Natural Language Processing with Artificial Learning Project Part II

Project Name: Detection of Hate Speech in Turkish Tweets

Purpose of the Project

The purpose of this project is to develop a machine learning-based system for detecting hate speech in Turkish tweets. It aims to achieve the following objectives:

- **Hate Speech Identification**: Automatically classify tweets into predefined categories such as hate speech, offensive content, or neutral language.
- **Balancing Class Imbalance**: Address the issue of class imbalance in the dataset through advanced techniques like synthetic tweet generation using **GPT-2** and data resampling using **SMOTE**
- **Real-Time Predictions**: Enable real-time predictions for newly provided tweets using the trained model, making the system applicable in real-world scenarios such as monitoring social media platforms.
- **Model Evaluation and Comparison**: Test multiple classification algorithms (e.g., Random Forest, XGBoost, LightGBM, ANN) to identify the most accurate model for hate speech detection.
- **Feature Representation**: Use advanced text processing techniques, including **Word2Vec embeddings**, to capture the contextual and semantic meanings of tweets.
- **Support for Turkish Language**: Incorporate specialized tools like **Zemberek** for Turkish language processing, ensuring accurate text analysis for a morphologically rich language.
- **Automation and Scalability**: Build a scalable and automated pipeline that can handle imbalanced datasets, generate synthetic data, and evaluate multiple classifiers for efficient and effective hate speech detection.

1. Installation and Import of Required Libraries

At the beginning of the code, the necessary libraries for NLP and machine learning are loaded:

- **JPype1 and Zemberek-Python**: Used for spelling correction and language analysis in Turkish text processing.
- **Transformers:** Text is produced using Hugging Face's GPT-2 model.
- **scikit-learn**: Essential tool for model training.
- **Word2Vec :** Creating word vectors.
- **Keras:** For developing Artificial Neural Network (ANN) model.

2. Spelling Correction with Zemberek

Using the Zemberek library, spelling errors in Turkish tweets are corrected.

- **Function:** correct_spelling(sentence) works on each tweet and returns the correct Turkish form.
-

3. Dataset Processing

Loading the Dataset

- Tweet and Tag columns are pulled from all sheets (tables) in the Excel file.
- These data are then combined.

Original class distribution:
Etiket
HİÇBİRİ 7722
NEFRET 2336
SALDIRGAN 166
NAME: COUNT, DTYPE: İNT6

Class Distribution

- The distribution of labels (hate speech classes) in the data set is checked.

Creating a Balanced Data Set

- **Problem:** Class imbalance in the data set can make it difficult for models to learn.
- **Solution:** A balanced data set is created.
  - For low-class tags, synthetic tweets are generated using GPT-2.
  - For tags with more classes, they are equalized by random sampling.

Balanced class distribution:
ETİKET
HİÇBİRİ 1000
NEFRET 1000
SALDIRGAN 1000
NAME: COUNT, DTYPE: İNT64

4. Representing Text with Word2Vec

Converting Tweets to Word Vectors

- The Word2Vec model is used to capture the contextual meaning of words.
- **Education:**
  - vector_size=200: Each word is represented by a 200-dimensional vector.
  - window=10: Increasing the context window between words.
  - sg=1: Skip-gram algorithm is preferred.
  - By averaging the word vectors of the tweets, each tweet is converted into a 200-dimensional vector.

5. Model Training and Evaluation

Classifiers Used

- Random Forest
- XGBoost
- LightGBM
- Logistic Regression
- Gradient Boosting
- Support Vector Machine (SVM)
- k-Nearest Neighbors (k-NN)

The following operations are performed for each model:

- SMOTE: Used to eliminate class imbalance.
- Training and prediction are done.

Performance Measurements:

- Accuracy, Precision, Recall, F1-Score.
- Confusion Matrix is visualized.

Artificial Neural Network (ANN) Model

- Input Layer: 256 neurons, relu activation function.

- Dropout Layers: 30% neurons are disabled to reduce overfitting.
- Output Layer: Neurons as many as the number of classes, softmax activation function.
- Education:
    - Epochs: 20
    - Batch size: 32
    - ANN is also trained on a balanced dataset with SMOTE+Tomek Links.

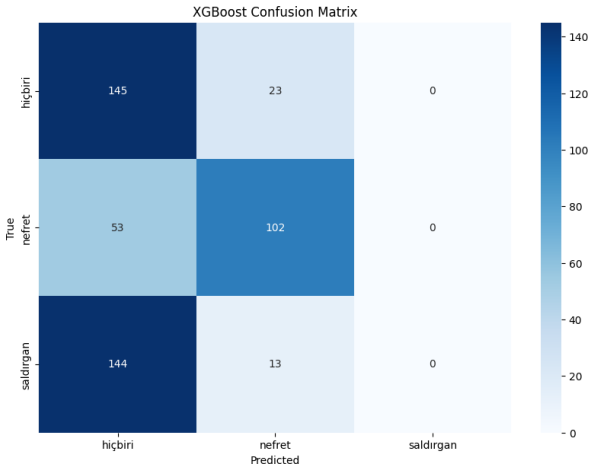
6. Performance Comparison

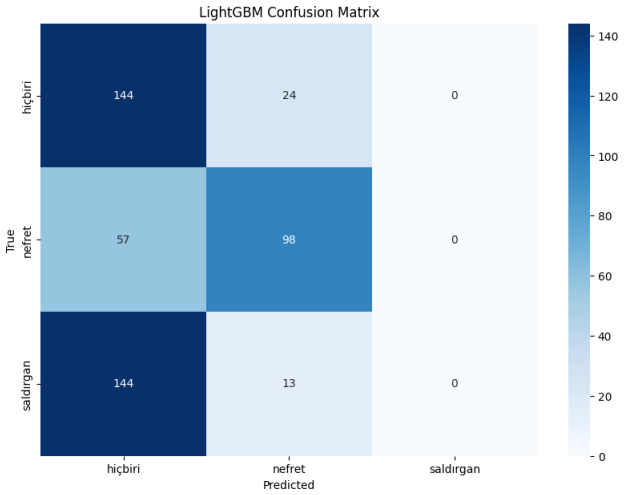
Best Model

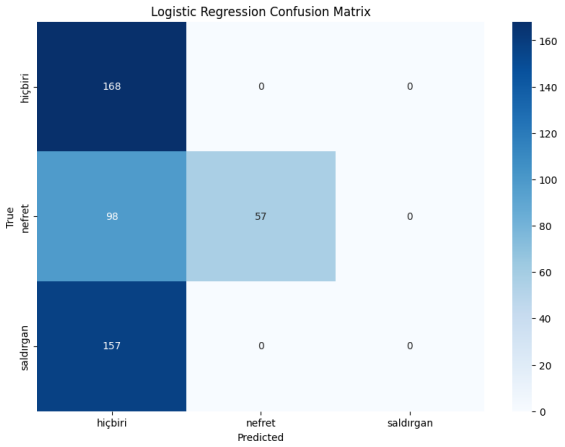
- All models are evaluated on the test set.
- The model that provides the highest accuracy is selected.

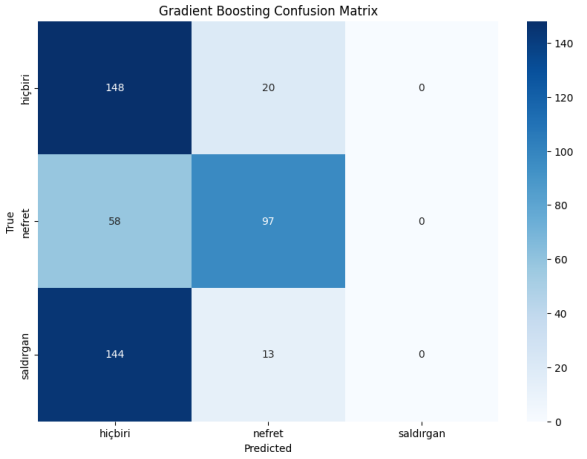
Performance Results

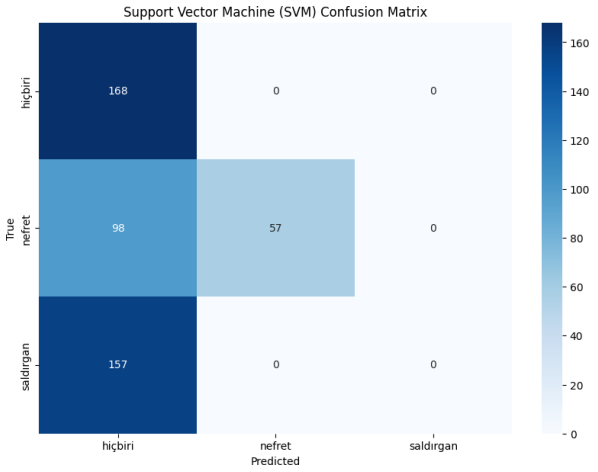
- Performance metrics are saved in a CSV file.

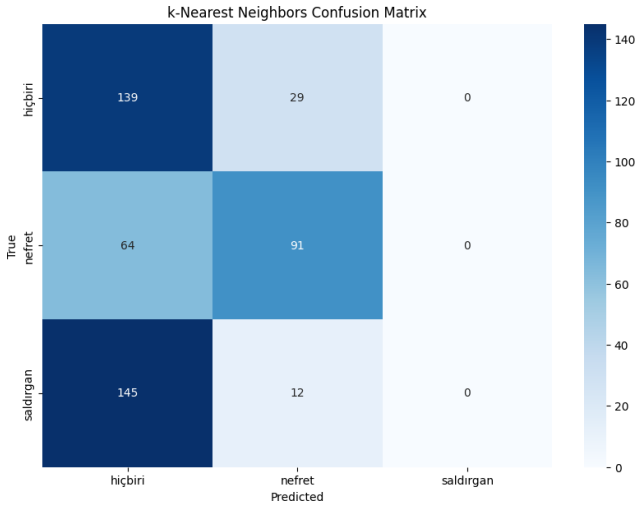


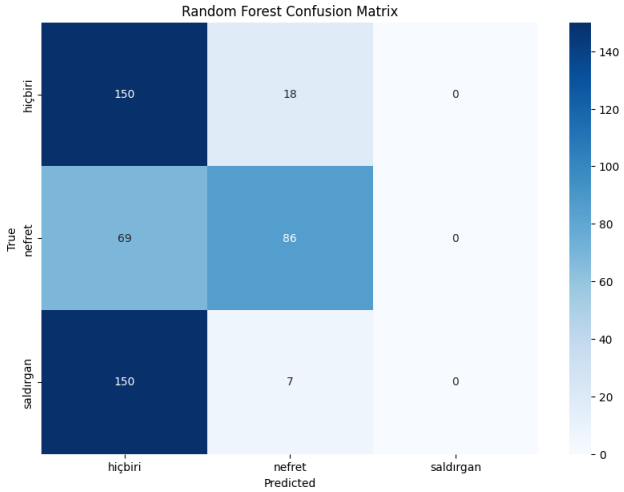












Performance Table

Classifier	Class	Precision	Recall	F1-Score	Accuracy
Random Forest	hiçbiri	0.47844827586206895	0.511520737327189	0.49443207126948774	0.595
Random Forest	nefret	0.42723004694835 68	0.46192893401015 23	0.44390243902439025	0.595
Random Forest	saldırgan	1.0	0.8333333333333334	0.9090909090909091	0.595
Random Forest	macro avg	0.63522610760347 52	0.60226100155689 16	0.61580847312826 24	0.595
Random Forest	weighted avg	0.62331265851815 88	0.595	0.60638574840698 8	0.595
XGBoost	hiçbiri	0.51739130434782 61	0.54838709677419 35	0.53243847874720 36	0.62
XGBoost	nefret	0.45794392523364 486	0.49746192893401 014	0.47688564476885 64	0.62
XGBoost	saldırgan	0.99358974358974 36	0.8333333333333334	0.90643274853801 17	0.62
XGBoost	macro avg	0.65630832439040 49	0.62639411968051 24	0.63858562401802 39	0.62
XGBoost	weighted avg	0.64549426437033 11	0.62	0.63013685522613 02	0.62
LightGBM	hiçbiri	0.5	0.539170506912442 4	0.518847006651884 7	0.6083333333333333
LightGBM	nefret	0.44019138755980863	0.467005076142132	0.45320197044334976	0.6083333333333333

LightGBM	saldırgan	0.9936305732484076	0.8387096774193549	0.9096209912536443	0.6083333333333333
LightGBM	macro avg	0.6446073202694055	0.6149617534913098	0.6272233227829596	0.6083333333333333
LightGBM	weighted avg	0.6333883166224769	0.6083333333333333	0.6184334883232945	0.6083333333333333
Logistic Regression	hiçbiri	0.4985163204747745	0.7741935483870968	0.6064981949458483	0.6233333333333333
Logistic Regression	nefret	0.4722222222222222	0.25888324873096447	0.3344262295081967	0.6233333333333333
Logistic Regression	saldırgan	1.0	0.8333333333333334	0.9090909090909091	0.6233333333333333
Logistic Regression	macro avg	0.6569128475656666	0.6221367101504649	0.6166717778483181	0.6233333333333333
Logistic Regression	weighted avg	0.6453430322013408	0.6233333333333333	0.6109716410121215	0.6233333333333333
Gradient Boosting	hiçbiri	0.4939271255060729	0.5622119815668203	0.5258620689655172	0.6
Gradient Boosting	nefret	0.4234693877551020 7	0.4213197969543147	0.4223918575063613	0.6
Gradient Boosting	saldırgan	0.9872611464968153	0.8333333333333334	0.9037900874635568	0.6
Gradient Boosting	macro avg	0.6348858865859968	0.6056217039514894	0.6173480046451 45	0.6
Gradient Boosting	weighted avg	0.6237270481183009	0.6	0.609047035270 82	0.6
Support Vector Machine (SVM)	hiçbiri	0.4873563218390804 5	0.9769585253456221	0.6503067484662577	0.6183333333333333
Support Vector Machine (SVM)	nefret	0.4	0.0203045685279187 8	0.0386473429951690 8	0.6183333333333333
Support Vector Machine (SVM)	saldırgan	1.0	0.8333333333333334	0.9090909090909091	0.6183333333333333
Support Vector Machine (SVM)	macro avg	0.6291187739463602	0.6101988090689581	0.5326816668507787	0.6183333333333333
Support Vector Machine (SVM)	weighted avg	0.6175938697318007	0.6183333333333333	0.5297016667968922	0.6183333333333333
k-Nearest Neighbors	hiçbiri	0.4920634920634920 4	0.5714285714285714	0.5287846481876333	0.6066666666666667
k-Nearest Neighbors	nefret	0.4427083333333333	0.4314720812182741 3	0.4370179948586118	0.6066666666666667
k-Nearest Neighbors	saldırgan	0.9935897435897436	0.8333333333333334	0.9064327485380117	0.6066666666666667
k-Nearest Neighbors	macro avg	0.6427871896621896	0.6120779953267262	0.6240784638614 19	0.6066666666666667
k-Nearest Neighbors	weighted avg	0.6313316862535612	0.6066666666666667	0.6157255081198886	0.6066666666666667
Artificial Neural Network (ANN) + SMOTE+Tomek Links	Overall				0.675