Performance Comparison of Synthetic and Non-Synthetic Data with a Machine Learning and Deep Learning Based Approach on Twitter Hate Speech Detection: Performance Analysis with Vectorization Methods

---

Introduction

Social media platforms provide an environment where individuals can freely express their views, but they have also become a medium where hate speech can spread rapidly. Especially on platforms with large user bases such as Twitter, the detection of hate speech has become an important issue in terms of digital security and public health. In this study, machine learning and deep learning methods were used with synthetic and non-synthetic datasets to detect hate speech in texts shared on the Twitter platform. The main purpose of the study is to compare the results obtained with the two types of datasets and to reveal the most effective modeling approaches in this field.

Methodology

The methods used in this study include data processing, feature extraction, and model training stages.

Datasets

The datasets used in the study were divided into two main groups: synthetic and non-synthetic. The same preprocessing procedures were applied to both datasets:

- **• Data processing steps:**
    - Tokenization and Cleansing: Data is split into words (tokenized) and cleaned of non-significant characters, stop words, and special characters.
    - Zemberek Spell Correction: Zemberek library was used to correct grammatical errors in Turkish texts.
    - Zemberek Normalize: Normalization operations are performed with the Zemberek library for consistent modeling of texts..

1. **1. Non-Synthetic Data:** A dataset consisting of existing hate speech texts.
2. **2. Synthetic Data:** Synthetic data generation, using Llama 3.1 7D language model, was structured to add new texts to the existing dataset and enrich the dataset. This process was structured to help balance the unbalanced classes.

To overcome the problems caused by imbalanced data classes, SMOTE (Synthetic Minority Oversampling Technique) method was applied. The same data processing and modeling procedures were applied to both data sets.

Feature Extraction

Texts were converted into numerical data using different vectorization methods:

- **TurkishWord2Vec:** A pre-trained word embedding model specific to Turkish language.
- **Fine-Tune Word2Vec:** Study-specific retrained Word2Vec model.
- **N-gram Temelli Models:**
    - **Word-gram:** Word-based n-gram representations.
    - **Char-gram:** Character-based n-gram features.
    - **TF-IDF and CountVectorizer**: N-gram methods are used with two vectorization methods. Along with these methods, **TF-IDF and CountVectorizer** based feature extraction are also applied. These two methods are used to provide frequency based representation of texts and to increase model performance.

• **Combined Methods:** Combination of word and character based features (e.g. Word Unigram + Word Bigram + Char Trigrams).

Model Training

The following models were trained to detect hate speech:

- **CatBoost ve XGBoost:** Gradient boosting algorithms.
- **MLPC-SGD:** The multilayer perceptron (MLP) model is optimized by stochastic gradient descent method.
- **ExtraTreesClassifier:** Tree-based ensemble learning model.
- **ANN:** A customized artificial neural network architecture.

Evaluation Metrics

- The following metrics were used to measure model performances:
- **Accuracy**
- **Precision**

- **Recall**
- **F1 Score**

Results and Analysis

Non-Synthetic Dataset Results:

Model	Vectorization Method	Accuracy	Precision	Recall	F1 Score
CatBoostC	TurkishWord2Vec	0.629794	0.716249	0.629794	0.66494
XGBoost	TurkishWord2Vec	0.766962	0.749744	0.766962	0.757297
MLPC-sgd	TurkishWord2Vec	0.725664	0.767823	0.725664	0.74114
ExtraTreesC	TurkishWord2Vec	0.770403	0.730594	0.770403	0.735169
ANN	TurkishWord2Vec	0.161259	0.771834	0.161259	0.249338
CatBoostC	FineTuneWord2Vec	0.538348	0.672612	0.538348	0.590817
XGBoost	FineTuneWord2Vec	0.721239	0.700948	0.721239	0.7093
MLPC-sgd	FineTuneWord2Vec	0.301868	0.650208	0.301868	0.379265
ExtraTreesC	FineTuneWord2Vec	0.752212	0.698773	0.752212	0.702753
ANN	FineTuneWord2Vec	0.23648	0.700203	0.23648	0.098889
CatBoostC	WordUnigram	0.704523	0.731224	0.704523	0.716179
XGBoost	WordUnigram	0.816126	0.799745	0.816126	0.797131
MLPC-sgd	WordUnigram	0.807276	0.808634	0.807276	0.807788
ExtraTreesC	WordUnigram	0.8353	0.828411	0.8353	0.811472
ANN	WordUnigram	0.816618	0.81445	0.816618	0.815446
CatBoostC	WordBigram	0.729597	0.710671	0.729597	0.707012
XGBoost	WordBigram	0.783677	0.758221	0.783677	0.755539
MLPC-sgd	WordBigram	0.648968	0.733189	0.648968	0.681984
ExtraTreesC	WordBigram	0.764503	0.75917	0.764503	0.760436
ANN	WordBigram	0.519174	0.760847	0.519174	0.592858
CatBoostC	CharBigram	0.761554	0.728806	0.761554	0.724274
XGBoost	CharBigram	0.82645	0.812186	0.82645	0.8001
MLPC-sgd	CharBigram	0.731072	0.790732	0.731072	0.750124
ExtraTreesC	CharBigram	0.790069	0.80668	0.790069	0.727241
ANN	CharBigram	0.137168	0.770219	0.137168	0.212203
CatBoostC	CharTrigram	0.741396	0.724967	0.741396	0.72919
XGBoost	CharTrigram	0.838741	0.832306	0.838741	0.816767
MLPC-sgd	CharTrigram	0.819076	0.821387	0.819076	0.819896
ExtraTreesC	CharTrigram	0.806293	0.832386	0.806293	0.756911
ANN	CharTrigram	0.783677	0.818615	0.783677	0.796108
CatBoostC	ChBigram+ChTrigram	0.741396	0.719161	0.741396	0.726477
XGBoost	ChBigram+ChTrigram	0.83825	0.824372	0.83825	0.815237

MLPC-sgd	ChBigram+ChTrigram	0.8294	0.827176	0.8294	0.828227
ExtraTreesC	ChBigram+ChTrigram	0.803835	0.816027	0.803835	0.752435
ANN	ChBigram+ChTrigram	0.762537	0.813005	0.762537	0.775348
CatBoostC	ChUnigram+ChTrigram	0.739921	0.718422	0.739921	0.724473
XGBoost	ChUnigram+ChTrigram	0.848083	0.854061	0.848083	0.825588
MLPC-sgd	ChUnigram+ChTrigram	0.8353	0.833341	0.8353	0.834234
ExtraTreesC	ChUnigram+ChTrigram	0.806293	0.828108	0.806293	0.757105
ANN	ChUnigram+ChTrigram	0.819567	0.836212	0.819567	0.824297
CatBoostC	WUniG+WBiG+ChTriG	0.764012	0.737197	0.764012	0.738754
XGBoost	WUniG+WBiG+ChTriG	0.844149	0.832686	0.844149	0.821763
MLPC-sgd	WUniG+WBiG+ChTriG	0.830383	0.838395	0.830383	0.833675
ExtraTreesC	WUniG+WBiG+ChTriG	0.813176	0.846978	0.813176	0.766256
ANN	WUniG+WBiG+ChTriG	0.60472	0.812273	0.60472	0.655022
CatBoostC	WUniG+WBiG+ChBiG+ChTriG	0.767453	0.738909	0.767453	0.738186
XGBoost	WUniG+WBiG+ChBiG+ChTriG	0.840708	0.826401	0.840708	0.818919
MLPC-sgd	WUniG+WBiG+ChBiG+ChTriG	0.830383	0.832346	0.830383	0.83127
ExtraTreesC	WUniG+WBiG+ChBiG+ChTriG	0.809243	0.821543	0.809243	0.762284
ANN	WUniG+WBiG+ChBiG+ChTriG	0.685349	0.811423	0.685349	0.709396

Synthetic Dataset Results:

Model	Vectorization Method	Accuracy	Precision	Recall	F1 Score
CatBoostC	TurkishWord2Vec	0.523996	0.556403	0.523996	0.531438
XGBoost	TurkishWord2Vec	0.782372	0.782442	0.782372	0.782356
MLPC-sgd	TurkishWord2Vec	0.591307	0.607418	0.591307	0.59582
ExtraTreesC	TurkishWord2Vec	0.805312	0.817553	0.805312	0.800627
ANN	TurkishWord2Vec	0.262904	0.581094	0.262904	0.215534
CatBoostC	FineTuneWord2Vec	0.509206	0.521106	0.509206	0.512013
XGBoost	FineTuneWord2Vec	0.740417	0.740764	0.740417	0.740551
MLPC-sgd	FineTuneWord2Vec	0.367039	0.44585	0.367039	0.370599
ExtraTreesC	FineTuneWord2Vec	0.792937	0.801651	0.792937	0.789837
ANN	FineTuneWord2Vec	0.17205	0.658466	0.17205	0.061152
CatBoostC	WordUnigram	0.519469	0.557968	0.519469	0.524308
XGBoost	WordUnigram	0.672502	0.675654	0.672502	0.673469
MLPC-sgd	WordUnigram	0.783278	0.787803	0.783278	0.784398
ExtraTreesC	WordUnigram	0.84425	0.849124	0.84425	0.841271
ANN	WordUnigram	0.827347	0.829456	0.827347	0.828018
CatBoostC	WordBigram	0.505886	0.522915	0.505886	0.44252
XGBoost	WordBigram	0.589798	0.630491	0.589798	0.5477
MLPC-sgd	WordBigram	0.628132	0.634909	0.628132	0.623613
ExtraTreesC	WordBigram	0.743435	0.748174	0.743435	0.739575

ANN	WordBigram	0.60821	0.74537	0.60821	0.630998
CatBoostC	CharBigram	0.560519	0.578275	0.560519	0.563778
XGBoost	CharBigram	0.759433	0.757384	0.759433	0.757541
MLPC-sgd	CharBigram	0.651675	0.658885	0.651675	0.654097
ExtraTreesC	CharBigram	0.823725	0.834671	0.823725	0.819606
ANN	CharBigram	0.485964	0.657462	0.485964	0.510712
CatBoostC	CharTrigram	0.564443	0.58376	0.564443	0.568469
XGBoost	CharTrigram	0.751283	0.748642	0.751283	0.748609
MLPC-sgd	CharTrigram	0.807727	0.811057	0.807727	0.808448
ExtraTreesC	CharTrigram	0.845759	0.854027	0.845759	0.84225
ANN	CharTrigram	0.793239	0.804247	0.793239	0.795865
CatBoostC	ChBigram+ChTrigram	0.565047	0.584583	0.565047	0.567764
XGBoost	ChBigram+ChTrigram	0.750377	0.747559	0.750377	0.747017
MLPC-sgd	ChBigram+ChTrigram	0.829762	0.831679	0.829762	0.830275
ExtraTreesC	ChBigram+ChTrigram	0.844854	0.854442	0.844854	0.84103
ANN	ChBigram+ChTrigram	0.793842	0.805826	0.793842	0.794556
CatBoostC	ChUnigram+ChTrigram	0.561425	0.579627	0.561425	0.564498
XGBoost	ChUnigram+ChTrigram	0.746755	0.743959	0.746755	0.744273
MLPC-sgd	ChUnigram+ChTrigram	0.833685	0.836285	0.833685	0.834389
ExtraTreesC	ChUnigram+ChTrigram	0.846665	0.85492	0.846665	0.843279
ANN	ChUnigram+ChTrigram	0.82946	0.830721	0.82946	0.829567
CatBoostC	WUniG+WBiG+ChTriG	0.575309	0.593766	0.575309	0.577804
XGBoost	WUniG+WBiG+ChTriG	0.772412	0.770017	0.772412	0.770196
MLPC-sgd	WUniG+WBiG+ChTriG	0.831874	0.834152	0.831874	0.832528
ExtraTreesC	WUniG+WBiG+ChTriG	0.846061	0.855475	0.846061	0.842378
ANN	WUniG+WBiG+ChTriG	0.49834	0.718788	0.49834	0.526348
CatBoostC	WUniG+WBiG+ChBiG+ChTriG	0.58859	0.601096	0.58859	0.589644
XGBoost	WUniG+WBiG+ChBiG+ChTriG	0.771204	0.769221	0.771204	0.769373
MLPC-sgd	WUniG+WBiG+ChBiG+ChTriG	0.830969	0.830624	0.830969	0.829509
ExtraTreesC	WUniG+WBiG+ChBiG+ChTriG	0.842741	0.853998	0.842741	0.838975
ANN	WUniG+WBiG+ChBiG+ChTriG	0.737096	0.75724	0.737096	0.74007

Overall Performance Comparison

Table 1 summarizes the average accuracy and F1 scores of the vectorization methods used with synthetic and non-synthetic datasets:

Vectorization Method	**F1 Skoru (Synthetic)**	**F1 Skoru (Non- Synthetic)**
Char Unigram + Char Trigram	**% 82.95**	**%82.42**
WUniG + WBiG + ChBiG + ChTriG	**% 74.00**	**%70.93**
TurkishWord2Vec	**% 80.00**	**% 75.72**
Fine-Tune Word2Vec	**% 78.98**	**% 70.27**

The results show that character-based methods (e.g. ChTrigram) generally provide the highest performance, while methods such as Fine-Tune Word2Vec exhibit poor performance.

Model Performances

Table 2, presents detailed results of the models running with character-based methods:

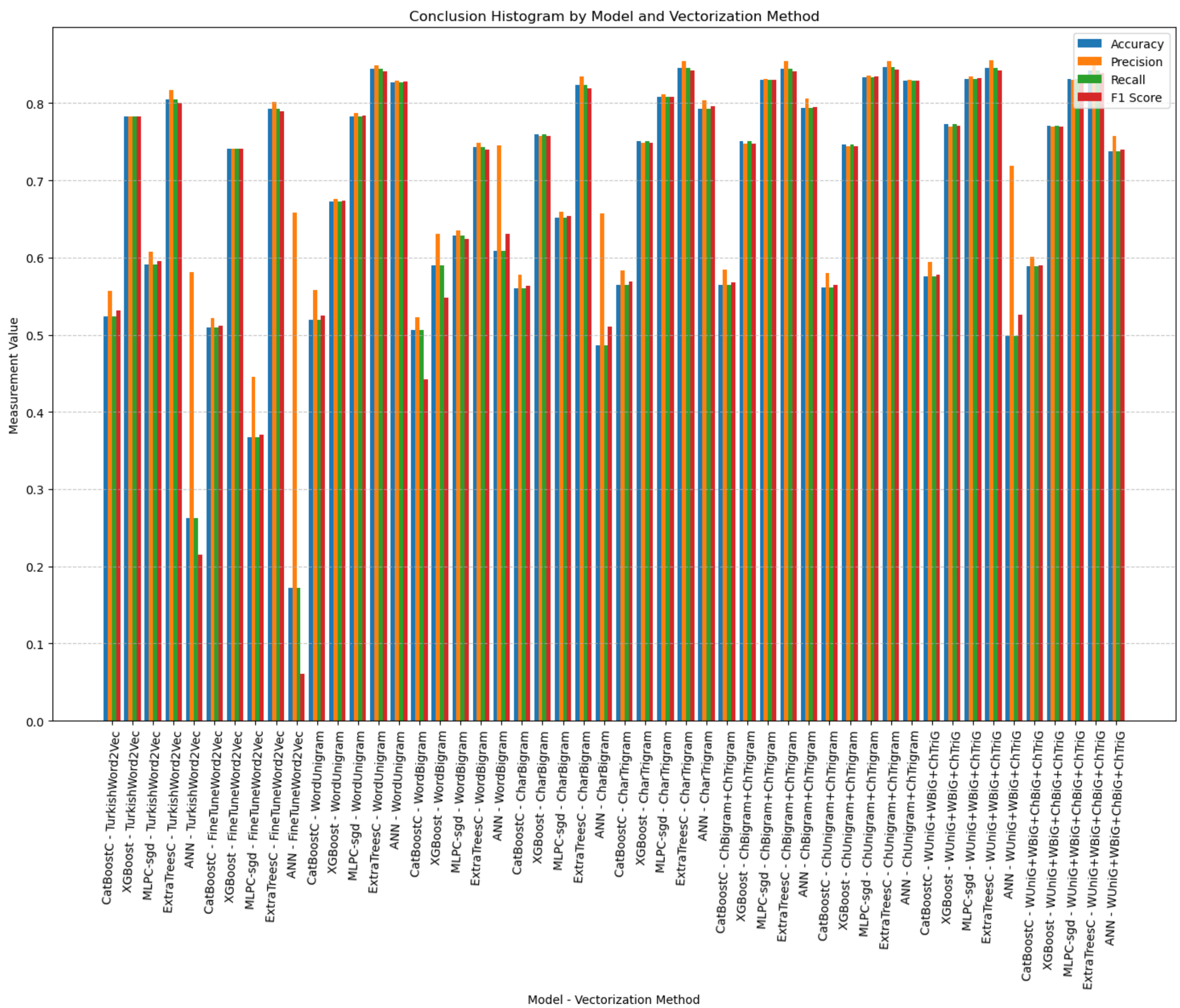
Model	Vectorization Method	Acc	F1 score
XGBoost	CharTrigram	%83.87	%81.67
ExtraTrees	ChUnigram+ChTrigram	%84.66	%84.32
MLPC-SGD	ChBigram+ChTrigram	%82.97	%82.82
CatBoostC	CharTrigram	%74.14	%72.91
ANN	CharTrigram	%78.36	%79.59

The Impact of Synthetic Data

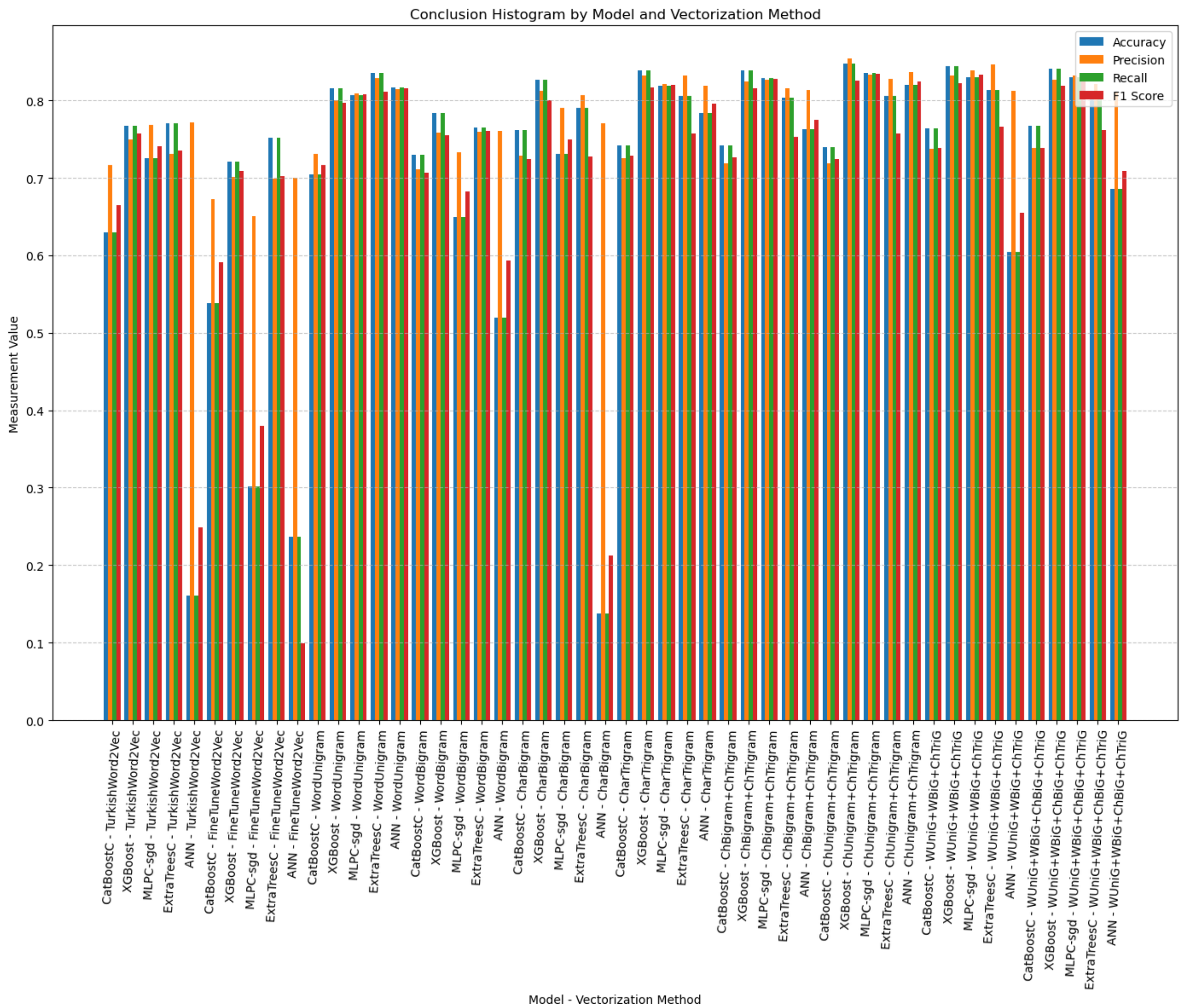
Using synthetic data was able to improve performance in models running on low amounts of data. However, in general, models running on non-synthetic data achieved higher accuracy and F1 scores. For example:

- It has been observed that the success rate between non-synthetic data and synthetic data is between 3% and 5% in favor of the synthetic one.
- The most successful vectorization group was Char Unigram + Char Trigram. The average difference between them was 4% - 5%.

Non-Synthetic:.



Synthetic:



Argument

The study results show that using synthetic data can improve model performance in some cases. However, models optimized with the current dataset generally provide higher accuracy and F1 score. In particular, it has been observed that character-based n-gram methods allow for a more effective representation of the word structure in Turkish. The use of methods such as TF-IDF and CountVectorizer has shown the effectiveness of frequency-based approaches and it has been observed that these techniques contribute in certain cases. In the future, the effectiveness of this method can be increased by using more advanced language models (e.g. GPT series) in synthetic data generation. In addition, the generalizability of model performances across platforms can be investigated by working on datasets from different social media platforms (e.g. Facebook, Instagram).