T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4288 Introduction to Machine Learning – Term Project

"Predicting Outcomes of Turkish Constitutional Court Decisions Using Explainable Artificial Intelligence"

Group Members

130319659 - Cihan Erdoğanyılmaz

150120998 - Abdelrahman Zahran

150120997 - Mohamad Nael Ayoubi

150120079 - Omar Sameh Belal

150120055 - Muhammed Talha Karagül

1. Abstract

This project aims to predict the outcomes of Turkish Constitutional Court decisions using machine learning and explainable AI methods. Various models, including Logistic Regression, Naive Bayes, Decision Tree, Random Forest, and Transformers, were employed to classify decisions as "violation" or "no violation." Data preprocessing, including stop word removal, lemmatization, and feature engineering, ensured robust text analysis. Results demonstrate the potential of combining traditional and modern machine learning approaches in the legal domain.

2. Introduction

The judicial system generates substantial textual data, presenting opportunities for automation and prediction. This project focuses on predicting Turkish Constitutional Court decision outcomes. We address challenges such as class imbalance and legal domain-specific preprocessing while evaluating various machine learning methods for explainability and performance.

3. Methodology

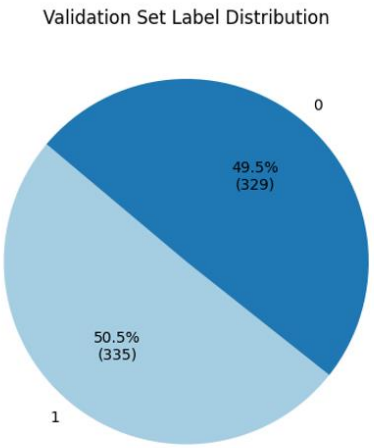
Data Collection

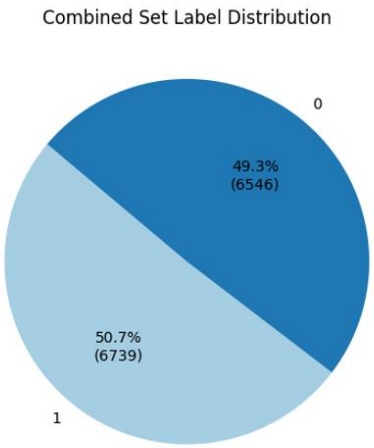
The dataset consists of 13,676 individual application decisions, cleaned to retain 13,286 consistent records focused on binary classification ("violation" and "no violation").

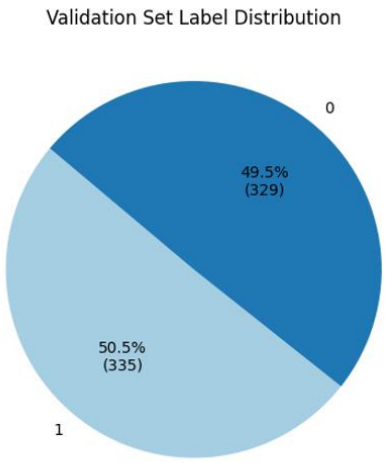
Data Preprocessing

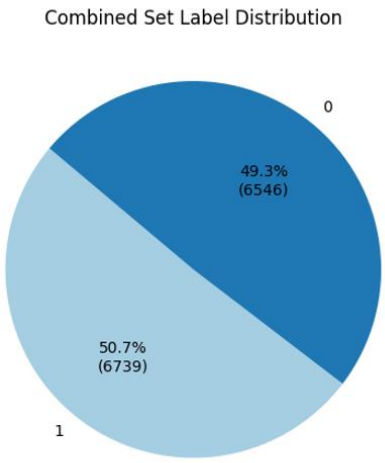
1. **Cleaning Steps**
    - Excluded decisions with multiple inconsistent outcomes.
    - Retained only decisions relevant to the first two significant rights.
2. **Text Preprocessing**
    - Converted text to lowercase.
    - Removed punctuation, irrelevant symbols, and domain-specific stop words (e.g., common Turkish and legal terms).

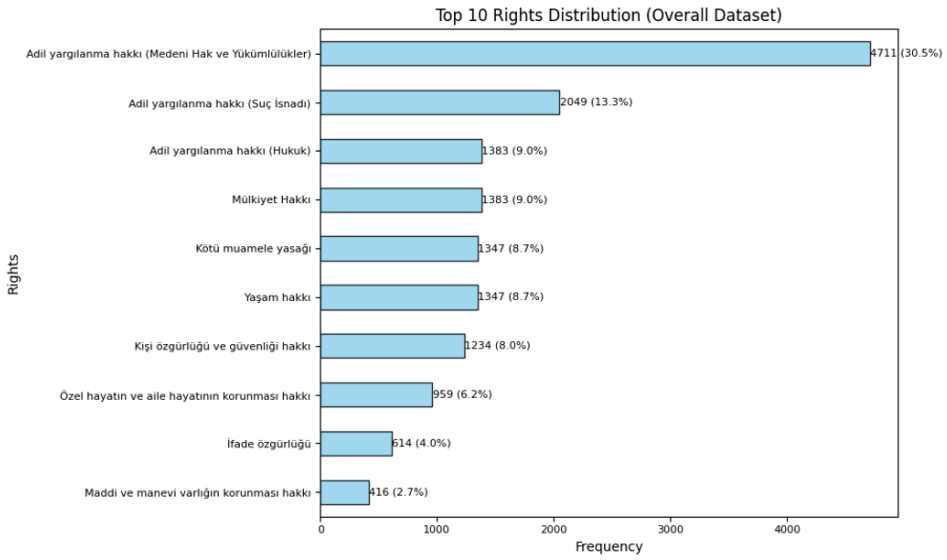
Distribution of Dataset











- Implemented lemmatization for root word normalization.

3. **Feature Engineering**
    - Used TF-IDF for traditional models and pre-trained tokenizers for Transformers.

TF-IDF for Traditional Models

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document relative to a collection (or corpus) of documents. It is a widely used feature extraction technique for traditional machine learning models in text classification tasks.

Steps Involved

1. **Tokenization**:
   The text was split into individual tokens (words or phrases) to create a vocabulary for the dataset.
2. **Term Frequency (TF)**:
   Calculated as the frequency of a term appearing in a document.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

3. **Inverse Document Frequency (IDF)**:
   Assessed the rarity of a term across all documents. Rare terms were given higher weights.

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

4. **TF-IDF Score**:
   The final TF-IDF value for each term was calculated as: Product of its TF and IDF scores
5. **Why TF-IDF?**

- TF-IDF effectively balances frequent terms (e.g., "and," "the") and rare but significant terms in the context (e.g., legal terms like "violation," "admissibility").

- It provides a sparse, numerical representation of text, suitable for traditional machine learning models such as Logistic Regression, Naive Bayes, Decision Tree, and Random Forest.

Implementation

- **Libraries Used**: Scikit-learn was used for generating the TF-IDF matrix.
- **Parameters Tuned**:
    - max_features: Limited the number of features to 5000 to reduce dimensionality.

Modeling Techniques

We used a 90-5-5 train-dev-test split. The models were trained using 90% of the data, evaluated on 5% as the development set for hyperparameter tuning, and finally tested on 5% of the data for the final evaluation.

1. **Logistic Regression** (handled by Omer)
    - Baseline model for binary classification with weighted loss for imbalanced data.
2. **Naive Bayes** (handled by Muhammed Talha)
    - Efficient probabilistic model for high-dimensional text data.
3. **Decision Tree** (handled by Abdelrahman)
    - Captures decision-making logic, aiding explainability.
4. **Random Forest** (handled by Muhammed Nael)
    - Ensemble model combining multiple decision trees for improved accuracy.
5. **Transformers** (handled by Cihan)
    - Fine-tuned pre-trained BERT models for contextual understanding of legal texts.

Evaluation Metrics

Models were evaluated using accuracy, precision, recall, and F1 score.

4. Results

1. Model Performance

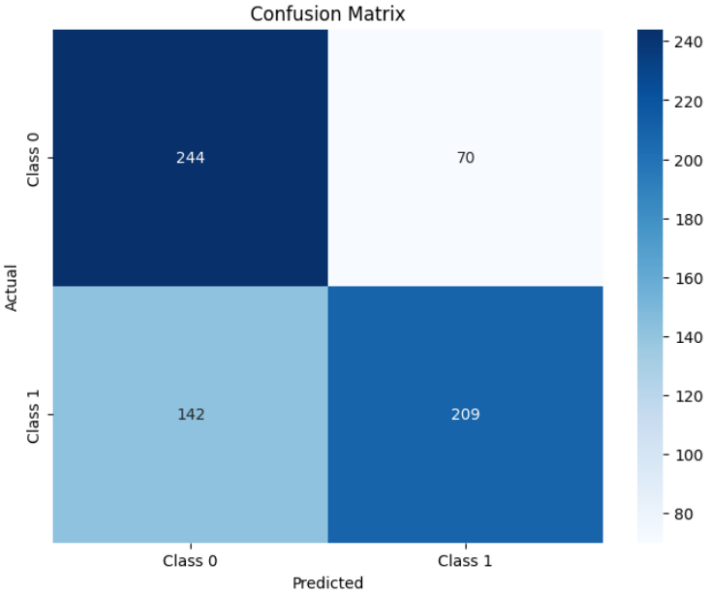
- Logistic Regression achieved baseline accuracy but struggled with class imbalance.
- Naive Bayes demonstrated speed and efficiency but lacked nuanced understanding.
- Decision Tree provided interpretable insights but showed lower generalization.
- Random Forest achieved high accuracy and robustness against overfitting.
- Transformers outperformed traditional models, excelling in contextual understanding and accuracy.

Model	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Class 0)	F1-Score (Class 1)
Naive Bayes	0.67	0.63	0.73	0.76	0.60	0.69	0.66
Logistic Regression	0.78	0.74	0.82	0.82	0.74	0.78	0.78
Decision Tree	0.74	0.79	0.70	0.59	0.86	0.68	0.77
Random Forest	0.78	0.73	0.85	0.86	0.71	0.79	0.77
Transformer (Turkish BERT)	0.80	0.80	0.81	0.78	0.82	0.79	0.81

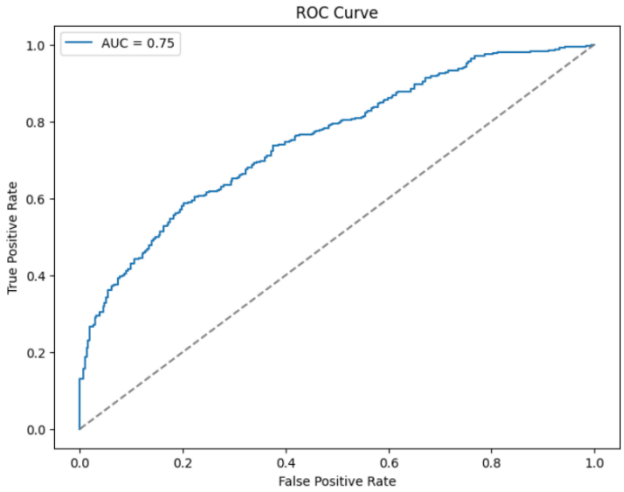
Why Transformers Outperform Traditional Models:

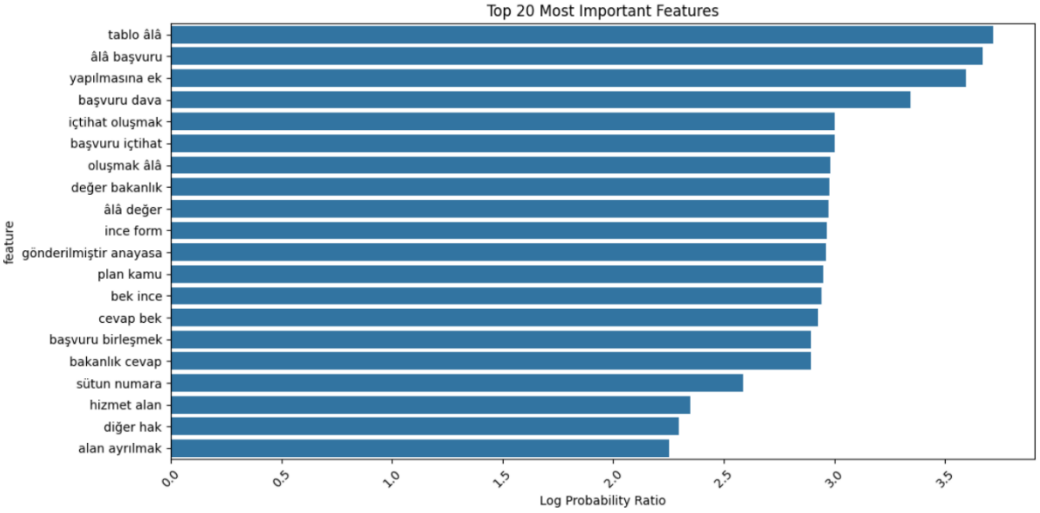
- **Contextual Understanding**: Transformers use self-attention mechanisms to understand relationships between all words in a sentence, allowing for nuanced comprehension of language.
- **Pre-trained Knowledge**: Models like BERT and GPT are pre-trained on vast corpora, equipping them with domain knowledge that traditional models lack.
- **Adaptability**: Fine-tuning on a specific task enables Transformers to capture subtle domain-specific patterns.

4.1 Naïve Bayes Results

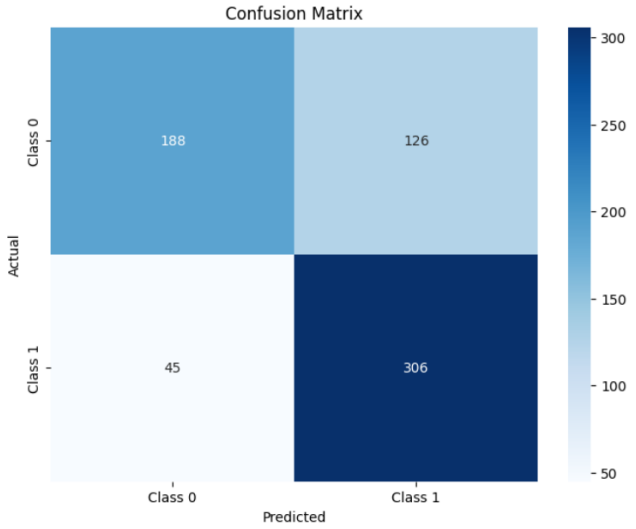


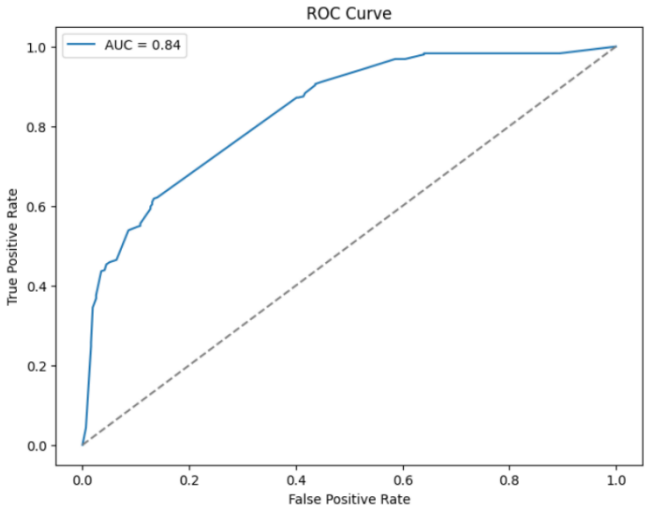

```
Accuracy: 0.68
Cross-Validation Scores: [0.66336634 0.67630745 0.67749604 0.69413629 0.68581616]
Mean CV Accuracy: 0.68
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.63      0.78      0.70       314
     Class 1       0.75      0.60      0.66       351

    accuracy                           0.68       665
   macro avg       0.69      0.69      0.68       665
weighted avg       0.69      0.68      0.68       665

ROC-AUC Score: 0.75

Top 10 Most Discriminative Features:
             feature  importance
3839        tablo âlâ    3.718489
4734      âlâ başvuru    3.670060
4439   yapılmasına ek    3.593095
461      başvuru dava    3.343185
2254  içtihat oluşmak    3.001016
466   başvuru içtihat    3.000796
3272      oluşmak âlâ    2.980506
1045   değer bakanlık    2.977153
4735        âlâ değer    2.974209
2129        ince form    2.966180
```


ROC Curve
AUC = 0.75
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Top 20 Most Important Features
tablo âlâ
âlâ başvuru
yapılmasına ek
başvuru dava
içtihat oluşmak
başvuru içtihat
oluşmak âlâ
değer bakanlık
âlâ değer
ince form
gönderilmiştir anayasa
plan kamu
bek ince
cevap bek
başvuru birleşmek
bakanlık cevap
sütun numara
hizmet alan
diğer hak
alan ayrılmak
feature
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Log Probability Ratio

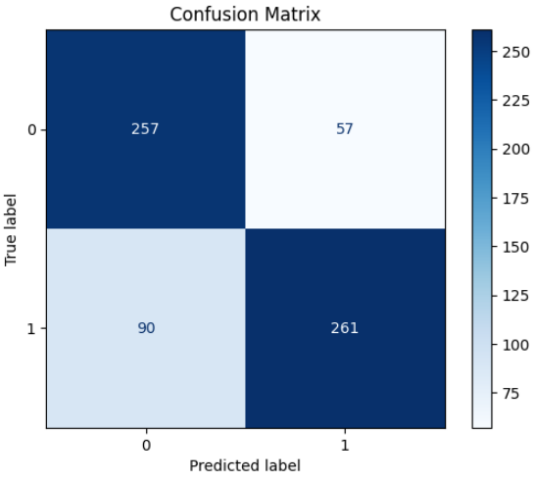
4.2 Decision Tree Results

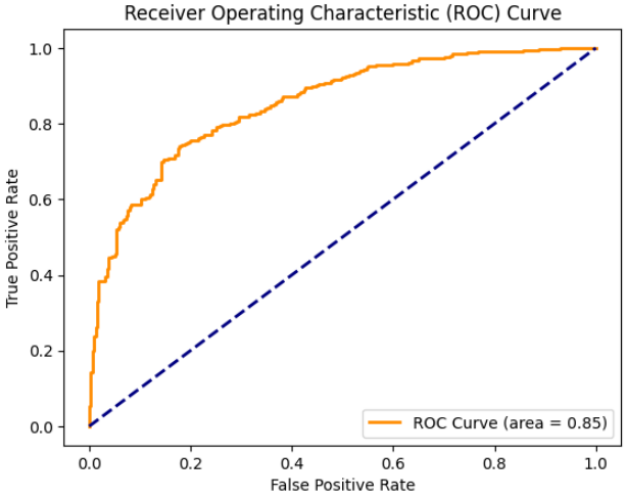




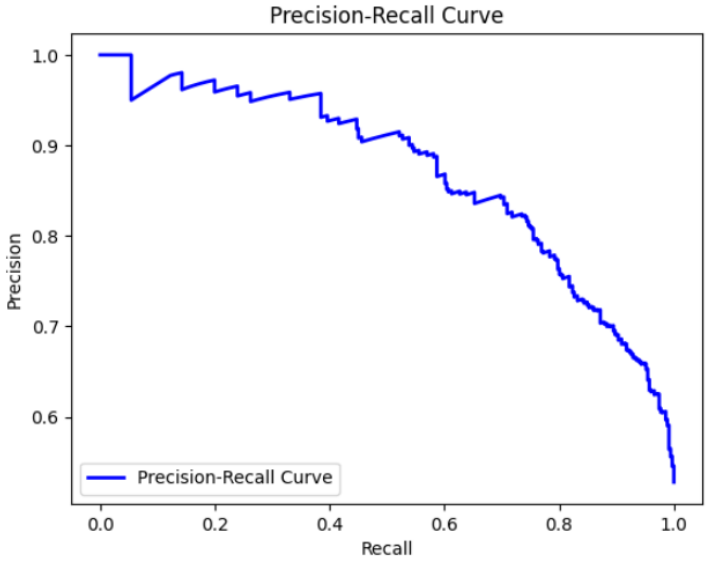

```
Accuracy: 0.74
Cross-Validation Scores: [0.72277228 0.7274168  0.73137876 0.74167987 0.75158479]
Mean CV Accuracy: 0.73
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.81      0.60      0.69       314
     Class 1       0.71      0.87      0.78       351

    accuracy                           0.74       665
   macro avg       0.76      0.74      0.73       665
weighted avg       0.75      0.74      0.74       665

ROC-AUC Score: 0.84
```

4.3 Logistic Regression Results

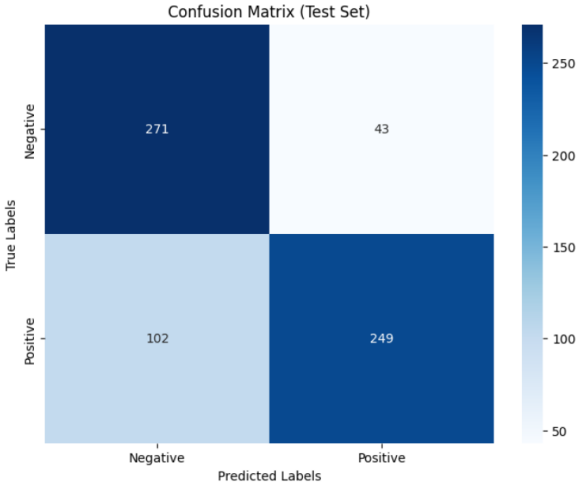


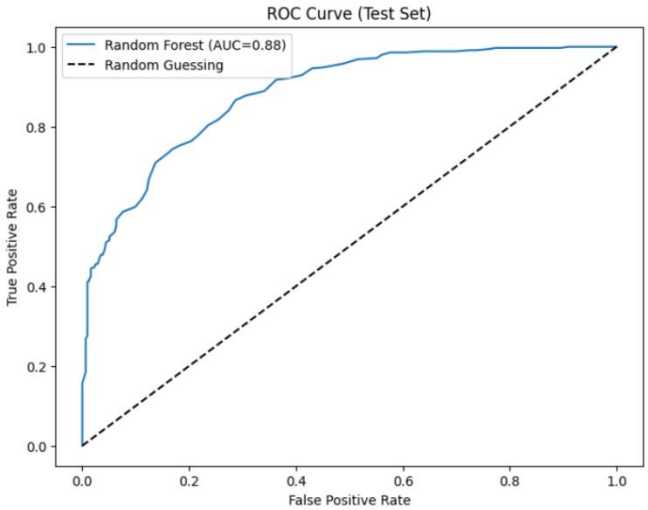


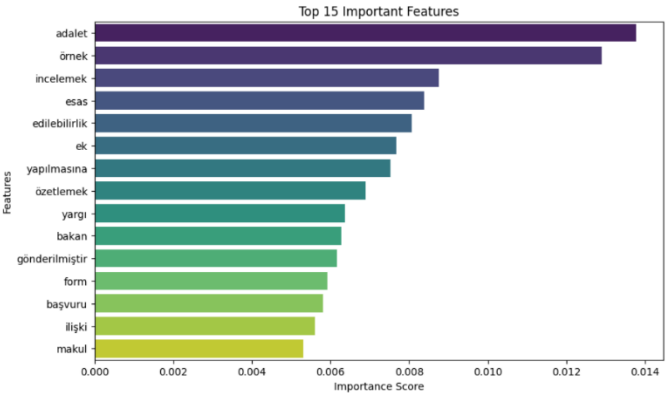

```
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.82      0.78       314
           1       0.82      0.74      0.78       351

    accuracy                           0.78       665
   macro avg       0.78      0.78      0.78       665
weighted avg       0.78      0.78      0.78       665
```

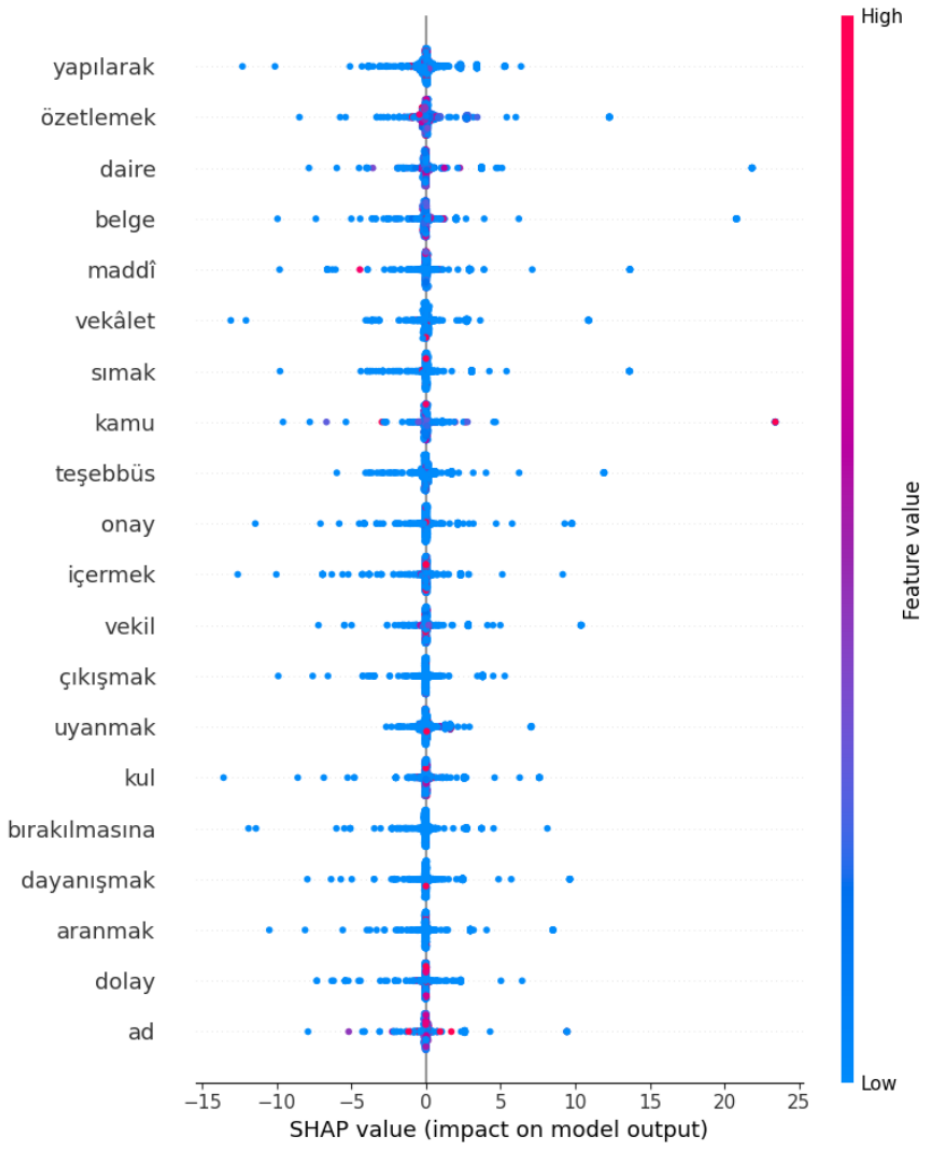


4.4 Random Forest Results





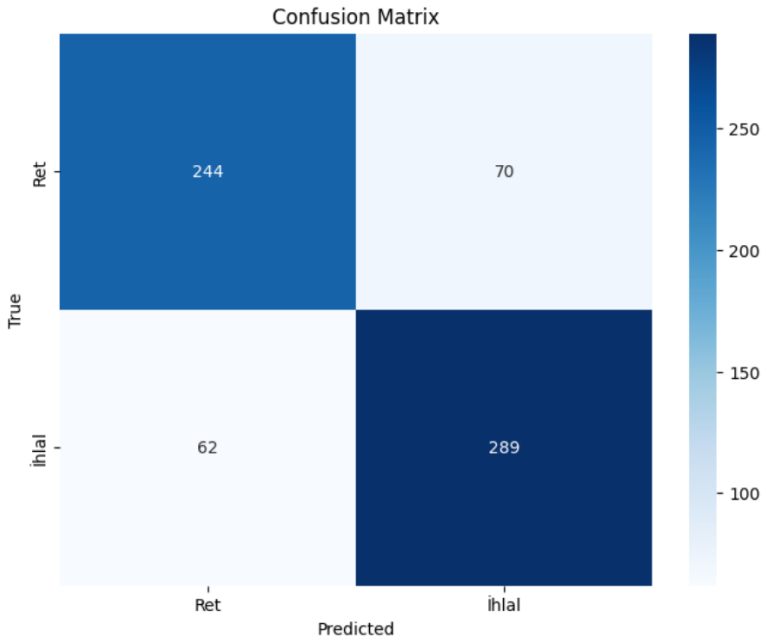

Top 15 Important Features
adalet
örnek
incelemek
esas
edilebilirlik
ek
yapılmasına
özetlemek
yargı
bakan
gönderilmiştir
form
başvuru
ilişki
makul
Features
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
Importance Score


yapılarak
özetlemek
daire
belge
maddî
vekâlet
sımak
kamu
teşebbüs
onay
içermek
vekil
çıkışmak
uyanmak
kul
bırakılmasına
dayanışmak
aranmak
dolay
ad
−15
−10
−5
0
5
10
15
20
25
SHAP value (impact on model output)
High
Feature value
Low

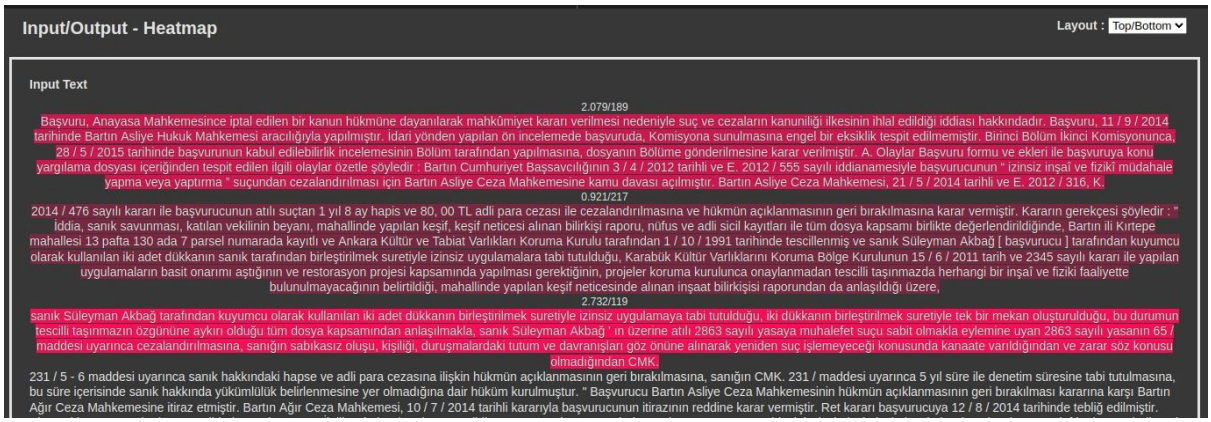
```
Cross-validation scores: [0.74064289 0.7685702  0.77333333 0.78544542 0.76178235]
Average CV score: 0.7659548383917676
CV score std: 0.01482747164500557

Test Set Performance:
              precision    recall  f1-score   support

           0       0.73      0.86      0.79       314
           1       0.85      0.71      0.77       351

    accuracy                           0.78       665
   macro avg       0.79      0.79      0.78       665
weighted avg       0.79      0.78      0.78       665
```

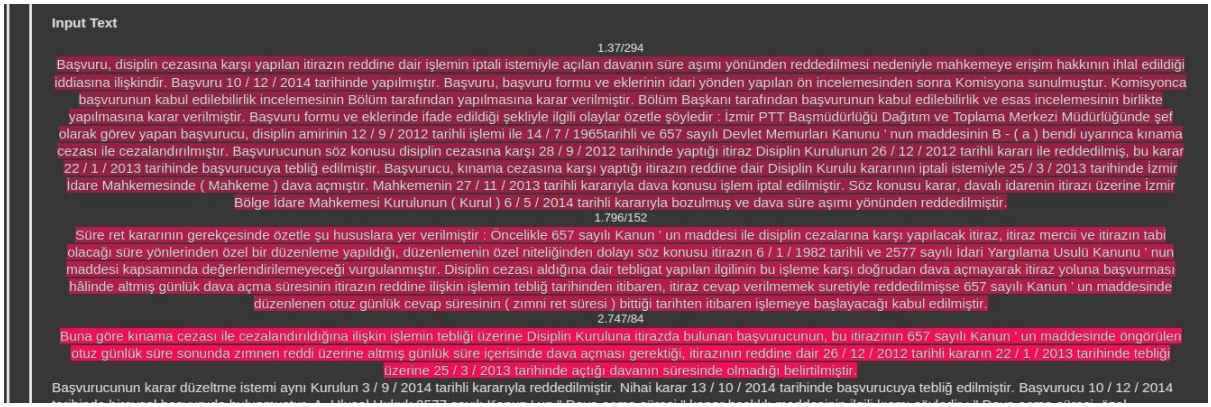
4.5 Transformer (Turkish BERT)

```
Accuracy: 0.8015037593984963
Precision: 0.8014119902958007
Recall: 0.8015037593984963
F1 Score: 0.8013414271708348

Classification Report:

              precision    recall  f1-score   support

         Ret       0.80      0.78      0.79       314
       İhlal       0.81      0.82      0.81       351

    accuracy                           0.80       665
   macro avg       0.80      0.80      0.80       665
weighted avg       0.80      0.80      0.80       665
```

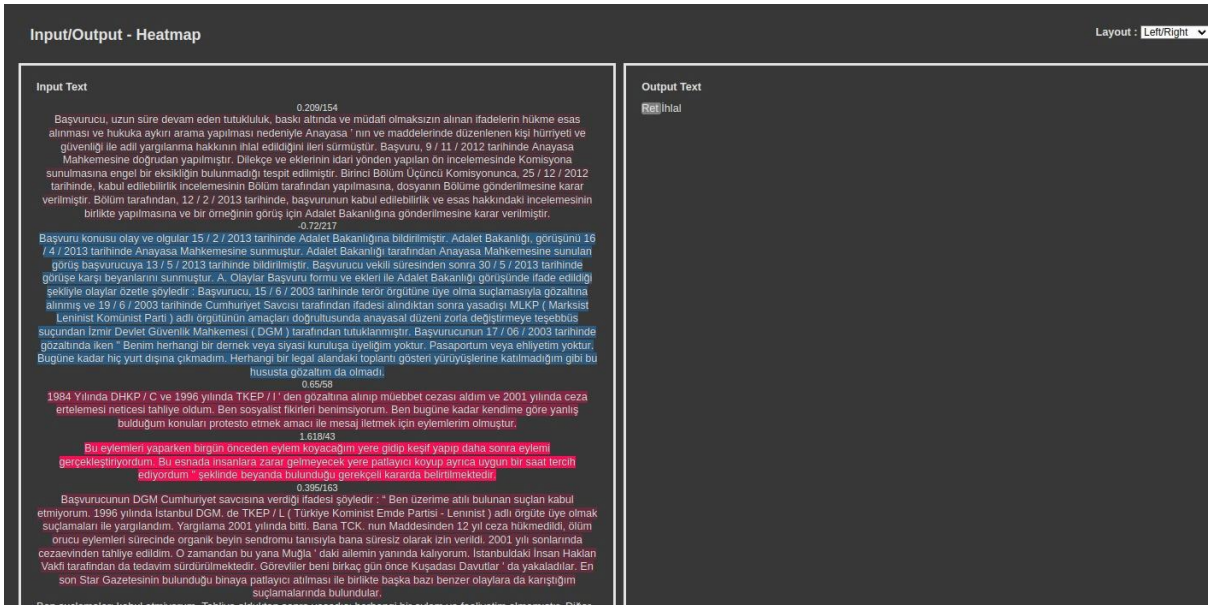


XAI Visualizations





Misclassified instance



5. Discussion

- Challenges

- **Legal-specific preprocessing**: Processing legal texts, including tasks such as Turkish stopword removal, lemmatization, and domain-specific tokenization, was time-intensive but necessary for improving model accuracy.

- Interpretation

- **Traditional models**: Logistic Regression, Naive Bayes, Decision Tree, and Random Forest models performed adequately. Among these, **Random Forest** provided the best balance of **explainability** and **performance**, offering a strong trade-off between interpretability and predictive power.
- **Transformers (Turkish BERT)**: While the transformer model excelled in terms of **accuracy** and demonstrated **superior contextual understanding**, it was **computationally intensive** and required significant hardware resources for training and inference.

- Solutions

- **Domain-specific preprocessing**: The application of legal-specific preprocessing, including **Turkish stopword removal** and **lemmatization**, improved the models' relevance and performance in legal text analysis.

6. Conclusion

This project demonstrates the potential of machine learning in legal text analysis. Combining traditional and modern approaches improved prediction accuracy and interpretability. Future work could explore ensemble strategies combining Transformers with Random Forest or other traditional models. Further domain adaptation and model fine-tuning will enhance practical application.

7. References

[1] E. Mumcuoglu, C. E. Öztürk, H. M. Ozaktas, and A. Koç, “Natural language processing in law: Prediction of outcomes in the higher courts of Turkey,” *Information Processing & Management*, vol. 58, no. 5, p. 102684, 2021.

[2] A. C. Aras, C. E. Öztürk and A. Koç, "Feedforward Neural Network Based Case Prediction in Turkish Higher Courts," *2022 30th Signal Processing and Communications Applications Conference (SIU)*, Safranbolu, Turkey, 2022, pp. 1-4.

[3] M. F. Sert, E. Yıldırım, and İ. Haşlak, “Using artificial intelligence to predict decisions of the Turkish Constitutional Court,” *Social Science Computer Review*, vol. 40, no. 6, pp. 1416–1435, 2021.

[4] E. Aydemir, "Estimation of Turkish Constitutional Court Decisions in Terms of Admissibility with NLP," *2023 IV International Conference on Neural Networks and Neurotechnologies (NeuroNT)*, Saint Petersburg, Russian Federation, 2023, pp. 17-20.

[5] O. Akça, G. Bayrak, A. M. Issifu and M. C. Ganiz, "Traditional Machine Learning and Deep Learning-based Text Classification for Turkish Law Documents using Transformers and Domain Adaptation," *2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, Biarritz, France, 2022, pp. 1-6.

[6] C. E. Öztürk, Ş. B. Özçelık and A. Koç, "Predicting Outcomes of the Court of Cassation of Turkey with Recurrent Neural Networks," *2022 30th Signal Processing and Communications Applications Conference (SIU)*, Safranbolu, Turkey, 2022, pp. 1-4.

[7] M. B. L. Virtucio et al., "Predicting Decisions of the Philippine Supreme Court Using Natural Language Processing and Machine Learning," *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Tokyo, Japan, 2018, pp. 130-135.

[8] O.-M. ¸Sulea, M. Zampieri, M. Vela, and J. van Genabith, “Predicting the law area and decisions of French Supreme Court cases,” in *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, Varna, Bulgaria, Sep. 2017, pp. 716–722.

[9] C. Erdoğanyımaz and B. Mengünoğul, "An Original Natural Language Processing Approach to Language Modeling in Turkish Legal Corpus: Improving Model Performance with Domain Classification by Using Recurrent Neural Networks," *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Antalya, Turkey, 2022, pp. 1-6.

[10] Erdoğanyılmaz, C., Mengünoğul, B., & Balci, M. (2023, September). Unveiling the Black Box: Investigating the Interplay between AI Technologies, Explainability, and Legal Implications. In 2023 8th International Conference on Computer Science and Engineering (UBMK) (pp. 569-574). IEEE.

[11] Turan, T., Küçüksille, E., & Alagöz, N. K. (2023). Prediction of Turkish Constitutional Court Decisions with Explainable Artificial Intelligence. Bilge International Journal of Science and Technology Research, 7(2), 128-141.

[12] A.Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[13] E. Roth, *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge, U.K.: Cambridge Univ. Press, 1988.

[14] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.

8. Appendices

- **Code Snippets**

Code example of using shap library for extracting features that have the higher impact

```
import shap
import numpy as np

# ensure the test set is in dense format for SHAP
X_test_dense = np.array(X_test)
X_test_dense = X_test_vectorized
explainer = shap.TreeExplainer(clf, feature_perturbation='interventional')

# generate SHAP values which measure the impact of each feature on the model output. The SHAP values are calculated for each class
shap_values = explainer.shap_values(X_test_dense, check_additivity=False)
feature_names = vectorizer.get_feature_names_out()
print(f"Number of features: {len(feature_names)}, Data columns: {X_test_dense.shape[1]}")
print(f"Shape of shap_values[1]: {shap_values[1].shape}")
print(f"Shape of X_test_dense: {X_test_dense.shape}")
```

✓ 4m 55.4s

```
Number of features: 5000, Data columns: 5000
Shape of shap_values[1]: (5000, 2)
Shape of X_test_dense: (665, 5000)
```

```
# Take values for class 1 (index 1)
selected_shap_values = shap_values[:, :, 1]  # (n_samples, n_features)
print(f"Selected SHAP values shape: {selected_shap_values.shape}")
print(f"X_test_dense shape: {X_test_dense.shape}")
```

✓ 0.0s

```
Selected SHAP values shape: (665, 5000)
X_test_dense shape: (665, 5000)
```

```
shap.summary_plot(
    selected_shap_values,
    X_test_dense,
    feature_names=feature_names
)
```

✓ 0.7s

Model training, performing cross validation and lastly predicting code example:

```
# Load data
train_data = load_data('train_processed.json')
dev_data = load_data('dev_processed.json')
test_data = load_data('test_processed.json')

# Combine train and dev data
train_dev_data = pd.concat([train_data, dev_data])
X_train_dev = train_dev_data['text']
y_train_dev = train_dev_data['labels']
X_test = test_data['text']
y_test = test_data['labels']

# Vectorize text data using TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X_train_dev_vectorized = vectorizer.fit_transform(X_train_dev).toarray()
X_test_vectorized = vectorizer.transform(X_test).toarray()

# Initialize and train Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)

cv_scores = cross_val_score(clf, X_train_dev_vectorized, y_train_dev, cv=5, scoring='f1')
print("\nCross-validation scores:", cv_scores)
print("Average CV score:", cv_scores.mean())
print("CV score std:", cv_scores.std())

# Train final model on full training data
clf.fit(X_train_dev_vectorized, y_train_dev)

# Evaluate on test set
y_pred = clf.predict(X_test_vectorized)
y_pred_proba = clf.predict_proba(X_test_vectorized)[:, 1]
```