Barcelona School of Economics

Classifying Political Parties Ideological Positions (Supervised Learning)

Hannes Schiemann | Simon Vellin | Ferran Boada

26 March, 2025

Overview

This paper provides a basic supervised learning approach to classify European political parties' ideological positions based on their respective manifesto and ideological scores extracted from the Chapel Hill Expert Survey (CHES).

- **Model Input**: Analyze manifesto texts (translated into English) for each election year, matching them to CHES ideological scores while clustering them into groups (left, center, right).
- **Methodology**: Reduce text spasticity through preprocessing, apply TF-IDF vectorization and feature selection (weighting), and finally train different models (Regression, Random Forest, LightGBM).
- **Results**: Achieved up to 85% accuracy with LightGBM for binary classification (Left vs. Right).

Contents

1 Dataset Overview **2**
1.1 The Manifesto Project . . . 2
1.2 Chapel Hill Expert Survey (CHES) . . . 3

2 Data Extraction **4**
2.1 Setup and Environment . . . 4
2.2 Original English Manifestos . . . 4
2.3 Translated Manifestos . . . 4
2.4 Final Cleaning and Structuring . . . 4

3 Preprocessing Functions **5**
3.1 Lowercasing and Stopwords Removal . . . 5
3.2 Stemming . . . 5
3.3 Lemmatization . . . 5
3.4 Application to Dataset . . . 6
3.5 Text_lemma, our final preprocessing choice . . . 6

4 Merge Manifestos with "lrgen" **6**

5 Predicting "lrgen" Exact Value **7**
5.1 Naive Models . . . 7
5.2 Moving Forward . . . 7

6 Classification **8**
6.1 Model Training and Hyperparameter Tuning . . . 8
6.2 Model Evaluation . . . 8
6.2.1 Two Groups - Random Forest vs Light GBM . . . 8
6.2.2 Three Groups - Random Forest vs Light GBM . . . 9
6.2.3 Feature Importance . . . 10

7 Conclusion **11**
7.1 Model Limitations . . . 11
7.2 Final Application . . . 11

8 Bibliography **12**

Part 1: Purpose

In this section, we describe the datasets used in our study, and display the frequency distribution of some of the keys columns.
We have narrowed the analysis to European countries to reduce the complexity of dealing with extremely different cultures. Indeed, to improve the chances of finding meaningful linguistic patterns in the manifestos, we need countries facing similar political themes.

1 Dataset Overview

1.1 The Manifesto Project

The Manifesto Project *(https://manifesto-project.wzb.eu)* collects and provides access to party manifestos from numerous countries and elections (translated in english for most elections). It is widely used in political science research to analyze party strategies, ideological shifts, and policy emphases over time.
For our study, we filter the database for European countries and use manifestos as a basis for linguistic patterns.

The data displays enough variation to justify a machine learning approach as shown in Table 1.

Column	Number of Distinct Values
text	1268
country	32
party name	476

Table 1: Label distribution in the manifesto dataset

In terms of countries represented, we might encounter some challenges related to country-specific features given the variety of political setup represented in the dataset.

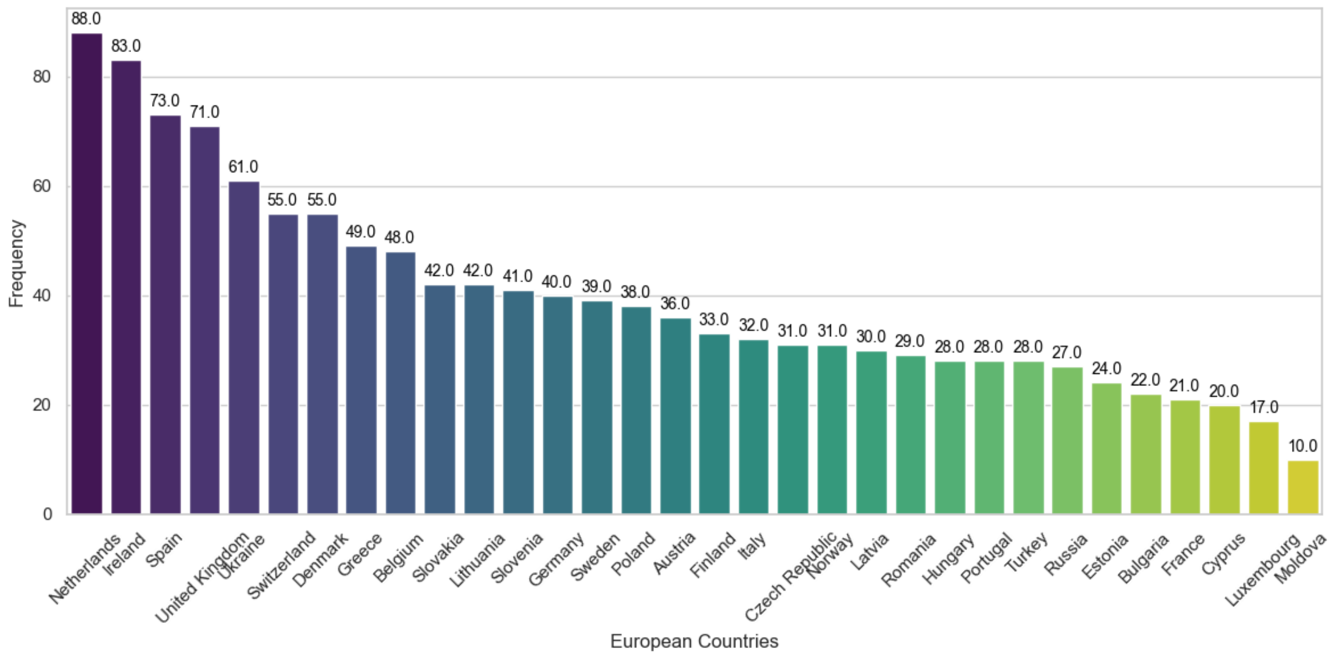
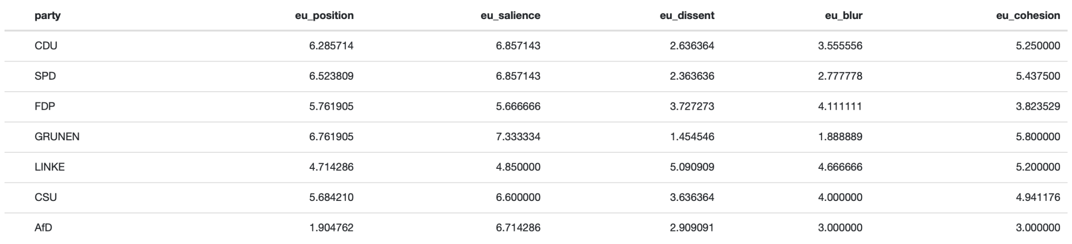


Figure 1: Distribution of European Countries within "Country Name" column

1.2 Chapel Hill Expert Survey (CHES)

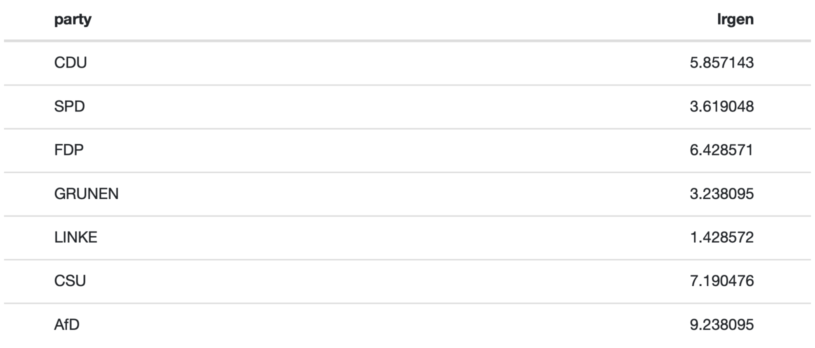
This survey *(https://www.chesdata.eu)* compiles expert evaluations of each political party in various European countries regarding their stance on various topics, such as *"Overall orientation of the party leadership towards European integration in 2019"* as shown for Germany in Figure 2 below.



party	eu_position	eu_salience	eu_dissent	eu_blur	eu_cohesion
CDU	6.285714	6.857143	2.636364	3.555556	5.250000
SPD	6.523809	6.857143	2.363636	2.777778	5.437500
FDP	5.761905	5.666666	3.727273	4.111111	3.823529
GRUNEN	6.761905	7.333334	1.454546	1.888889	5.800000
LINKE	4.714286	4.850000	5.090909	4.666666	5.200000
CSU	5.684210	6.600000	3.636364	4.000000	4.941176
AfD	1.904762	6.714286	2.909091	3.000000	3.000000

Figure 2: German Example: mean estimates from the expert ratings (higher number = strong pro-European position)

These estimates are then used to compute a single measure of the party placement left–right axis, known as **"lrgen"**. It is widely used to map political parties on key issues at the regional level.



party	lrgen
CDU	5.857143
SPD	3.619048
FDP	6.428571
GRUNEN	3.238095
LINKE	1.428572
CSU	7.190476
AfD	9.238095

Figure 3: German Example: lrgen values in 2019

Overall, this second dataset is essential to categorize the parties along the left–right ideological spectrum, which enables subsequent steps in our study. Fortunately, the dataset is well-balanced in terms of *center v.s extremes*, as well as in terms of *left v.s right* (Figure 1).

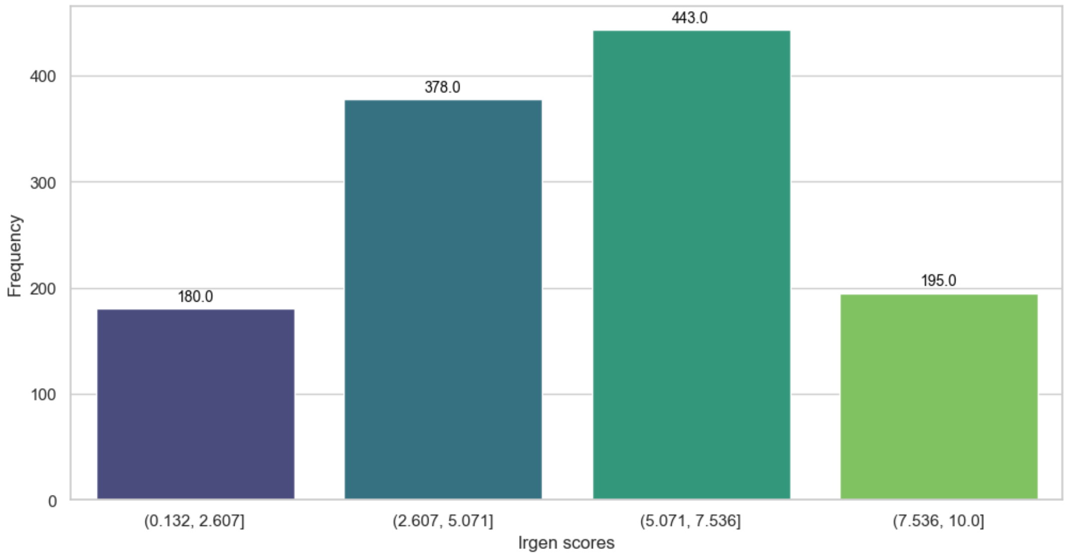


Figure 4: Distribution of "lrgen"

2 Data Extraction

The extraction of political manifestos from the Manifesto Project involved a multi-stage pipeline built using the `manifestoR` package in R, complemented by a final data cleaning step in Python. The objective was to retrieve both original English manifestos and their translated counterparts in a consistent format suitable for our later text analysis.

2.1 Setup and Environment

Before initiating the extraction, the API key provided by the Manifesto Project was stored as an environment variable. The `manifestoR` package was then used to authenticate and access the dataset.

2.2 Original English Manifestos

The extraction of original English manifestos followed a straightforward approach using the standard corpus function.

- The full manifesto metadata was loaded using `mp_maindataset()`.
- Documents in English were filtered using `mp_availability()`.
- The English texts were retrieved via `mp_corpus()` and looped over to extract their metadata (country, party, year) and full text.
- Texts were concatenated where necessary and saved to a CSV file named `english_manifestos.csv`.

2.3 Translated Manifestos

Retrieving the translated manifestos required a different method, using the bilingual corpus functionality.

- We used `mp_metadata()` to identify documents that were either originally in English or had an official English translation.
- The bilingual corpus was loaded using `mp_corpus_df_bilingual()`, with the English translation selected via the `translation = "en"` parameter.
- In this case the format of the data retrieved by the package method is different; many manifestos corpus were split so that each row corresponded to one line of the text - including metadata tags. Given that we were interested in a compact format for our project, we used parallel processing to concatenate text efficiently across all documents using the `future.apply` package.
- The resulting dataset included the year and party, with the country inferred from the party code.
- Manifestos were aggregated by party and year, combining multiple entries into unified documents for analytical consistency.
- The final output was saved as `translated_manifestos.csv`.

2.4 Final Cleaning and Structuring

After obtaining both the English and translated datasets, we used a Python Jupyter notebook section to clean and standardize the data:

- Duplicates and empty texts were removed.
- Consistent column naming and format alignment were applied.
- Final datasets were merged and saved individually, ready to be preprocessed for our project tasks.

Part 2: Data Wrangling

3 Preprocessing Functions

To ensure the manifestos are suitable for a machine learning model, we applied a series of preprocessing steps designed to standardize text representation while preserving meaningfulness. Our pipeline consists of the following transformations:

3.1 Lowercasing and Stopwords Removal

Each manifesto undergoes an initial normalization step to converts text to lowercase, removes punctuation and tokenizes words, and filter out stopwords. The below function does so:

```python
def preprocess_lower(text):
    # Lowercase the text
    text_lower = text.lower()

    # Remove punctuation (everything except word characters and whitespace)
    text_no_punct = re.sub(r'[^\w\s]', '', text_lower)

    # Tokenize the cleaned text
    tokens = word_tokenize(text_no_punct)
    return " ".join(tokens)
```

Listing 1: preprocess lower

3.2 Stemming

We now apply stemming using the Porter Stemmer on preprocessed text from previous function to reduce words to their root forms. This is done by stripping suffixes (e.g., *"running"* → *"run"*). The below function does so:

```python
def preprocess_stem(text):
    # Get the cleaned text from preprocess_lower
    text_clean = preprocess_lower(text)

    # Tokenize once
    tokens = word_tokenize(text_clean)

    # Initialize the Porter Stemmer
    porter = SnowballStemmer("english")

    # Stem each token
    stemmed_tokens = [porter.stem(token) for token in tokens]
    return " ".join(stemmed_tokens)
```

Listing 2: preprocess stem

3.3 Lemmatization

We use the `spaCy` library to map words to their dictionary base forms (e.g., *"better"* → *"good"*), while filtering out punctuation and stopwords. The below function does so:

```python
sp = spacy.load("en_core_web_sm")
def preprocess_lemma(text):
    # Process text using spaCy
    doc = sp(text)

    # Extract the lemma of each token, filtering out tokens flagged as punctuation or stopwords.
    lemmatized_tokens = [token.lemma_ for token in doc if not token.is_stop and not token.is_punct
and token.lemma_.strip() != '']
    return " ".join(lemmatized_tokens)
```

Listing 3: preprocess lemma

3.4 Application to Dataset

The preprocessing functions are applied sequentially to the dataset:

1. `text_no_stopwords`: Baseline cleaning.
2. `text_stem`: Stemming applied to `text_no_stopwords`.
3. `text_lemma`: Lemmatization applied to `text_no_stopwords`.

This approach allows flexibility in selecting the most effective text representation for further analysis.

3.5 Text_lemma, our final preprocessing choice

After trying all three approaches, we chose stop word and punctuation removal combined with lemmatization over stemming for our final results to ensure more accurate and meaningful linguistic analysis.

- Stop word and punctuation removal helps eliminate non-informative elements, allowing us to focus on the substantive content of the manifestos.
- Stemming, in contrast, often produces distorted or non-standard word forms that can obscure the intent behind the language used in manifestos (for example, we didn't want to reduce words like *organization* and *organ* to the same root).
- Lemmatization was preferred over stemming because it reduces words to their base or dictionary form while preserving their actual meaning, which is crucial when analyzing nuanced political language.

By using lemmatization, we maintained the semantic integrity of the text, resulting in cleaner and more interpretable data for downstream analysis.

4 Merge Manifestos with "lrgen"

Since manifestos are released for every election and scores are released every 4 years (as from 1999), we need to map the available "lrgen" score to the corresponding political party for each election. As such, we decided to match the 'earliest' score release after the manifesto publication.

```
# Ensure both DataFrames are sorted by year before using merge_asof
manifestos = manifestos.sort_values(by=['election_year', 'party'])
metadata = metadata.sort_values(by=['year', 'party'])
# Perform merge_asof
df = pd.merge_asof(
    left=manifestos,
    right=metadata,
    left_on='election_year',    # Sorting column in manifestos
    right_on='year',            # Sorting column in metadata
    by=['party'],               # Additional match condition
    direction='forward'         # Matches with the closest in forward direction
    )
```

The dataset now consist of 667 manifestos labeled with the corresponding "lrgen" score of the party at the time of publication.

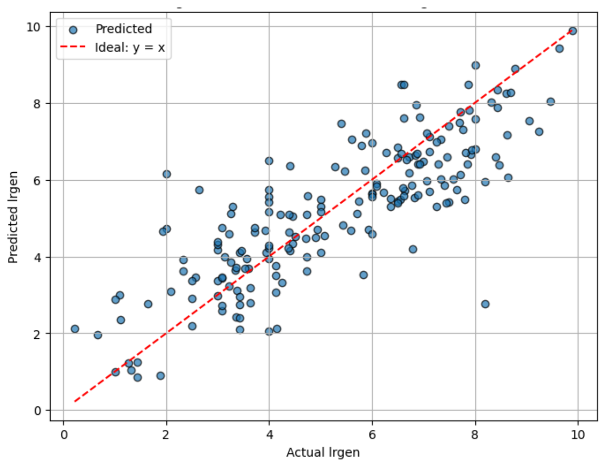
Part 3: Implementation

Finally, we can train a first model to predict the lrgen scores based on manifesto text by transforming manifesto text into a document term matrix using the Count Vectorizer, and fit a model using existing lrgen values on document vectors

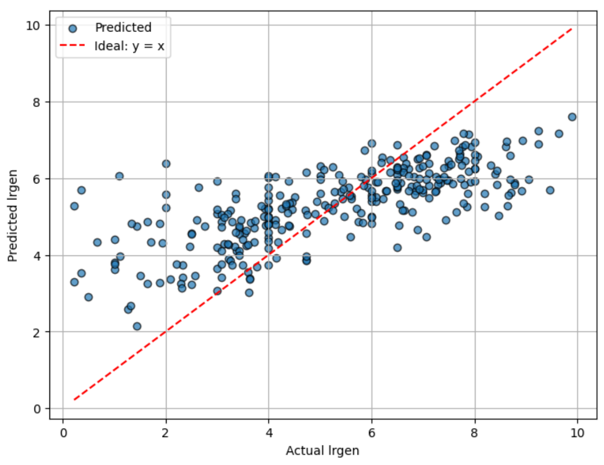
5 Predicting "lrgen" Exact Value

Our first attempt consists of directly training the model to predict "lrgen" as a continuous variable.

5.1 Naive Models



(a) Linear Regression with TF-IDF ($R^2 = 0.803$)



(b) Random Forest with TF-IDF ($R^2 = 0.467$)

Figure 5: Naive Models - predicting "lrgen" from "text_lemma"

5.2 Moving Forward

As expected, the random forest works poorly when predicting 'lrgen' as a continuous variable, since tree-based methods often struggle with high-dimensional inputs such as TF-IDF vectors, as partitioning the feature space to minimize variance becomes challenging with thousands of features.

Meanwhile, linear regression seems to do a decent job at predicting the 'lrgen' score based on the text tokens. An inspection of the most decisive features, i.e. the tokens with the biggest coefficients suggests that this is partly due to some overfitting to our sample (specific parties or countries) and that this would not translate to a broader sample of manifestos.

Feature	**Coefficient**
jobbik	4.391799
immigration	3.810833
reform	3.726814
french	3.448189
cypriot	3.442600

Table 2: Top Positive Features

Feature	**Coefficient**
public	-3.272438
enhedslisten	-3.425974
sf	-3.441506
lds	-3.842202
poverty	-3.916955

Table 3: Top Negative Features

6 Classification

Ultimately we decided to turn away from predicting the actual score to classifying parties into blocks using two different splits: Left | Right and Left | Center | Right.
This approach avoids the complexities and potential overfitting associated with predicting fine-grained continuous scores. In order to perform the classification we will train both a Random Forest and a Light GBM Model and compare their performance.

6.1 Model Training and Hyperparameter Tuning

To begin, we split our dataset randomly at a 20% cut-off to create an evaluation set leaving us with 541 observations to train on and 136 observations to evaluate the model performance. Next we train our classification models on the training set. The pipeline consists of three steps:

1. **Tokenization:** Use the CountVectorizer to split the text into tokens, where each token's numeric value corresponds to its frequency in the manifesto.
2. **TF-IDF Transformation:** Convert the raw token counts into weighted features using the TF-IDF Transformer, which emphasizes unique and informative words by combining term frequency and inverse document frequency.
3. **Model Training:** Pass these transformed features to classification models (such as Random Forest or LightGBM) to perform the final classification.

Additional to the model specific hyperparameter tuning we also explore whether using the TF-IDF Transformer improves performance and tune the parameters of the CountVectorizer. These include:

- ngram: controlling whether the text is split into bigrams (single-word tokens), bigrams (two-word tokens) or both
- min_df: controlling in how many of all manifestos the token must appear to be included as a feature
- max_df: controlling in how many it can appear at maximum

Especially the parameter min_df is crucial for our classification task. When setting it to zero every single word that appears even once in a single manifesto would be included as a feature. This would lead to an explosion of the feature space. In contrast, when setting the threshold too high we exclude too many features from which the model coulld potentially learn. Additionally, by setting this threshold appropriately we can filter out terms that are specific to certain manifestos or parties which prevents the models from overfitting as in the naive linear regression model.

6.2 Model Evaluation

In this section we present the hyperparameter tuned models and test their performance on the evaluation set.

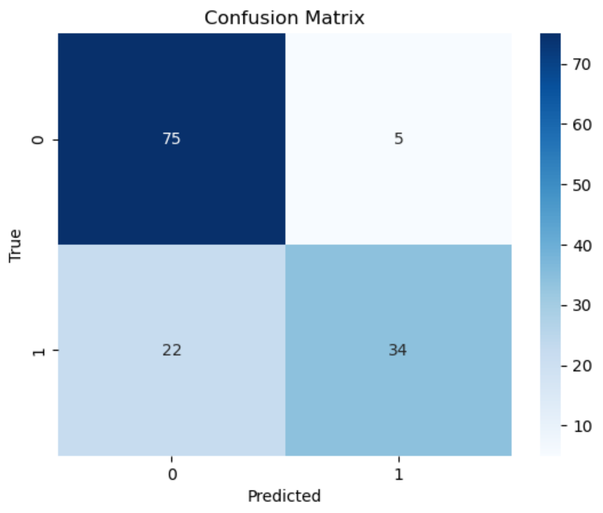
6.2.1 Two Groups - Random Forest vs Light GBM

Here we defined parties as "left" if their "lregen" score is below 0.5 and as "right" vice versa and trained the models based on these labels to classify unseen manifestos as "left" or "right".

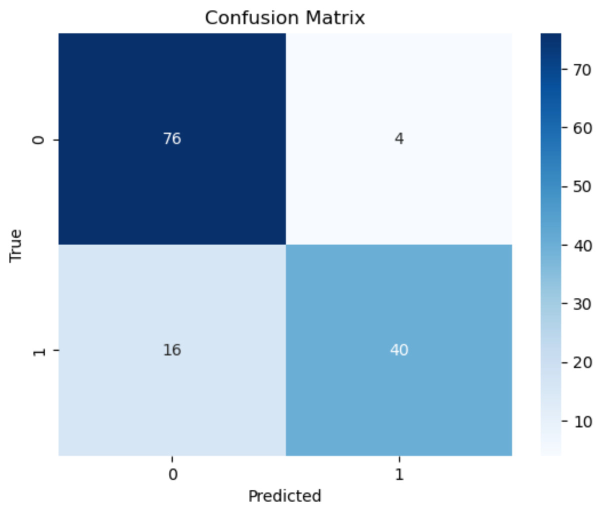
Parameter	**Random Forest**	**LightGBM**
`use_idf`	False	True
`max_df`	0.6	0.8
`min_df`	0.01	0.01
`ngram_range`	(1, 2)	(1, 2)

Table 4: Hyperparameter Settings for Random Forest and LightGBM

Below we present the Confusion Matrix from the classification on the test set. Both models are quite good at correctly classifying manifestos as "right", and a little less accurate when detecting "left". Here LightGBM outperforms Random Forest by detecting 40 of the 56 "right" labels vs 34 out of 22 by the Random Forest. The overall performance of LightGBM is also better with an accuracy of 0.85 compare to 0.8 by Random Forest.



(a) Random Forest with TF-IDF ($accuracy = 0.8$)



(b) Light GBM with TF-IDF ($accuracy = 0.85$)

Figure 6: Hyperparameter Models with 2-Split

6.2.2 Three Groups - Random Forest vs Light GBM

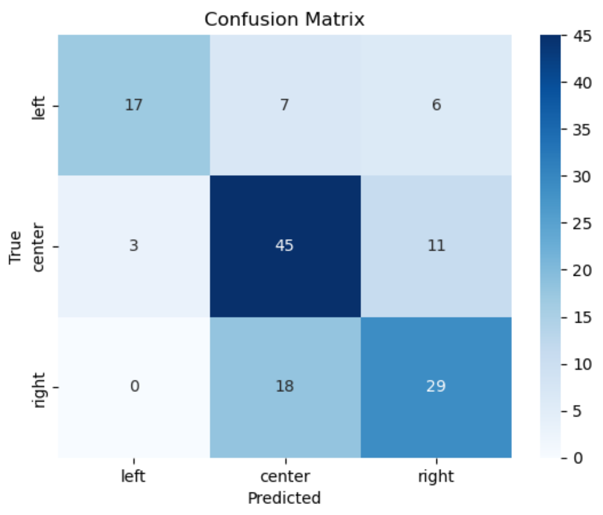
Here we split parties into three classes: "left", "center" and "right". The splits are made at lrgen < 3.5 and lrgen > 6.5, and selected these parameters:

Parameter	**Random Forest**	**LightGBM**
`use_idf`	True	True
`max_df`	0.8	0.8
`min_df`	0.1	0.1
`ngram_range`	(1, 2)	(1, 2)

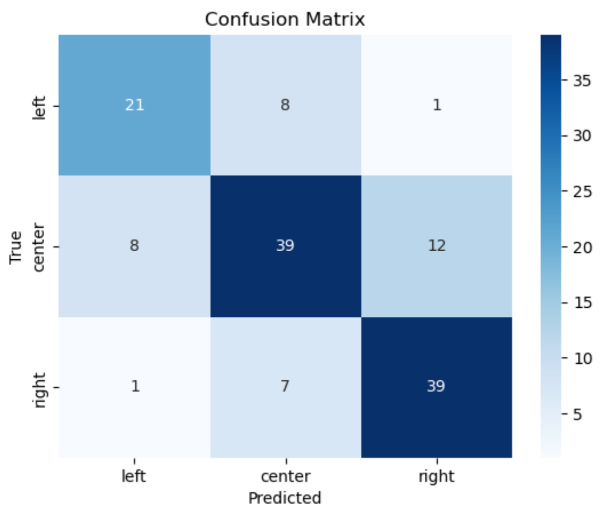
Table 5: Hyperparameter Settings for Random Forest and LightGBM

When extending to three categories, the confusion matrices reveal a noticeable increase in off-diagonal entries for both models, indicating more false positives and false negatives, especially around the “Center” category. For example, several true center manifestos are misclassified as either left or right, and vice versa, resulting in fewer true positives along the main diagonal.

Between the two models, LightGBM again achieves relatively more diagonal concentration—meaning it correctly identifies more left, center, or right manifestos—and thus incurs fewer off-diagonal misclassifications than Random Forest. Despite this improvement, both models see heightened misclassification errors overall due to the increased complexity of distinguishing three ideological blocks.



(a) Random Forest with TF-IDF ($accuracy = 0.67$)



(b) Light GBM with TF-IDF ($accuracy = 0.71$)

Figure 7: Hyperparameter Models with 3-Split

6.2.3 Feature Importance

Examining the most decisive features of both models suggests they have greater potential for broader generalisability compared to the naive linear regression model. Although the most important features differ between the two models, they remain intuitive and interpretable, and notably do not include country- or party-specific terms.

Feature	Importance
poverty	283
trade union	205
criminal	147
working	147
inequality	121
freedom choice	98
participation	90
competitiveness	84
discrimination	79
stability	76
breed	68
immigration	67
attract	67
abuse	64
competitive	56
ecological	55

(a) Top 20 important features (absolute scale)

Feature	Importance
poverty	0.006068
working	0.004827
competitiveness	0.003285
trade union	0.003217
discrimination	0.003072
inequality	0.002739
minimum wage	0.002638
immigrant	0.002430
police officer	0.002294
worker right	0.002279
sentence	0.002267
immigration	0.002172
officer	0.002154
criminal	0.002145
class	0.002134
war	0.002065

(b) Top 20 important features (normalized scale)

Table 6: Comparison of Feature Importance Across Scales

7 Conclusion

The goal of this study was to explore whether we could use machine learning to position political parties along the left-right spectrum. Initially, we attempted to predict continuous ideological scores directly from manifesto texts using regression, but this approach proved complex and prone to overfitting. Consequently, we shifted to a classification framework, first dividing parties into two categories (left vs. right) and then into three groups (left, center, right), which simplified the modeling task and allowed us to use robust models such as Random Forest and LightGBM.

It is generally challenging to classify parties perfectly because political ideologies exist on a continuum, and many parties occupy positions near the boundaries between categories. Parties that are close to these thresholds often exhibit mixed or moderate policy stances, making it difficult for any model to draw a sharp line between, for example, left and center or center and right. Even with sophisticated text analysis, subtle shifts in language or emphasis can lead to slight changes in perceived ideological position, which means that misclassifications are almost inevitable when working with inherently ambiguous, nuanced political discourse. Given these challenges, we consider our models to achieve reasonable performance overall.

7.1 Model Limitations

While our approach demonstrates promising results, it is not without limitations. The use of a bag-of-words representation, although computationally efficient, reduces text to mere frequency counts and loses crucial contextual, syntactic, and semantic nuances inherent in political discourse. This means that subtle differences in word order and idiomatic expressions, which are especially relevant in manifestos, are not captured, potentially leading to misinterpretation of a party's ideological signals. Moreover, our reliance on the Chapel Hill Expert Survey (CHES) to assign ideological scores introduces a layer of subjectivity, as these expert judgments, though respected, may not fully encapsulate the dynamic and multifaceted nature of party ideologies across different political contexts. Finally, because our training data is drawn from European manifestos in a certain time p, the external validity of our models is limited; the linguistic and contextual patterns they learn may not generalize well to parties in other regions, under different political systems or in different time periods.

7.2 Final Application

In the end, labeling political parties along a spectrum is always going to be somewhat subjective. Here we are going to present how our model classifies the major political parties in Germany based on their manifestos for the election in 2021 for which there was no lrgen score provided in the CHES data set.

Party Name	**Predicted Label**
Alliance‘90/Greens	left
The Left	left
Social Democratic Party of Germany	left
Free Democratic Party	right
Christian Democratic Union/Christian Social Union	center
Alternative for Germany	right

Table 7: Predicted labels for German 2021 manifestos

8 Bibliography

References

[1] Lehmann, P., Franzmann, S., Al-Gaddooa, D., Burst, T., Ivanusch, C., Regel, S., Riethmüller, F., Volkens, A., Weßels, B., & Zehnter, L. (2024). *The Manifesto Data Collection* (Manifesto Project [MRG/CMP/MARPOR], Version 2024a). Berlin: Wissenschaftszentrum Berlin für Sozialforschung (WZB) / Göttingen: Institut für Demokratieforschung (IfDem). `https://doi.org/10.25522/manifesto.mpds.2024a`.

[2] Rovny, J., Bakker, R., Hooghe, L., Jolly, S., Marks, G., Polk, J., Steenbergen, M., & Vachudova, M. A. (2024). *25 Years of Political Party Positions in Europe: The Chapel Hill Expert Survey, 1999–2024* [Working paper]. Chapel Hill Expert Survey Project.