

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
from sklearn.model_selection import train_test_split
# decision tree visualization
from sklearn.tree import export_graphviz
import six
import sys
sys.modules['sklearn.externals.six'] = six
from six import StringIO
from IPython.display import Image
import pydotplus
```

```
df = pd.read_csv('https://raw.githubusercontent.com/rlevy820/scotus-research/refs/heads/main/data/Case%20Centered%20
df = df[df['partyWinning'] != 2]
df.head()
```

	caseId	docketId	caseIssuesId	voteId	dateDecision	decisionType	usCite	sctCite	ledCite	lexisCite	...	authori
0	1946-001	1946-001-01	1946-001-01-01	1946-001-01-01	11/18/1946	1	329 U.S. 1	67 S. Ct. 6	91 L. Ed. 3	1946 U.S. LEXIS 1724	...	
1	1946-002	1946-002-01	1946-002-01-01	1946-002-01-01	11/18/1946	1	329 U.S. 14	67 S. Ct. 13	91 L. Ed. 12	1946 U.S. LEXIS 1725	...	
2	1946-002	1946-002-02	1946-002-02-01	1946-002-02-01-01	11/18/1946	1	329 U.S. 14	67 S. Ct. 13	91 L. Ed. 12	1946 U.S. LEXIS 1725	...	
3	1946-002	1946-002-03	1946-002-03-01	1946-002-03-01-01	11/18/1946	1	329 U.S. 14	67 S. Ct. 13	91 L. Ed. 12	1946 U.S. LEXIS 1725	...	
4	1946-002	1946-002-04	1946-002-04-01	1946-002-04-01-01	11/18/1946	1	329 U.S. 14	67 S. Ct. 13	91 L. Ed. 12	1946 U.S. LEXIS 1725	...	

5 rows × 53 columns

```
data1 = df[['term', 'caseOrigin', 'issueArea', 'petitioner', 'respondent', 'lcDispositionDirection', 'declarationUnc
data2 = data1.dropna()
```

```
inputs = data2[['term', 'caseOrigin', 'issueArea', 'petitioner', 'respondent', 'lcDispositionDirection', 'declaratio
target = data2['partyWinning']
```

```
X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.2)
```

```
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)
```

▼ DecisionTreeClassifier ⓘ ?

```
DecisionTreeClassifier()
```

```
model.score(X_test, y_test)
```

0.749811320754717

prompt: create a UI that allows one to select input features and it gives the models prediction, make them dropdown:

```
import ipywidgets as widgets
from IPython.display import display, clear_output
import warnings
warnings.filterwarnings('ignore')
```

```
# Create dropdown widgets for each input feature
term_widget = widgets.Dropdown(options=sorted(df['term'].unique()), description='Term:')
caseOrigin_widget = widgets.Dropdown(options=sorted(df['caseOrigin'].unique()), description='Case Origin:')
issueArea_widget = widgets.Dropdown(options=sorted(df['issueArea'].unique()), description='Issue Area:')
petitioner_widget = widgets.Dropdown(options=sorted(df['petitioner'].unique()), description='Petitioner:')
respondent_widget = widgets.Dropdown(options=sorted(df['respondent'].unique()), description='Respondent:')
lcDispositionDirection_widget = widgets.Dropdown(options=sorted(df['lcDispositionDirection'].unique()), description='LC Disposition Direction:')
declarationUnc_widget = widgets.Dropdown(options=sorted(df['declarationUncertainty'].unique()), description='Declaration Uncertainty:')
partyWinning_widget = widgets.Dropdown(options=sorted(df['partyWinning'].unique()), description='Party Winning')
```

```

issueArea_widget = widgets.Dropdown(options=sorted(df['issueArea'].unique()), description='Issue Area:')
petitioner_widget = widgets.Dropdown(options=sorted(df['petitioner'].unique()), description='Petitioner:')
respondent_widget = widgets.Dropdown(options=sorted(df['respondent'].unique()), description='Respondent:')
lcDispositionDirection_widget = widgets.Dropdown(options=sorted(df['lcDispositionDirection'].unique()), description='lcDispositionDirection')
declarationUncon_widget = widgets.Dropdown(options=sorted(df['declarationUncon'].unique()), description='Declaration Uncon')

# Create a button to trigger prediction
predict_button = widgets.Button(description='Predict')

# Create an output widget to display the prediction
output_widget = widgets.Output()

# Function to handle button click event
def on_predict_button_clicked(b):
    with output_widget:
        clear_output()
        input_data = {
            'term': term_widget.value,
            'caseOrigin': caseOrigin_widget.value,
            'issueArea': issueArea_widget.value,
            'petitioner': petitioner_widget.value,
            'respondent': respondent_widget.value,
            'lcDispositionDirection': lcDispositionDirection_widget.value,
            'declarationUncon': declarationUncon_widget.value
        }

        # Convert input data to a DataFrame
        input_df = pd.DataFrame([input_data])

        # Make prediction using the trained model
        prediction = model.predict(input_df)[0]

        # Display the prediction
        if prediction == 0:
            print("Denied")
        elif prediction == 1:
            print("Affirmed")

# Assign the function to the button's on_click event
predict_button.on_click(on_predict_button_clicked)

# Display the widgets
display(term_widget, caseOrigin_widget, issueArea_widget, petitioner_widget, respondent_widget,
        lcDispositionDirection_widget, declarationUncon_widget, predict_button, output_widget)

```

Term:	1954
Case Origin:	15.0
Issue Area:	1.0
Petitioner:	9.0
Respondent:	1.0
LC Disposit...	2.0
Declaration...	1.0

Predict

Affirmed

