: Comparison of evaluations



# Results - Logistic Regression



# Results - Decision Tree

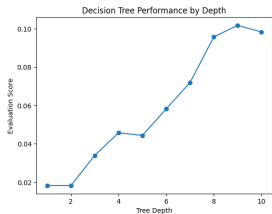


Figure: Exact match

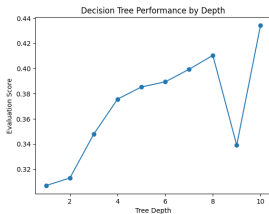


Figure: F1-score

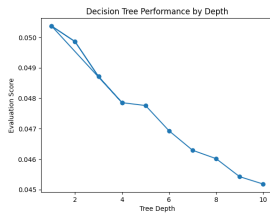


Figure: Hamming loss

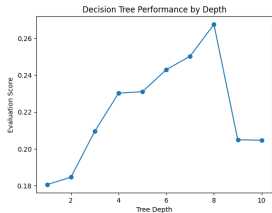


Figure: Intersection

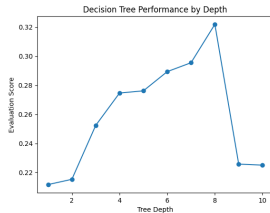
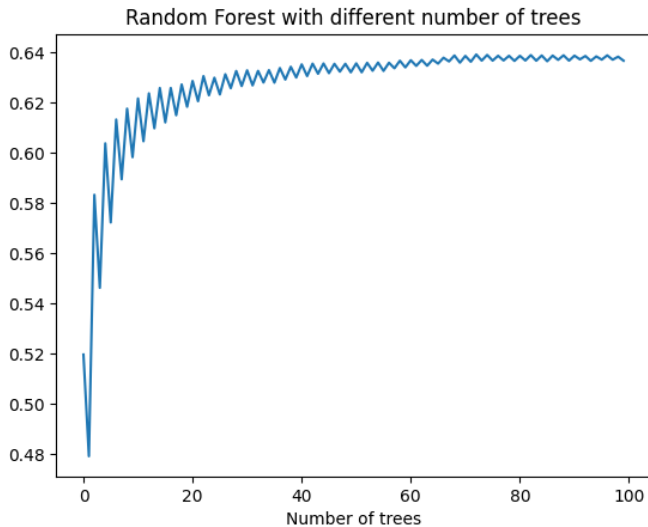


Figure: Recall



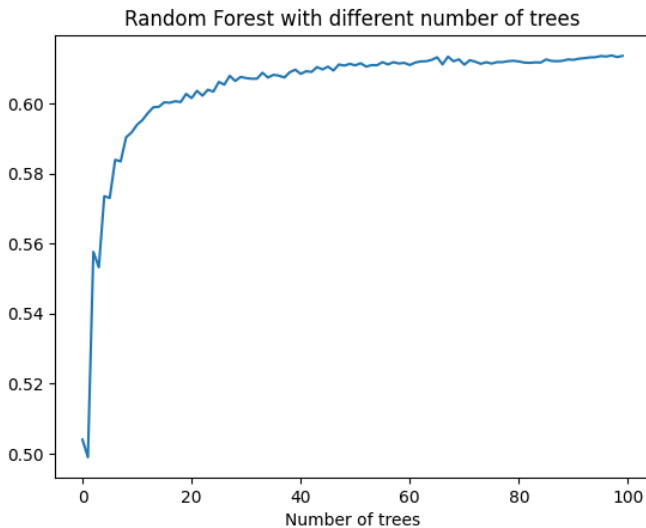
# Results - Random Forest

Figure: Different number of trees, no depth limit



# Results - Random Forest

Figure: Different number of trees, depth limited to 100

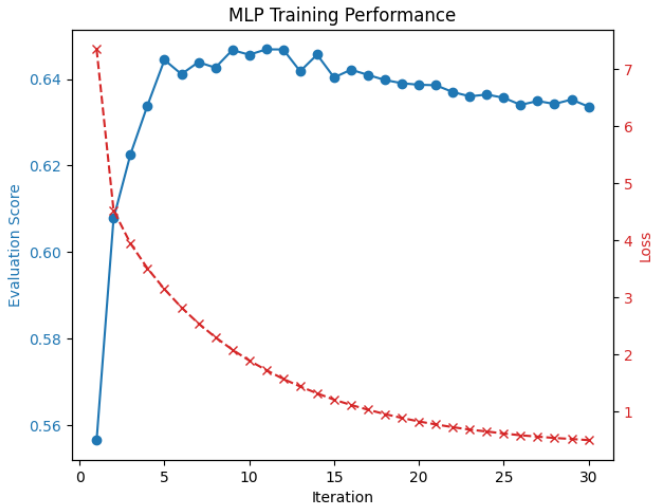


# Results - Support Vector Machine



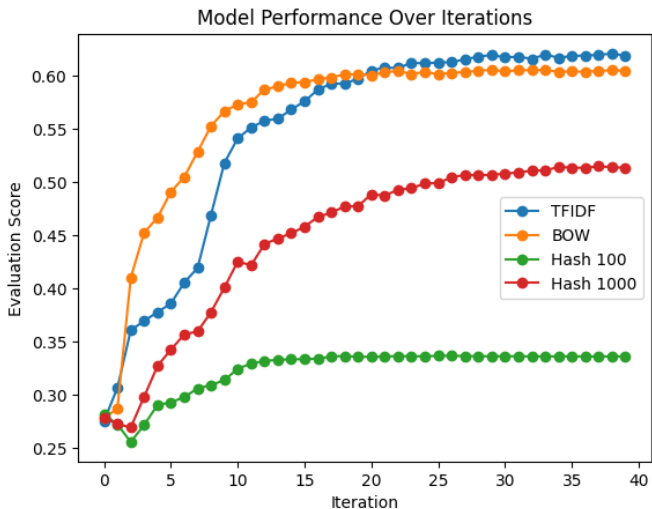
# Results - Neural Network

Figure: Multilayer Perceptron



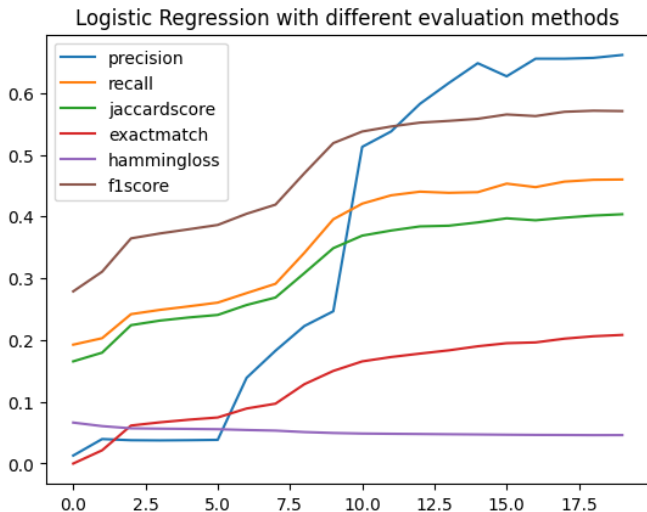
# Input preprocessing

Figure: Logistic Regression, F1-score

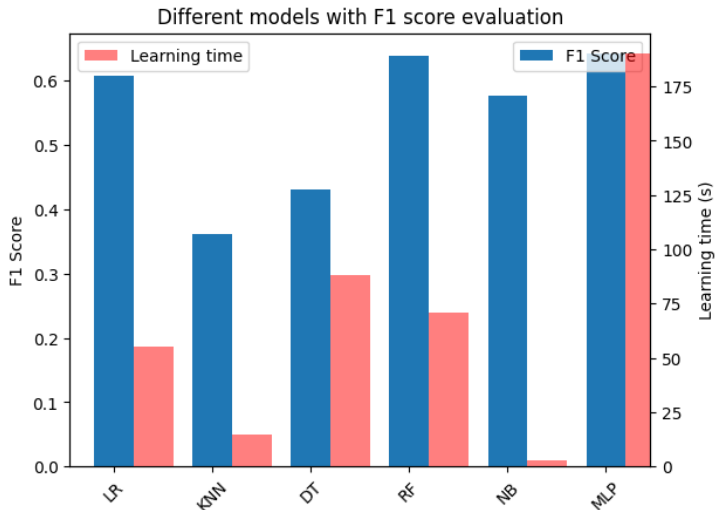


# Evaluation metrics

Figure: Logistic Regression, F1-score



# Models Comparison



# Conclusions

- From the input preprocessors the most effective was **TF-IDF** (that makes sense, as it carries the most information)



# Conclusions

- From the input preprocessors the most effective was **TF-IDF** (that makes sense, as it carries the most information)
- The best model was **neural network**, with **random forest** and **logistic regression** not far behind (additionally neural network was the most time consuming to train)

# Conclusions

- From the input preprocessors the most effective was **TF-IDF** (that makes sense, as it carries the most information)
- The best model was **neural network**, with **random forest** and **logistic regression** not far behind (additionally neural network was the most time consuming to train)
- **KNN** indeed was far from being the best model but **naive bayes**, which was the fastest, was quite good

# Conclusions

- From the input preprocessors the most effective was **TF-IDF** (that makes sense, as it carries the most information)
- The best model was **neural network**, with **random forest** and **logistic regression** not far behind (additionally neural network was the most time consuming to train)
- **KNN** indeed was far from being the best model but **naive bayes**, which was the fastest, was quite good
- **Decision trees** were empirically proven (again) that they are not the best choice

# Conclusions

- From the input preprocessors the most effective was **TF-IDF** (that makes sense, as it carries the most information)
- The best model was **neural network**, with **random forest** and **logistic regression** not far behind (additionally neural network was the most time consuming to train)
- **KNN** indeed was far from being the best model but **naive bayes**, which was the fastest, was quite good
- **Decision trees** were empirically proven (again) that they are not the best choice
- **SVM** did not perform too bad, nor too good we suspect that, based on how it works, it could **eventually** perform way better (but that would require a lot of time)