1. **Introduction**

   Text classification is a technique that plays an important role in the field of natural language processing (NLP) and allows texts to be classified into certain categories. Within the scope of this project, the TTC4900 dataset will be used to classify Turkish news texts.

   Within the scope of the project, pre-processing techniques such as tokenization and lemmatization will be applied during the processing of texts. In the feature extraction phase, Bag of Words (BoW), TF-IDF and Word2Vec methods will be used. Then, the classification process will be performed with different machine learning algorithms and deep learning models.

   In addition to classical machine learning techniques, advanced neural network architectures (CNN, LSTM, BiLSTM, GRU) and Transformer-based models (BERT) will be used. Model performances will be evaluated with metrics such as accuracy, precision, recall and F1-score, and the most successful model will be determined.

   This study aims to determine the most effective approach in the Turkish text classification process by comparing different methods.

2. **Methodology**

   2.1 Dataset

   When the dataset is examined, it is seen that there are equal numbers of news from each category. The dataset contains a total of 7 categories, including politics, world, economy, culture, health, sports and technology, and each category contains 700 data. This balanced distribution provides a suitable structure to fairly evaluate the performance of text classification models. There is no missing data in the dataset, and each news is labeled with text content belonging to a specific category.

   2.2 Preprocessing

   Data preprocessing operations were performed for Turkish news texts. First, the dataset was loaded and the categories were verified to be balanced. The texts were converted to lower case, special characters were cleaned and stopwords were removed. The texts were divided into words by applying tokenization and lemmatization was performed.

2.3 Feature Extraction

- **Bag of Words (BoW):** It is a feature extraction method that converts text data into numerical form and represents the text over word frequencies by taking into account the frequencies of words in a text.

```
df = pd.read_csv("processed_with_labels_7allV03.csv")

# Bag of Words i□in CountVectorizer kullan□m□
max_features = 5000  # En s□k kullan□lan 500 kelime
count_vectorizer = CountVectorizer(max_features=max_features)

# Bag of Words matrisi olu□turma
sparse_matrix = count_vectorizer.fit_transform(df['processed_text']).toarray()

# Sparse matrisi DataFrame'e d□n□□t□rme
bow_df = pd.DataFrame(sparse_matrix, columns=count_vectorizer.get_feature_names_out())

# Orijinal DataFrame'e ba□lama
final_df = pd.concat([bow_df])
```

- **Term Frequency-Inverse Document Frequency (TF-IDF):** It is a feature extraction method used to determine the importance of a word in a text and calculates the frequency (TF) of the word in the document and weights it according to its prevalence (IDF) among all documents.

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

# □□lenmi□ metinlerin bulundu□u CSV dosyas□n□ oku
df = pd.read_csv("processed_with_labels_7allV03.csv")

# TF-IDF i□in TfidfVectorizer kullan□m□
tfidf_vector = TfidfVectorizer(max_features=5000)
tfidf_matrix = tfidf_vector.fit_transform(df['processed_text']).toarray()

# TF-IDF terim adlar□n□ al ve DataFrame olu□tur
terms = tfidf_vector.get_feature_names_out()
Tfidf_df = pd.DataFrame(tfidf_matrix, columns=terms)



# Yeni DataFrame'i CSV olarak kaydet
Tfidf_df.to_csv("tfidf_7allV03.csv", index=False)
```

- **Word2Vec:** It is a deep learning-based feature extraction method that represents words in vector space and aims to capture semantic relationships between words; It enables language models to extract better meaning by transforming words into similar vectors when they occur in similar contexts.

```
# Word2Vec modelini eğit
tokenized_texts = LineSentence('processed_with_labels_7allV03.csv', max_sentence_length=5000)


word2vec_model = gensim.models.Word2Vec(sentences=tokenized_texts, vector_size=500, window=10, min_count=1, workers=4, sg=1, epochs=20)


# Modeli kaydet
word2vec_model.save("3_word2vec_model.model")

def get_document_vector(text, model):
    words = text.split()
    word_vecs = []
    for word in words:
        if word in model.wv:
            word_vecs.append(model.wv[word])
    if word_vecs:
        return np.mean(word_vecs, axis=0)
    else:
        return np.zeros(model.vector_size)


    # Generate document vectors
print("\nGenerating document vectors...")
doc_vectors = []
for text in df['processed_text']:
    doc_vectors.append(get_document_vector(text, word2vec_model))



# Convert to numpy array
word2vec_features = np.array(doc_vectors)


# Save Word2Vec features
word2vec_df = pd.DataFrame(word2vec_features)
word2vec_df.to_csv('word2vec_features.csv', index=False)
print(f"Word2Vec features shape: {word2vec_features.shape}")
```

2.4 **Model Implementation**

- **Traditional Machine Learning Models**
  We employed several traditional machine learning models for text classification, including Logistic Regression, XGBoost, Decision Tree, RandomForest, K-NN, and LightGBM. Each of these models was tuned and optimized to achieve the best classification performance. Logistic Regression was used with a maximum iteration of 1000 to ensure convergence, while XGBoost was configured with 'mlogloss' as the evaluation metric. Decision Tree and RandomForest classifiers were chosen for their interpretability and ensemble learning capabilities. The K-Nearest Neighbors (K-NN) algorithm was applied to leverage similarity-based classification, and LightGBM was utilized for its efficiency in handling large datasets with high speed and accuracy.

- **Deep Learning Models**
  Artificial Neural Networks (ANN) have been implemented with only TF-IDF, Word2Vec and Bag of Words (BoW) feature extraction methods. The ANN model has been tested with various layer configurations to better understand and classify text features.

  Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM) and Gated Recurrent Units (GRU) models were used in combination with randomly initialized embeddings and pre-trained Word2Vec embeddings.

  - **Random Embeddings**: During the training process, randomly initialized embeddings are generated with a specific distribution, allowing the model to learn task-specific word representations from scratch. This approach allows the model to discover word meanings specific to the training data.

```python
# Load your processed dataset with text and labels
data = pd.read_csv('processed_with_labels_7allV03.csv')
texts = data['processed_text'].values
labels = data['label'].values

# Tokenization for random embeddings
max_words = 10000  # Maximum number of words to keep
max_len = 100      # Maximum length of each text

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
X_random = pad_sequences(sequences, maxlen=max_len)

# Split the data
X_random_train, X_random_test, y_train, y_test = train_test_split(
    X_random, labels, test_size=0.2, random_state=42
)
```

- **Word2Vec Embeddings**: Pre-trained Word2Vec embeddings capture the semantic relationships between words, allowing the model to generalize better. These embeddings aim to obtain more meaningful classification results by preserving the context of the words.

```
# Load your Word2Vec features
word2vec_features = pd.read_csv('word2vec_features.csv')
X_w2v = word2vec_features.values

# Split Word2Vec data
X_w2v_train, X_w2v_test, y_train_w2v, y_test_w2v = train_test_split(
    X_w2v, labels, test_size=0.2, random_state=42
)

# Reshape for CNN input
X_w2v_train_reshaped = X_w2v_train.reshape(X_w2v_train.shape[0], X_w2v_train.shape[1], 1)
X_w2v_test_reshaped = X_w2v_test.reshape(X_w2v_test.shape[0], X_w2v_test.shape[1], 1)
```

Additionally, two different BERT-based models were used:

- **Multilingual BERT:** The 'bert-base-multilingual-uncased' model is used to understand multilingual texts. This model has a wide range of uses as it provides a common representation for texts in different languages.

```
train_dataset_multilingual = TextDataset(train_encodings_multilingual, train_labels)
val_dataset_multilingual = TextDataset(val_encodings_multilingual, val_labels)
test_dataset_multilingual = TextDataset(test_encodings_multilingual, test_labels)

# Multilingual BERT Modeli
multilingual_model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-uncased', num_labels=len(unique_labels))
```

- **Fine-tuned BERT for Turkish**: Focused on Turkish texts using the 'savasy/bert-turkish-text-classification' model. Trained on the grammatical features of the Turkish language, this model helps to obtain more accurate results.

```
train_dataset_turkish = TextDataset(train_encodings_turkish, train_labels)
val_dataset_turkish = TextDataset(val_encodings_turkish, val_labels)
test_dataset_turkish = TextDataset(test_encodings_turkish, test_labels)

# Türkçe BERT Modeli
turkish_model = BertForSequenceClassification.from_pretrained('savasy/bert-turkish-text-classification', num_labels=len(unique_labels))
```

2.5 Evaluation Metrics

In addition to accuracy, precision, recall, F1 score values for all models, we also plotted ROC curve and confusion matrix graphs. Train/loss accuracy and loss graphs were also added for deep learning models.

3. **Results**

3.1 Classification Metrics

Part 1 (ML + ANN)

Results for BOW:
Model Evaluation Report
==

Model Comparison:

	Model	Accuracy	Precision	Recall	F1-Score	AUC
0	Logistic Regression	0.8929	0.8931	0.8929	0.8926	0.9828
1	XGBoost	0.8806	0.8818	0.8806	0.8803	0.9851
2	Decision Tree	0.7143	0.7191	0.7143	0.7141	0.8357
3	RandomForest	0.8653	0.8663	0.8653	0.8648	0.9830
4	K-NN	0.4939	0.6977	0.4939	0.5157	0.8205
5	LightGBM	0.8949	0.8957	0.8949	0.8948	0.9888
6	Neural Network	0.8949	0.8968	0.8949	0.8951	0.9875

Results for TF-IDF:
Model Evaluation Report
==

Model Comparison:

	Model	Accuracy	Precision	Recall	F1-Score	AUC
0	Logistic Regression	0.9143	0.9138	0.9143	0.9139	0.9922
1	XGBoost	0.8704	0.8710	0.8704	0.8701	0.9838
2	Decision Tree	0.7173	0.7236	0.7173	0.7193	0.8359
3	RandomForest	0.8735	0.8740	0.8735	0.8730	0.9837
4	K-NN	0.8327	0.8333	0.8327	0.8329	0.9625
5	LightGBM	0.8867	0.8874	0.8867	0.8864	0.9868
6	Neural Network	0.8949	0.9016	0.8949	0.8955	0.9906

Results for Word2Vec:
Model Evaluation Report
==

Model Comparison:

	Model	Accuracy	Precision	Recall	F1-Score	AUC
0	Logistic Regression	0.9112	0.9112	0.9112	0.9110	0.9940
1	XGBoost	0.9051	0.9060	0.9051	0.9054	0.9918
2	Decision Tree	0.7031	0.7047	0.7031	0.7031	0.8280
3	RandomForest	0.8806	0.8820	0.8806	0.8809	0.9877
4	K-NN	0.8694	0.8728	0.8694	0.8692	0.9690
5	LightGBM	0.9133	0.9141	0.9133	0.9135	0.9911
6	Neural Network	0.9112	0.9110	0.9112	0.9110	0.9918

Combined model

Performance Metrics Comparison:

	Metric	Random Embeddings	Word2Vec
0	Accuracy	0.8388	0.7316
1	Precision	0.8490	0.7381
2	Recall	0.8388	0.7316
3	F1-Score	0.8405	0.7318
4	Mean AUC	0.9761	0.9465

Multilingual BERT

Multilingual BERT

	precision	recall	f1-score	support
0	0.90	0.90	0.90	70
1	0.83	0.86	0.85	70
2	0.89	0.90	0.89	70
3	0.94	0.93	0.94	70
4	0.87	0.84	0.86	70
5	0.94	0.93	0.94	70
6	0.97	0.99	0.98	70
accuracy			0.91	490
macro avg	0.91	0.91	0.91	490
weighted avg	0.91	0.91	0.91	490

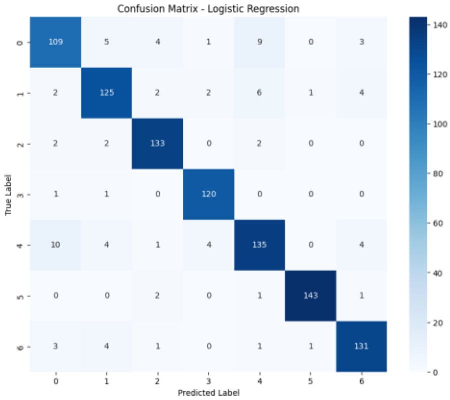
Fine-tuned BERT

Turkish BERT

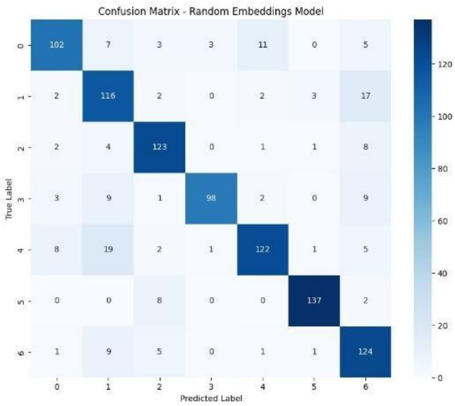
	precision	recall	f1-score	support
0	0.94	0.96	0.95	70
1	0.93	0.91	0.92	70
2	0.93	0.97	0.95	70
3	0.99	1.00	0.99	70
4	0.96	0.91	0.93	70
5	0.99	0.97	0.98	70
6	1.00	1.00	1.00	70
accuracy			0.96	490
macro avg	0.96	0.96	0.96	490
weighted avg	0.96	0.96	0.96	490

3.2 Confusion Matrix

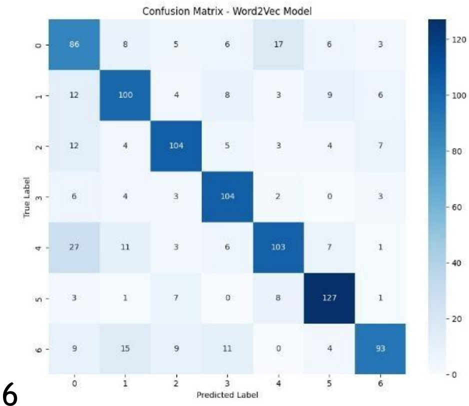
Best model of ML+ANN
(TD-IDF Logistic Regression)



Combined model with random embedding

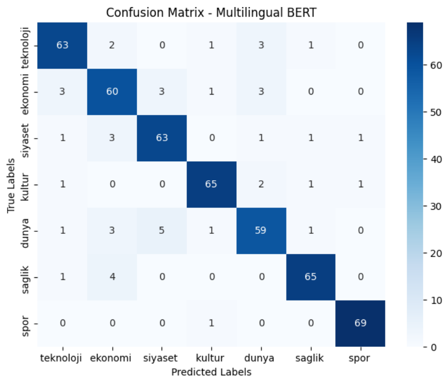


Combined model with Word2vec embedding

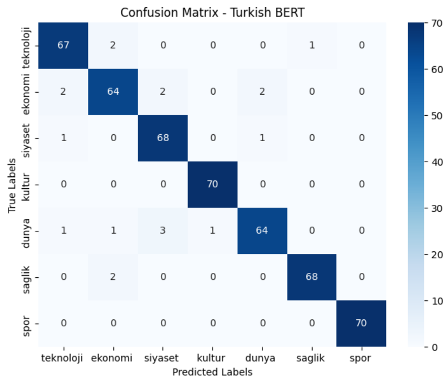


6

Multilingual BERT

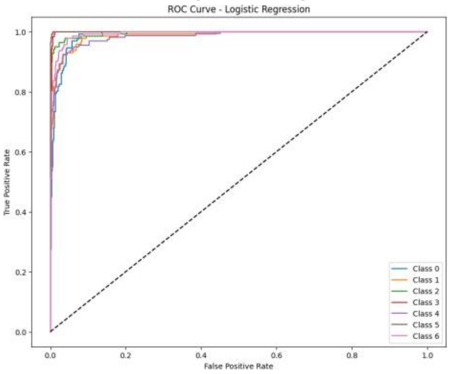


Fine-tuned BERT

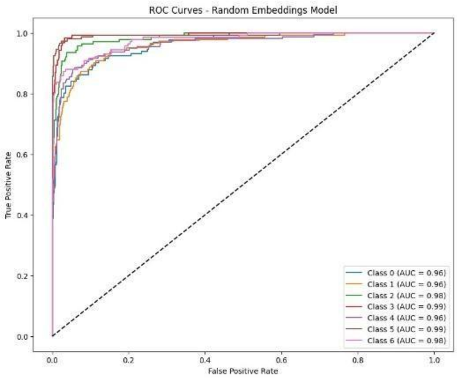


3.3 ROC – AUC

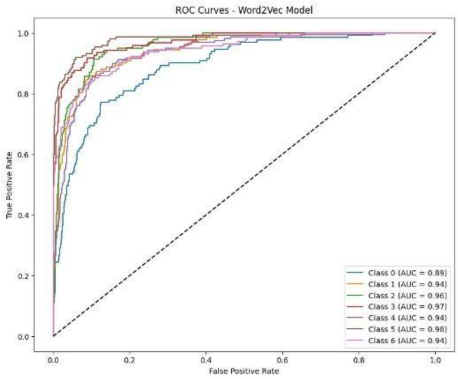
**Best model of ML+ANN
(TD-IDF Logistic Regression)**



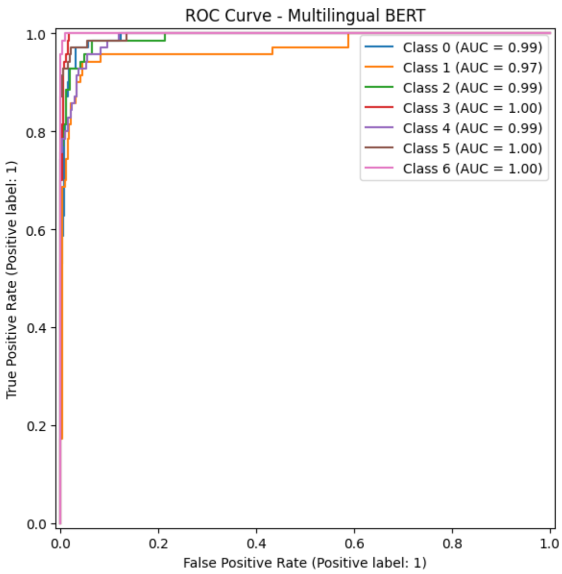
Combined model with random embedding

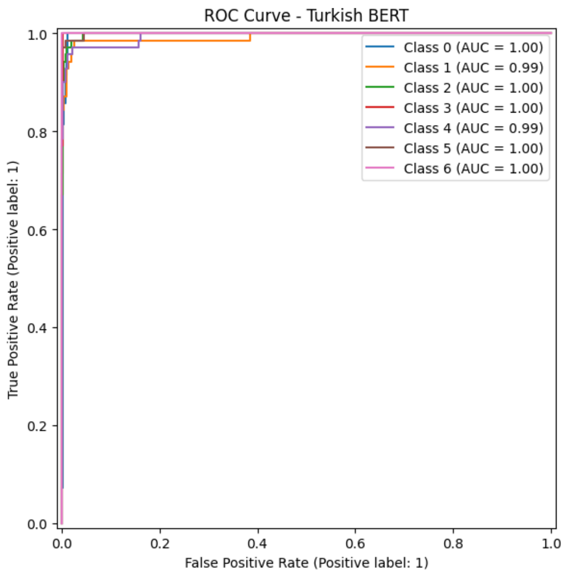


Combined model with Word2vec embedding

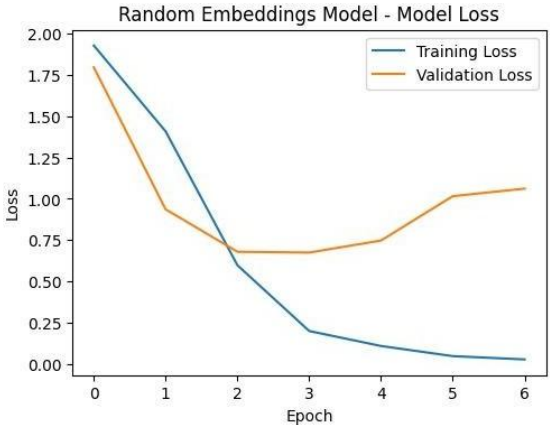


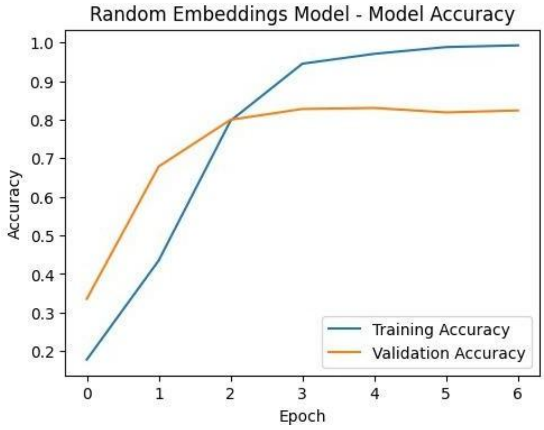
BERT

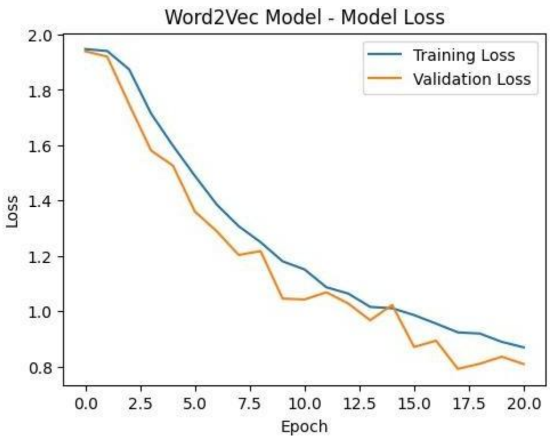


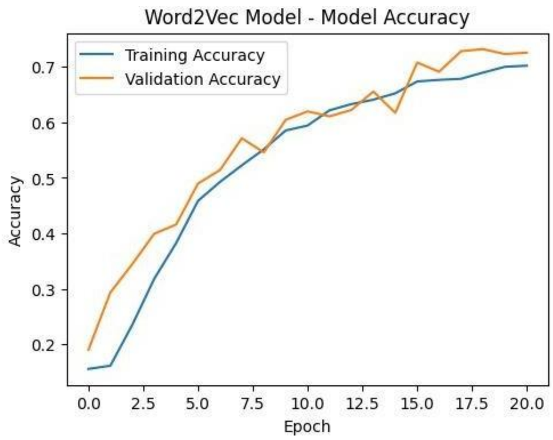


3.4 Train/Test Accuracy Loss









Multilingual BERT

Epoch	Training Loss	Validation Loss	Accuracy
1	0.352700	0.358740	0.906122
2	0.258900	0.276823	0.922449
3	0.174800	0.276628	0.924490

Fine-tuned BERT

Epoch	Training Loss	Validation Loss	Accuracy
1	0.098500	0.174488	0.961224
2	0.091500	0.161177	0.967347
3	0.013600	0.167012	0.965306

4. **Analyzing**

In this study, various machine learning and deep learning models were used to classify Turkish news texts. The dataset used has a balanced distribution and contains equal number of data in each category. This allowed us to fairly evaluate the performance of the models.

In the pre-processing stage, techniques such as tokenization and lemmatization were applied to the texts, and Bag of Words (BoW), TF-IDF and Word2Vec methods were used for feature extraction. Then, classification was performed using traditional machine learning algorithms such as logistic regression, XGBoost, decision tree, random forest, K-NN and LightGBM and deep learning models such as CNN, LSTM, BiLSTM, GRU.

Part 1: Machine Learning and Artificial Neural Networks (ML + ANN)

When machine learning and artificial neural network (ANN) models were compared, the best result was obtained in the logistic regression model with TF-IDF feature extraction. This model showed the highest performance with 91% accuracy rate. This result shows that logistic regression is an effective method for classifying Turkish news texts when used with TF-IDF.

Part 2: Combined Methods

When the experiments conducted with combined models using word2vec embedding and random embedding were compared, models trained with random embeddings gave better results. This approach allowed the model to learn word meanings from scratch from the training data, and thus higher performance was achieved. Random embeddings allowed the model to discover word meanings specific to the training data.

Part 3: BERT Models

Among the BERT-based models, the fine-tuned BERT model, which was specifically trained for Turkish texts, gave better results. This model was trained by focusing on the grammatical features of the Turkish language and helped to obtain more accurate results. The fine-tuned BERT model showed higher performance in Turkish text classification.

Model performances were evaluated with metrics such as accuracy, precision, recall and F1 score, and the most successful model was determined. In addition, ROC curve and complexity matrix graphs were plotted. The results revealed that deep learning models, especially BERT-based models, showed higher performance in Turkish text classification.

This study aims to determine the most effective approach by comparing different methods in the process of classifying Turkish news texts.

5. References

- https://www.kaggle.com/datasets/savasy/ttc4900
- https://www.kaggle.com/code/alperenclk/for-beginner-nlp-and-word2vec
- https://www.kaggle.com/code/erdal002/turkish-text-classification
- https://www.kaggle.com/code/ayhanc/bert-multilingual-for-turkish-text-classification
- https://huggingface.co/savasy/bert-turkish-text-classification
- https://arxiv.org/pdf/2401.17396