CaseNext2

April 20, 2025

[20]:
```python
import sys
import warnings
warnings.filterwarnings('ignore')

# ====================== VERSION COMPATIBILITY CHECK ======================
try:
    import numpy as np
    import pandas as pd
    from sklearn import __version__ as sklearn_version

    # Check versions
    numpy_version = np.__version__
    pandas_version = pd.__version__

except ImportError as e:
    print(f"Critical error: {str(e)}")
    print("Please install required packages using:")
    print("pip install numpy pandas scikit-learn openpyxl")
    sys.exit(1)

# ====================== MAIN IMPORTS ======================
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import classification_report, accuracy_score

# ====================== FUNCTION DEFINITIONS ======================
def load_data():
    """Load data from CSV or Excel file with proper error handling"""
    try:
        file_path = input("Enter path to data file (CSV or Excel): ").strip()
        # Remove surrounding quotes if present
        file_path = file_path.strip('"\'')
```

```
        if file_path.lower().endswith('.csv'):
            df = pd.read_csv(file_path)
        elif file_path.lower().endswith(('.xls', '.xlsx')):
            df = pd.read_excel(file_path, engine='openpyxl')
        else:
            print("Error: Unsupported file type. Please provide a CSV or Excel␣
 ↪file.")
            sys.exit(1)

        #print(f"\nData loaded successfully with {len(df)} rows and {len(df.
 ↪columns)} columns.")
        return df

    except Exception as e:
        print(f"Error loading file: {str(e)}")
        sys.exit(1)

def preprocess_data(df):
    """Clean and prepare the data for modeling."""
    if 'Priority_Label' not in df.columns:
        print("Error: Required 'Priority_Label' column not found.")
        sys.exit(1)

    # Clean and validate priority labels
    df['Priority_Label'] = df['Priority_Label'].astype(str).str.strip()
    valid_labels = ['High', 'Medium', 'Low', ' High', ' Medium', ' Low']
    df = df[df['Priority_Label'].isin(valid_labels)]

    if len(df) == 0:
        print("Error: No valid priority labels found after cleaning.")
        sys.exit(1)

    # Define feature types
    FEATURE_DEFINITIONS = {
        'Court_Name': 'categorical',
        'Case_Type': 'categorical',
        'Urgency_Tag': 'categorical',
        'Advocate_Names': 'categorical',
        'Legal_Sections': 'categorical',
        'Past_History': 'categorical',
        'Estimated_Impact': 'categorical',
        'Media_Coverage': 'categorical',
        'Days_to_Resolution': 'numerical'
    }

    # Select available features
```

```
    available_features = set(df.columns) - {'Priority_Label'}
    categorical_cols = [col for col in available_features
                        if FEATURE_DEFINITIONS.get(col, 'categorical') ==␣
 ↪'categorical']
    numerical_cols = [col for col in available_features
                      if FEATURE_DEFINITIONS.get(col, 'categorical') ==␣
 ↪'numerical']

    return df, categorical_cols, numerical_cols

def train_models(X_train, y_train, categorical_cols, numerical_cols):
    """Train and return both logistic regression and random forest models."""
    # Preprocessing pipeline
    numeric_transformer = Pipeline([
        ('imputer', SimpleImputer(strategy='mean')),
        ('scaler', StandardScaler())
    ])

    categorical_transformer = Pipeline([
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('onehot', OneHotEncoder(handle_unknown='ignore'))
    ])

    preprocessor = ColumnTransformer([
        ('num', numeric_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

    # Model pipelines
    lr_pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', LogisticRegression(
            max_iter=1000,
            random_state=42,
            class_weight='balanced',
            multi_class='multinomial'
        ))
    ])

    rf_pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(
            n_estimators=100,
            random_state=42,
            class_weight='balanced'
        ))
    ])
```

```
    #print("\nTraining models...")
    lr_pipeline.fit(X_train, y_train)
    rf_pipeline.fit(X_train, y_train)
    #print("Training completed successfully.")

    return lr_pipeline, rf_pipeline

def evaluate_models(models, X_test, y_test):
    """Evaluate model performance on test data."""
    #print("\nModel Evaluation Results:")
    for name, model in models.items():
        y_pred = model.predict(X_test)
        #print(f"\n{name} Performance:")
        #print(classification_report(y_test, y_pred))
        #print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")

def predict_and_rank(models, X, original_df, n_cases=None):
    """Generate predictions and rank cases by priority."""
    # Get predictions from both models
    lr_proba = models['Logistic Regression'].predict_proba(X)
    rf_proba = models['Random Forest'].predict_proba(X)

    # Average probabilities
    avg_proba = (lr_proba + rf_proba) / 2
    priority_scores = avg_proba.max(axis=1)
    predicted_labels = avg_proba.argmax(axis=1)

    # Create results dataframe
    results = original_df.copy()
    results['Priority_Score'] = priority_scores
    results['Predicted_Priority'] = predicted_labels

    # Sort by priority score
    results = results.sort_values('Priority_Score', ascending=False)

    return results.head(n_cases) if n_cases else results

def get_user_selection():
    """Prompt user for number of cases to display."""
    print("\nSelect number of cases to display:")
    print("1. Top 10 highest priority cases")
    print("2. Top 50 highest priority cases")
    print("3. Top 100 highest priority cases")
    print("4. All cases (ranked by priority)")

    while True:
```

```
        choice = input("Enter your choice (1-4): ")
        if choice in ['1', '2', '3', '4']:
            return {
                '1': 10,
                '2': 50,
                '3': 100,
                '4': None
            }[choice]
        print("Invalid input. Please enter 1-4.")

# ====================== MAIN EXECUTION ======================
def main():
    # Load and prepare data
    #print("==== Legal Case Priority Classifier ====")
    df = load_data()
    df, categorical_cols, numerical_cols = preprocess_data(df)

    # Prepare features and target
    X = df.drop(columns=['Priority_Label'])
    y = df['Priority_Label']

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y)

    # Train models
    lr_model, rf_model = train_models(
        X_train, y_train, categorical_cols, numerical_cols)

    # Evaluate models
    evaluate_models({
        'Logistic Regression': lr_model,
        'Random Forest': rf_model
    }, X_test, y_test)

    # Generate predictions
    n_cases = get_user_selection()
    ranked_cases = predict_and_rank(
        {'Logistic Regression': lr_model, 'Random Forest': rf_model},
        X, df, n_cases)

    # Display results
    print("\n" + "="*50)
    print(f"\n{'All' if n_cases is None else f'Top {n_cases}'} Priority Cases:")
    display_cols = ['Priority_Score', 'Predicted_Priority'] + categorical_cols␣
 ↪+ numerical_cols
    display_cols = [col for col in display_cols if col in ranked_cases.columns]
```

```
    print(ranked_cases[display_cols].to_string())

    # Save results option
    if input("\nSave results to CSV? (y/n): ").lower() == 'y':
        output_path = input("Enter output filename (e.g., results.csv): ").
 ↪strip()
        ranked_cases.to_csv(output_path, index=False)
        print(f"Results saved to {output_path}")

if __name__ == "__main__":
    main()
```

```
Enter path to data file (CSV or Excel): "C:\Users\dell\Downloads\Editable
Sheet.xlsx"

Select number of cases to display:
1. Top 10 highest priority cases
2. Top 50 highest priority cases
3. Top 100 highest priority cases
4. All cases (ranked by priority)
Enter your choice (1-4): 1

==================================================

Top 10 Priority Cases:
    Priority_Score  Predicted_Priority Parties_Involved      Court_Name
Legal_Sections                          Case_Summary Case_Status Past_History
Urgency_Tag        Advocate_Names Filing_Date Media_Coverage Estimated_Impact
Hearing_Date Case_Type Case_ID
46        0.882743                   2     Company vs Z   Supreme Court  Family
Act 1955  Brief description of the legal case.    Resolved          Yes
Regular    L. Verma, A. Singh  2023-10-18            Yes              Low
2024-03-04     Civil  C10046
33        0.879301                   2     Company vs Z   Supreme Court
Article 32  Brief description of the legal case.    Resolved          Yes
Regular   L. Verma, K. Sharma  2023-09-27             No             High
2024-09-20     Civil  C10033
2         0.869443                   2     Company vs Z   Supreme Court
CRPC 125  Brief description of the legal case.    Resolved          Yes
Regular    L. Verma, R. Mehta  2023-08-28            Yes           Medium
2024-07-24     Civil  C10002
36        0.856286                   2     Company vs X      High Court  Family
Act 1955  Brief description of the legal case.     Pending          Yes
Regular   K. Sharma, R. Mehta  2023-03-22            Yes              Low
2023-07-17       PIL  C10036
23        0.841950                   2        Govt vs Y      High Court
Article 32  Brief description of the legal case.     Pending           No
Regular    A. Singh, A. Singh  2023-02-20            Yes              Low
```

```
2023-08-18     Civil  C10023
9         0.827224                   2     Company vs X      High Court
Article 32  Brief description of the legal case.      Closed          Yes
Regular    R. Mehta, R. Mehta  2023-04-14             No              Low
2023-05-24     Civil  C10009
38        0.818429                   1        Govt vs X  District Court
IPC 302  Brief description of the legal case.     Pending          Yes
Regular    A. Singh, R. Mehta  2023-11-01             No             High
2024-01-10    Family  C10038
21        0.816040                   1        Govt vs Y   Supreme Court     IPC
420, 467  Brief description of the legal case.     Pending          Yes
Regular  K. Sharma, K. Sharma  2023-04-21             No             High
2023-10-27    Family  C10021
28        0.811968                   0     Company vs Y  District Court  Family
Act 1955  Brief description of the legal case.    Resolved          No
Emergency   R. Mehta, K. Sharma  2023-01-11            Yes              Low
2023-12-18    Family  C10028
4         0.808517                   0  Individual vs Y      High Court
IPC 302  Brief description of the legal case.      Closed          No
Regular    L. Verma, R. Mehta  2023-11-04            Yes           Medium
2024-01-24       PIL  C10004

Save results to CSV? (y/n): n
```

[ ]: