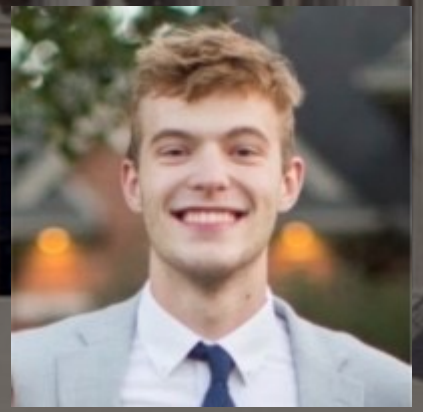
Supreme Court Judgement Predictions

Hailey Naugle



Brandon Owens

The Data

The Preparation

EDA

The Models

The Data

The Preparation

EDA

The Models

Data Sets

Supreme Court Cases

Supreme Court Judges

US Presidents

Supreme Court Cases

	name	term	first_party	second_party	facts	decision_type	disposition	issue_area	first_party_winner
0	Roe v. Wade	1971	Jane Roe	Henry Wade	<p>In 1970, Jane Roe (a fictional name used in...	majority opinion	reversed	NaN	True
1	Stanley v. Illinois	1971	Peter Stanley, Sr.	Illinois	<p>Joan Stanley had three children with Peter ...	majority opinion	reversed/remanded	Civil Rights	True
2	Giglio v. United States	1971	John Giglio	United States	<p>John Giglio was convicted of passing forged...	majority opinion	reversed/remanded	Due Process	True
3	Reed v. Reed	1971	Sally Reed	Cecil Reed	<p>The Idaho Probate Code specified that "male...	majority opinion	reversed/remanded	Civil Rights	True
4	Miller v. California	1971	Marvin Miller	California	<p>Miller, after conducting a mass mailing cam...	majority opinion	vacated/remanded	First Amendment	True
...	...	...	...	...	...	...	...	...	...

Supreme Court Judges

Index	Justice Name	Supreme Court Term Start	Supreme Court Term End	Appointing President
1	Jackson, Ketanji Brown(Associate Justice)	June 30, 2022	--	Biden, Joseph R.
2	Coney Barrett, Amy(Associate Justice)	October 27, 2020	--	Trump, Donald J.
3	Kavanaugh, Brett M.(Associate Justice)	October 6, 2017	--	Trump, Donald J.
4	Gorsuch, Neil M.(Associate Justice)	April 10, 2017	--	Trump, Donald J.
5	Kagan, Elena(Associate Justice)	August 7, 2010	--	Obama, Barack H.
...	...	...	...	...
117	Rutledge, John(Associate Justice)	February 15, 1790	March 5, 1791	Washington, George
118	Cushing, William(Associate Justice)	February 2, 1790	September 13, 1810	Washington, George
119	Blair, John(Associate Justice)	February 2, 1790	October 25, 1795	Washington, George
120	Jay, John(Chief Justice)	October 19, 1789	June 29, 1795	Washington, George
121	Wilson, James(Associate Justice)	October 5, 1789	August 21, 1798	Washington, George

US Presidents

President	Party
George Washington	Independent
John Adams	Federalist
Thomas Jefferson	Democratic-Republican
James Madison	Democratic-Republican
James Monroe	Democratic-Republican
John Quincy Adams	Democratic-Republican/National Republican
Andrew Jackson	Democratic
Martin Van Buren	Democratic
William Henry Harrison	Whig

Party Affiliation Assumption

117	Rutledge, John(Associate Justice)	February 15, 1790	March 5, 1791	Washington, George

+

George Washington	Independent

= John Rutledge associated as an "Independent"

The Data

The Preparation

EDA

The Models

Data Cleaning

Get All Dates in Terms of Years

```
judges['start_year'] = pd.to_datetime(judges['Supreme Court Term Start']).dt.year
judges['end_year'] = pd.to_datetime(judges['Supreme Court Term End'].replace("--","01-Mar-24")).dt.year
```

Match Judges to their Party Affiliations

```
judges['president_last_name'] = judges["Appointing President"].str.extract(r'^(\w+),')
presidents['president_last_name']= presidents['President '].str.extract(r'\s(\w+)$')
```

```
president_party_dict = presidents.set_index('president_last_name')['Party '].to_dict()
judges['party'] = np.where(judges['president_last_name'].isin(president_party_dict.keys()),
                           judges['president_last_name'].map(president_party_dict),
                           None)
judges['party'] = judges['party'].dropna().apply(lambda x: x.strip())
```

`Democratic` + `Republican` != `Democratic-Republican`

Match Cases to Judges to Get Counts for Each Ideology

```
cases["conservative"] = 0
cases["liberal"] = 0
cases["neutral"] = 0
```

```
for case_index, term in enumerate(cases.term):
    for judge_index, (start, end) in enumerate(zip(judges.start_year, judges.end_year)):
        if (start <= term <= end):
            judge_ideology = judges.ideology[judge_index]
            cases.loc[case_index, judge_ideology] += 1
        else:
            pass
```

Remove Rows Missing "`Issue_area` " and "`first_party_winner`"

```
cases = cases.loc[cases.issue_area.notnull()]
cases = cases.loc[cases.first_party_winner.notnull()]
```

Merged data

	name	first_party	second_party	facts	majority_vote	minority_vote	decision_type	first_party_winner	disposition	issue_area	conservative	liberal	neutral
1	Stanley v. Illinois	Peter Stanley, Sr.	Illinois	<p>Joan Stanley had three children with Peter ...	5	2	majority opinion	True	reversed/remanded	Civil Rights	5	4	0
2	Giglio v. United States	John Giglio	United States	<p>John Giglio was convicted of passing forged...	7	0	majority opinion	True	reversed/remanded	Due Process	5	4	0
3	Reed v. Reed	Sally Reed	Cecil Reed	<p>The Idaho Probate Code specified that "male...	7	0	majority opinion	True	reversed/remanded	Civil Rights	5	4	0
4	Miller v. California	Marvin Miller	California	<p>Miller, after conducting a mass mailing cam...	5	4	majority opinion	True	vacated/remanded	First Amendment	5	4	0

Natural Language Processing

Regularize Facts Statements

```
import nltk
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer


nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def tokenizer(text):
    ''' Tokenize document text by stripping the text into individual tokens, removing digit characters, removing punctuation, remove stop words, and lemmatizing the tokens.

    Parameters
    ----------
    text : str
        The body of text that you would like to tokenize

    Returns
    -------
    cleaned_doc_tokens : str
        The cleaned and tokenized text

    Example
    -------
    text = "Hailey Naugle was quite the important contributor to this Supreme Court analysis. - Brandon Owens"

    clean_text = tokenizer(text)

    Will output:
        "hailey naugle important contributor supreme court analysis brandon owens"

    '''

    text_tokens = []
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        sent_tokens = nltk.word_tokenize(sentence)
        sent_tokens = [lemmatizer.lemmatize(word.lower()) for word in sent_tokens
                       if (word.lower() not in stop_words) and (word not in string.punctuation) and (len(word) > 1) and not(word.isdigit())]
        text_tokens += sent_tokens

    cleaned_doc_tokens = ' '.join(text_tokens)

    return cleaned_doc_tokens
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

count_vectorizer = CountVectorizer()
```

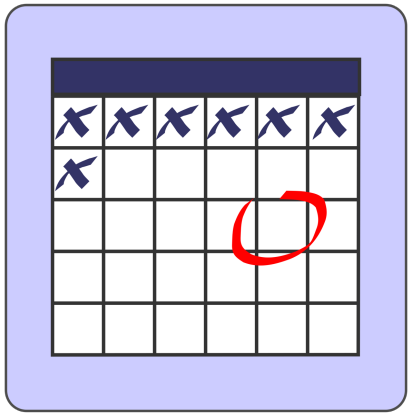
```
counts_vectorized = count_vectorizer.fit_transform(cases["cleaned_facts"])
counts_array = counts_vectorized.toarray()
counts_df = pd.DataFrame(counts_array, columns = count_vectorizer.vocabulary_.keys())
```

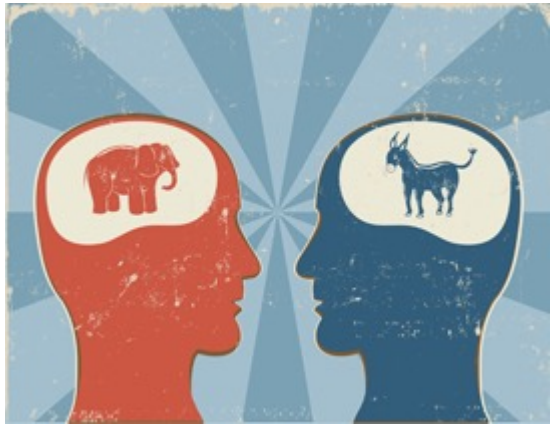
Ex:

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Features

Non-NLP:





NLP:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

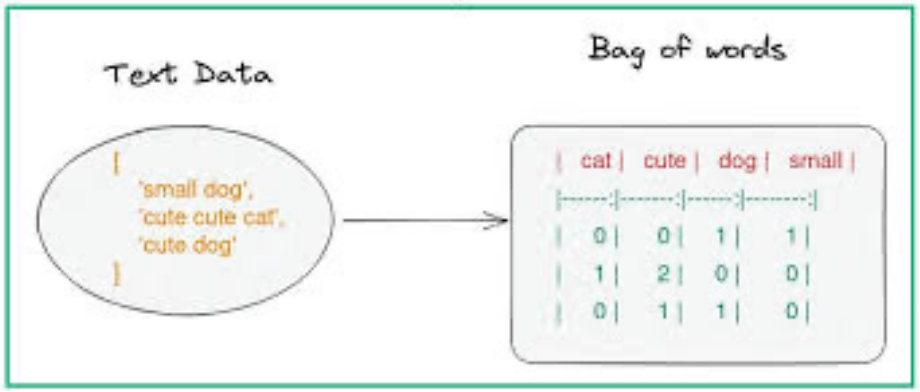
TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents



The Data

The Preparation

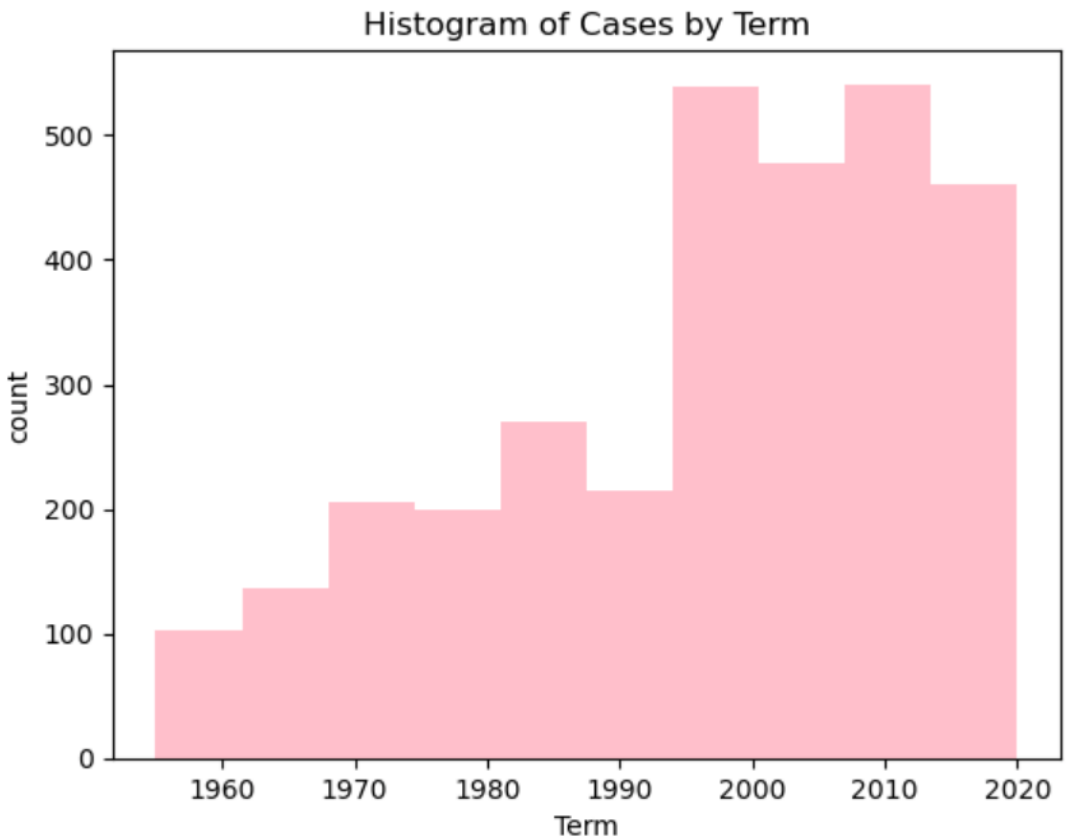
EDA

The Models

Cases By Term

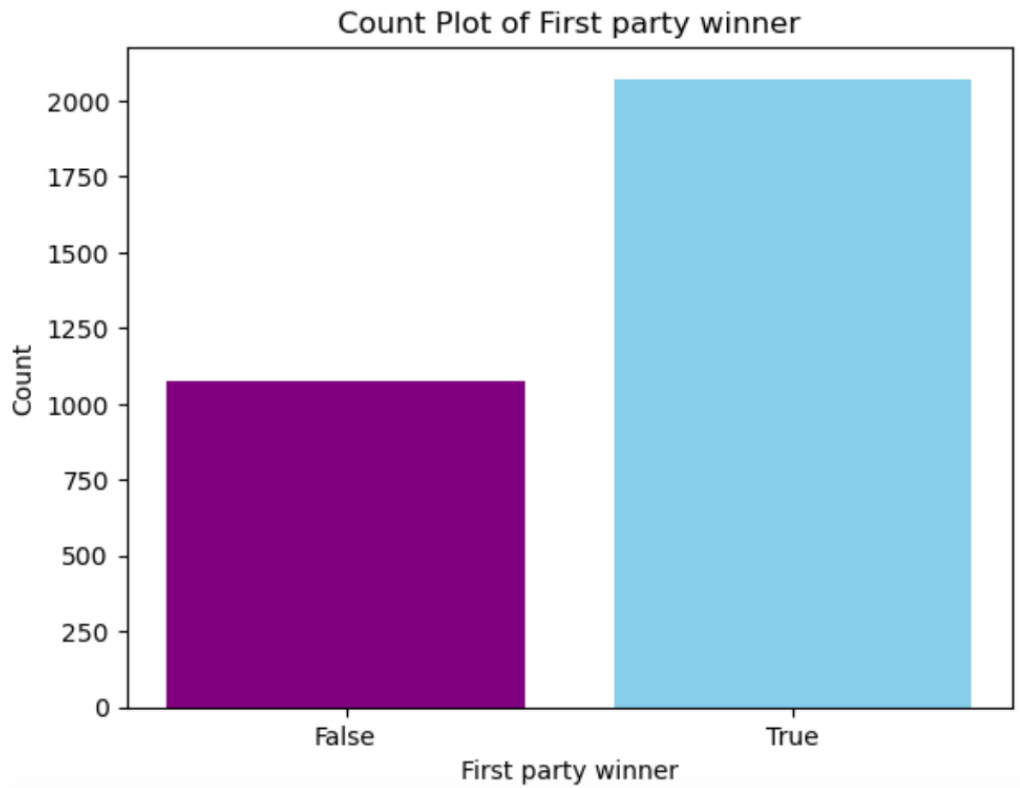
```
ax = cases.feat_term.hist(color = 'pink', grid = False)
ax.set_title('Histogram of Cases by Term')
ax.set_xlabel('Term')
ax.set_ylabel('count')
```

```
Text(0, 0.5, 'count')
```

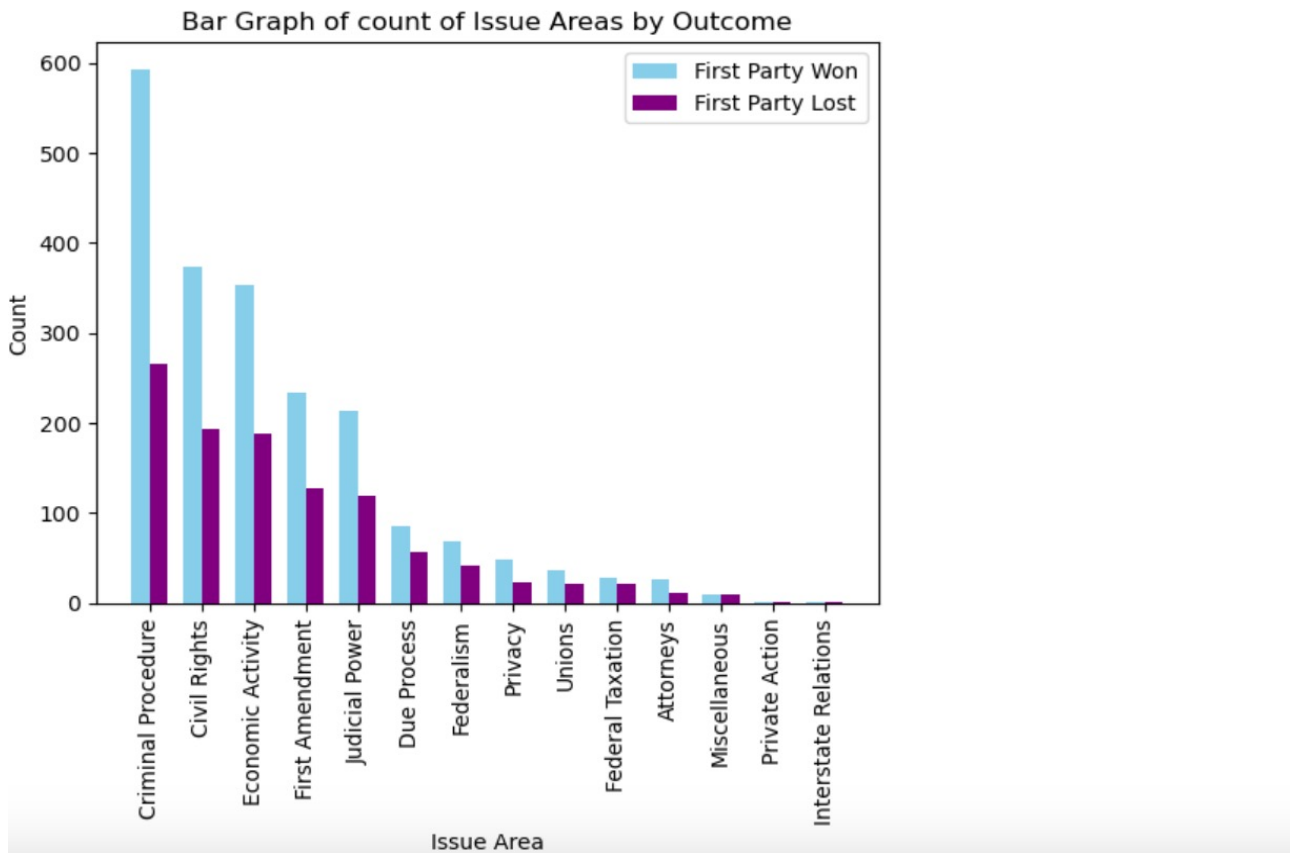


```
winner_counts = cases.first_party_winner.value_counts()


plt.figure()
plt.bar(winner_counts.index, winner_counts, color = ['skyblue', 'purple'])
plt.title('Count Plot of First party winner')
plt.xlabel('First party winner')
plt.ylabel('Count')
plt.xticks( [0,1], [False, True])
```

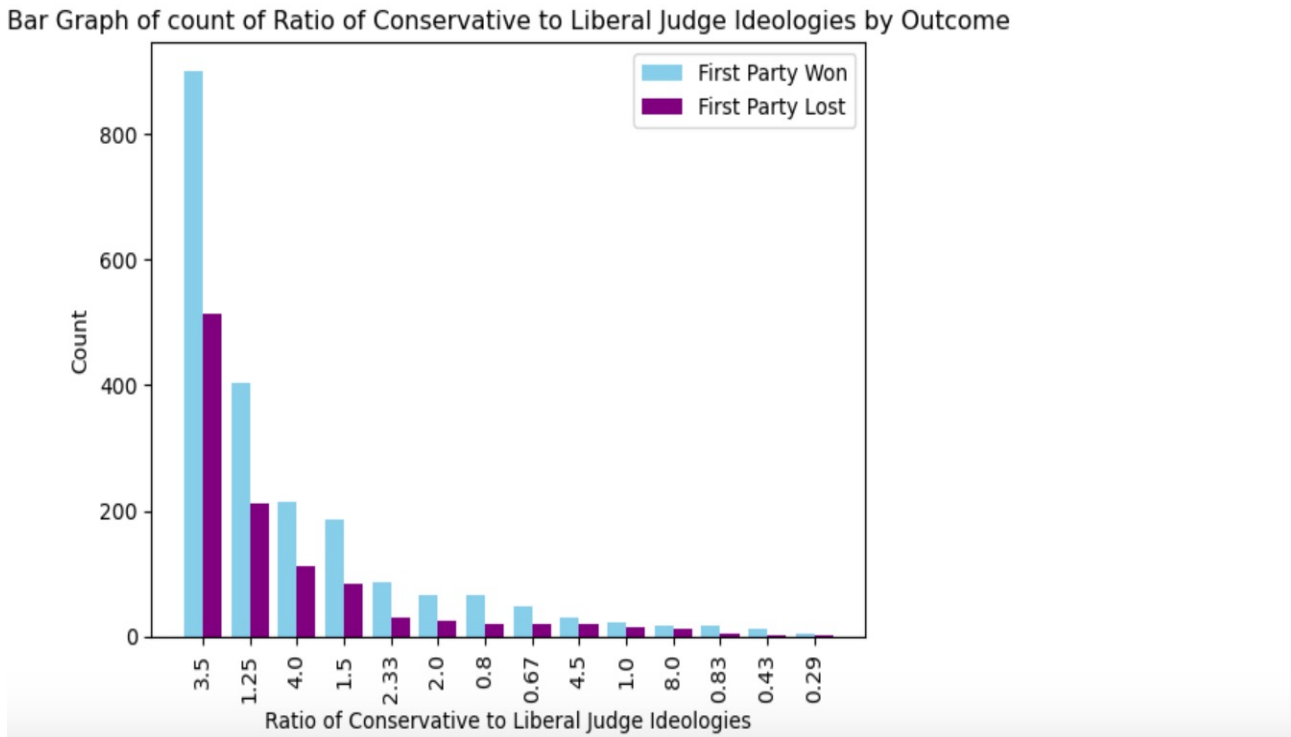
```
([<matplotlib.axis.XTick at 0x123dd9b5120>,
  <matplotlib.axis.XTick at 0x123dd9b50f0>],
 [Text(0, 0, 'False'), Text(1, 0, 'True')])
```



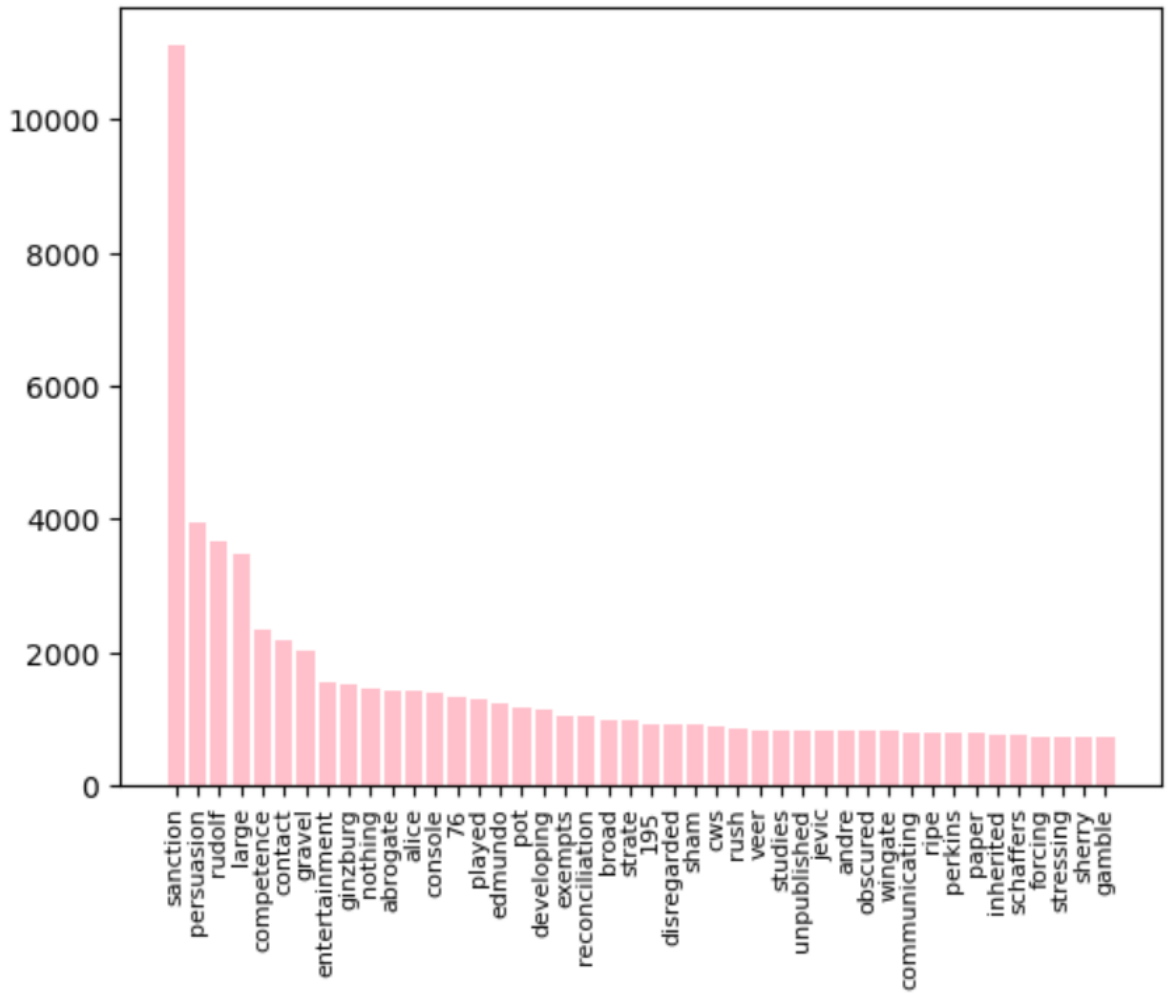
```
issue_counts = cases2.issue_area.value_counts()
issue_true = cases2[cases2.first_party_winner == True].issue_area.value_counts()
issue_false = cases2[cases2.first_party_winner == False].issue_area.value_counts()

bar_width = 0.35
bar_p1 = np.arange(len(issue_counts.index))
bar_p2 = bar_p1 + bar_width

plt.bar(bar_p1, issue_true, width=bar_width, label='First Party Won', color='skyblue')
plt.bar(bar_p2, issue_false, width=bar_width, label='First Party Lost', color='purple')
plt.xlabel('Issue Area')
plt.ylabel('Count')
plt.xticks(bar_p1 + bar_width/2, issue_counts.index.tolist(), rotation = 90)
plt.title('Bar Graph of count of Issue Areas by Outcome')
plt.legend()
plt.show()
```



```
ideology_counts = cases.ratio_conservative_to_liberal.value_counts()
ideology_true = cases[cases.first_party_winner == True].ratio_conservative_to_liberal.value_counts()
ideology_false = cases[cases.first_party_winner == False].ratio_conservative_to_liberal.value_counts()

bar_width = 0.4
bar_p1 = np.arange(len(ideology_counts.index))
bar_p2 = bar_p1 + bar_width
loc = bar_p1 + bar_width/2
rounded_ideology_ratios = [round(x,2) for x in ideology_counts.index.tolist()]


plt.bar(bar_p1, ideology_true, width=bar_width, label='First Party Won', color='skyblue')
plt.bar(bar_p2, ideology_false, width=bar_width, label='First Party Lost', color='purple')
plt.xlabel('Ratio of Conservative to Liberal Judge Ideologies')
plt.ylabel('Count')
plt.xticks(loc, rounded_ideology_ratios, rotation = 90)
plt.title('Bar Graph of count of Ratio of Conservative to Liberal Judge Ideologies by Outcome')
plt.legend()
plt.show()
```



```
sums = counts_df.sum()
sums.sort_values(ascending=False, inplace=True)

fig, ax = plt.subplots()
ax.bar(sums.index[sums > 700],sums[sums >700], color="pink")
ax.xaxis.set_tick_params(rotation=90, labelsize = 8)
```



The Data

The Preparation

EDA

The Models

Baseline

What do these disposition terms mean exactly?

- Reversed/Remanded: The Supreme Court disagrees with the decision of the lower court and sent it back for further consideration.
- Affirmed: The Supreme Court agrees and upholds the lower court's decision.
- Vacated/Remanded: The Supreme Court nullified the decision of the lower court and sent it back for further consideration.
- Reversed-in-Part: The Supreme Court partially disagreed with the decision of the lower court but did not send the case back for further consideration.
- Vacated: The Supreme Court nullified the decision of the lower court without sending it back for further consideration.
- Vacated-in-Part/Remanded: The Supreme Court nullified part of the decision of the lower court and sent it back for further consideration.

Comparison of all the models

	No-NLP	Only NLP		Features + NLP	
		Counts	TFIDF	Counts	TFIDF
Logistic Regression	**0.667**	0.578	0.667	0.657	0.638
Decision Tree	0.594	0.573	0.554	**0.658**	0.655
Random Forest	0.595	**0.668**	0.657		
KNN	0.606	0.635	0.560		
XGB				0.657	0.657
Baseline: 0.640					


"That's all Folks!"