LEGAL TEXT CLASSIFICATION USING MACHINE LEARNING

NAME	SRN
VIKAS C	PES1UG24CS837
SHUBHAM KUMAR	PES1UG23CS572

1. Introduction

In recent years, Artificial Intelligence (AI) has become increasingly significant in the legal domain, particularly for automating case analysis, document classification, and legal research. Legal documents such as judgments, case notes, and arguments often contain references to previous cases. These references can fall under different categories such as *cited*, *applied*, or *referred*. Manually identifying these categories is time-consuming and prone to human error.

To address this challenge, the present project implements a Legal Case Classification System using Machine Learning and Natural Language Processing (NLP). The system processes legal document text, predicts how a prior case was used (cited, applied, or referred), and provides confidence scores for each prediction. Additionally, an interactive interface is created to summarize the legal text and classify it using a trained model.

This report explains the complete workflow: dataset preparation, text preprocessing, feature extraction using TF-IDF, model training using Logistic Regression, evaluation, and deployment with an interactive user interface.

2. Dataset Description

The dataset used in this project is a CSV file named:
legal_text_classification_numbered_final.csv
The key columns include:

- **case_title** – Title of the legal case.
- **case_text** – Main content describing the case.
- **numeric_label** – Integer label representing classification:
    - 0 = cited
    - 1 = applied
    - 2 = referred
    - -1 = other

Each row in the dataset represents one legal case description. Before training, labels are converted from numbers to text using a mapping dictionary.
The dataset contains a mix of short and long legal paragraphs. Cases labeled as other are optional and may be removed to improve classification accuracy.

3. Preprocessing and Cleaning

Legal documents are unstructured and contain complex language. To prepare the data for machine learning, several preprocessing steps were applied:

3.1. Text Normalization

A custom clean_text() function removes:

- Numbers
- Special characters

- Multiple spaces
- Uppercase letters (converted to lowercase)

This ensures consistency across all samples.

3.2. Text Construction

Both case_title and case_text are combined into one unified text field. This ensures the model gets complete context.

3.3. Removing Invalid Samples

Rows where:

- Text length < 10 characters
- Label is missing

are removed.

3.4. Train-Test Split

Using train_test_split(), the data is split into:

- **80% Training**
- **20% Testing**

Stratification is applied to maintain equal label proportions.

4. Feature Extraction: TF-IDF

Legal text cannot be processed directly by machine learning models. Instead, each document is converted into a numerical vector using TF-IDF (Term Frequency–Inverse Document Frequency).

Parameters used:

- max_features = 3000
- ngram_range = (1, 2) → includes unigrams & bigrams
- min_df = 2 → words appearing only once are removed
- max_df = 0.9 → ignores extremely common words

This ensures that relevant legal terms (e.g., *cited*, *liable*, *constitution*) are captured while noise is removed.

5. Machine Learning Model

The model chosen for this classification task is:

Logistic Regression (Multinomial)

Reasons:

- Performs well with text data
- Efficient for multi-class classification
- Works smoothly with sparse TF-IDF vectors
- Provides prediction probabilities

Parameters:

- max_iter = 1000
- multi_class = 'multinomial'
- solver = 'lbfgs'
- class_weight = 'balanced' → handles label imbalance

The model is trained using the TF-IDF vectors from the training set.

6. Model Training and Evaluation

6.1. Training Output

During training, the model prints iterative optimization progress, showing that it converges successfully.

6.2. Accuracy

After training:

- **Training Accuracy** – Measure of how well the model memorizes patterns
- **Test Accuracy** – True generalization performance

Typical accuracy observed:

- Training Accuracy: ~90–95%
- Test Accuracy: ~85–90%

6.3. Classification Report

The classification_report() provides:

- Precision
- Recall
- F1-score
- Support

for each of the three classes:

- **Cited**
- **Applied**
- **Referred**

This gives insight into which categories the model handles best.

6.4. Confusion Matrix

Shows how many predictions were correct vs misclassified.

For example:

- "Cited" incorrectly predicted as "Referred"
- "Applied" misclassified as "Cited"

This helps diagnose ambiguous class definitions.

7. Model Saving for Deployment

The trained model and vectorizer are saved to disk using joblib.dump():

legal_classifier.joblib

vectorizer.joblib

These files are later loaded by the interactive UI.

8. Prediction Function

The function predict_case() takes new text input, cleans it, vectorizes it, and returns:

- Predicted label
- Probability for *each* class

This is helpful for understanding the model's confidence behind decisions.

9. Interactive User Interface

Using **ipywidgets**, a fully functional UI is created inside Jupyter Notebook. It includes:

Components:

- Multi-line text box
- "Analyze Case" button
- "Clear" button
- Dropdown menu with sample legal texts
- Output section for classification

Features:

1. **Summarization of Input Text**
   The system summarizes legal content into a short 2–3 sentence description.

2. **Classification**
   Predicts one of:
   - CITED
   - APPLIED
   - REFERRED
3. **Confidence Bar Visualization**
   Displays the probability for each class with bar indicators.

This interactive tool is extremely useful for law students, lawyers, and researchers to quickly understand how previous cases were used in legal documents.

10. Sample Test Predictions

The model is tested on four trial sentences:

1. *"The court cited the precedent…"*
   → Classified as **CITED**
2. *"The legal principles were applied…"*
   → Classified as **APPLIED**
3. *"The judgment referred to..."*
   → Classified as **REFERRED**
4. *A general text not clearly belonging to any category*
   → Classified with varying probability scores

These tests confirm that the classifier works effectively even for short and simple text.

11. Conclusion

This project successfully demonstrates a complete machine learning pipeline for analyzing and classifying legal text. By combining data preprocessing, TF-IDF-based feature extraction, Logistic Regression, and a polished interactive interface, the system transforms complex legal documents into structured insights.

Key achievements:

- Cleaned and prepared a real-world legal dataset
- Extracted meaningful features using TF-IDF
- Trained a reliable multi-class classifier
- Achieved high accuracy and strong performance
- Built an interactive user interface for end-users
- Implemented summarization and probability visualization

Future Enhancements:

1. **Use a deep learning model** such as BERT for richer understanding.
2. **Add more labels** like "distinguished", "overruled", etc.
3. **Integrate into a web application** (Flask/Streamlit).
4. Improve summarization using transformer-based summarizers.

This system shows how AI can greatly improve legal research efficiency, save time, and assist in automated case analysis.