COMP90049 Assignment 2: Predicting Supreme Court Rulings

Anonymous

1 Introduction

Predicting Supreme Court outcomes offers insights into judicial behavior and decision-making. This project aims to predict whether a Supreme Court ruling will reverse or affirm a lower court decision using machine learning models. The dataset includes over 4,500 cases, each represented by 17 features, such as case details, participants, and information about the presiding judges. The target label indicates whether the lower court's decision was reversed (successful appeal) or affirmed (unsuccessful appeal). Framed as a binary classification task, the project seeks to evaluate model performance in predicting these judicial outcomes

To address this task, we focus on three research questions: (1) How do baseline models compare with more complex machine learning models in predicting case outcomes? (2) How do additional features related to the court's decision (e.g., majority ratio, decision date) impact model performance? (3) Does the political orientation of the judges influence the decision outcome?

This project aims not only to develop predictive models but also to critically analyze the performance of different models and features in the context of legal decisions.

2 Literature Review

Predicting Supreme Court decisions has been an area of significant research, with datasets such as Super-SCOTUS playing a key role. Fang et al. (2023a,b) introduced this dataset, which links various court materials, including oral arguments and case metadata, to final rulings. Their research highlights the strong correlation between justices' language in court and their voting patterns, contributing to the growing body of work on judicial behavior prediction.

For representing court discussions, we employ Sentence-BERT introduced by Reimers and Gurevych (2019), which transforms textual data into 384-dimensional embeddings. This method enables efficient similarity comparisons across court hearings, enhancing our model's ability to process linguistic features.

For implementing the Model of Deep Neural Network, beyond legal applications, We also search for recent advances in deep learning which offer essential insights. He et al. (2015) introduced residual learning, solving vanishing gradient issues and improving network depth scalability. Keskar et al. (2017) explored large-batch training's effect on model generalization, while Li et al. (2017) visualized the loss landscape, revealing that flatter minima often lead to better generalization. These methods directly inform our work by improving model performance and generalization in high-dimensional datasets.

3 Method

3.1 Data Pre-processing

The dataset used in this project consists of various features extracted from Supreme Court case metadata and oral arguments. So initial preprocessing steps were crucial to prepare the data for model training. The following transformations were applied:

- Unrelated columns such as `case_id` and `title` were removed, as they provide no predictive value for the model.
- Cases with zero values in the `court_hearing_length` column were replaced with a small constant to avoid division by zero errors during feature scaling.
- Textual features like `court_hearing` were converted into 384-dimensional embeddings using the Sentence-BERT model.
- Categorical features, such as `petitioner`, `respondent`, and `issue_area`, were trans-

formed using one-hot encoding to create binary vectors for each category.

- Numerical features, including `utterances_number` and `court_hearing_length`, were standardized using `StandardScaler` to normalize the distribution of the data.

These steps ensure that the dataset is in a suitable format for model training, with all features properly scaled and transformed to facilitate learning.

3.2 Feature Engineering

Feature engineering was performed to extract meaningful insights from the existing features and improve model performance. Several new features were created:

- **Lagged Time**: The difference between the `argument_date` and the `year` to capture the length of time (into years) taken before official argument.
- **Speech Rate**: The ratio of the number of utterances (`utterances_number`) to the `court_hearing_length`, providing insights into the pace of the court's discussion.
- **Hearing Length in Months**: The difference between the `decision_date` and the `argument_date` to capture the length of time (into months) taken for the court to reach a decision.
- **Liberal Justices and Conservative Justices**: These features represent the number of justices voting for the decision from each political orientation. They are derived from the `justices` feature.
- **Liberal Ratio**: The ratio of liberal justices relative to the total number of justices voting, indicating the political leaning of the court in each case.

The final feature set used for model training is a combination of existing and newly engineered features. This comprehensive feature set is designed to capture the complex dynamics of Supreme Court decision-making and improve the model's ability to predict outcomes.

3.3 Machine Learning Models

To address the task of predicting Supreme Court decisions, we implemented two baseline models and two machine learning models: Logistic Regression (LR) and a Deep Neural Network (DNN).

3.3.1 Baseline Models

The first baseline model used was a Majority Class Baseline, where the model always predicts the majority class in the dataset. The second baseline, Random Guessing, randomly predicts either class (reversed or affirmed) with equal probability. These baselines set a minimal standard against which more sophisticated models can be measured.

3.3.2 Logistic Regression (LR)

For our LR model, we employed a grid search to optimize hyperparameters. The visualization process is shown in Figure 1. The grid search was configured as follows:

- `C`: [0.001, 0.01, 0.1, 1, 10, 100], controlling regularization strength.
- `penalty`: 'l2', applying L2 regularization to prevent overfitting.
- `solver`: 'lbfgs', 'liblinear', two solvers that handle small to medium-sized datasets well.

To ensure robust performance, we also used 5-fold cross-validation during the grid search, evaluating each configuration based on the metrics which will be mentioned next.

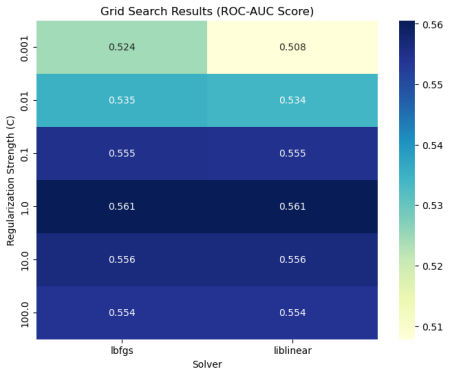

Figure 1: Grid-Search of Logistic Regression Model

3.3.3 Deep Neural Network (DNN)

A more complex model, the DNN, was designed to capture nonlinear relationships within the dataset. The network architecture consists of several fully connected layers:

- An input layer of size corresponding to the number of input features.
- Three hidden layers with 128, 64, and 32 neurons, respectively. Each hidden layer

is followed by batch normalization and ReLU activation for stable training and non-linearity.

- Dropout with a rate of 0.3 after each hidden layer to prevent over-fitting.
- A final output layer with a single neuron and Sigmoid activation for binary classification.

The model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 128. We employed early stopping to stop training after 10 epochs with no improvement in validation performance, preventing over-fitting. The model was trained for a maximum of 200 epochs, with k-fold cross-validation (set $k = 5$) to ensure robust evaluation. The learning curve of the DNN model is shown in Figure 2, which highlights the convergence of the model during training.

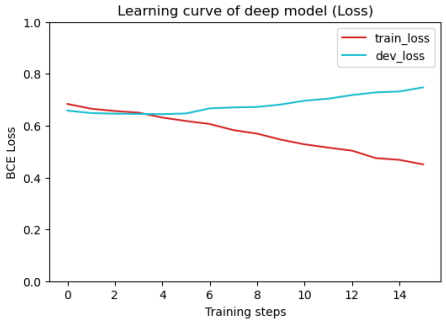

Figure 2: Learning Curve of Deep Neural Network Model

The validation results for each fold, including the best validation and training loss and accuracy, are presented in Table 1. On average, the model achieved a validation accuracy of 63.27% with a best validation loss of 0.6549 across the five folds.

Fold	Best Validation Loss	Best Validation Accuracy	Best Training Loss	Best Training Accuracy
1	0.6604	0.6371	0.6451	0.6346
2	0.6492	0.6511	0.6429	0.6446
3	0.6570	0.6236	0.6299	0.6463
4	0.6457	0.6497	0.6409	0.6379
5	0.6623	0.6020	0.6188	0.6577
Average	**0.6549**	**0.6327**	**0.6355**	**0.6442**

Table 1: K-Fold Validation Results for the Deep Neural Network Model

3.4 Evaluation Metrics

Given that this is a binary classification problem, several evaluation metrics were used to assess model performance:

- **Confusion Matrix**: This provides insights into the number of true positives, true negatives, false positives, and false negatives predicted by the model. It helps to assess the model's precision and recall, which are critical in understanding the trade-off between false positives and false negatives.
- **ROC Curve**: The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate, offering a comprehensive evaluation of model performance across different thresholds. The Area Under the Curve (AUC) was used as a summary statistic to compare models, as it balances sensitivity and specificity.

3.5 Experiments Strategy

For RQ1, we compared the performance of two baseline models (Majority Class, Random Guessing) with Logistic Regression (LR) and a Deep Neural Network (DNN) to assess their ability to predict Supreme Court outcomes.

In RQ2, we evaluated how adding decision-related features (`hearing_length_months`, `majority_ratio`) affected DNN performance. The same hyperparameters and k-fold cross-validation setup from RQ1 were applied to ensure fair comparison.

For RQ3, we examined the impact of political orientation, using the `liberal_ratio` as a feature in the DNN, to explore how the ratio of liberal to conservative justices influences court decisions.

4 Results

4.1 RQ1: Baseline Models vs. Machine Learning Models

The Majority Class Baseline always predicts the most frequent class, which in our dataset is "Reverse." The detailed performance metrics for the Majority Class Baseline are shown in Table 2.

Class	Precision	Recall	F1-Score	Support
Affirm	0.00	0.00	0.00	210
Reverse	0.64	1.00	0.78	367
Accuracy		0.64		
Macro avg	0.32	0.50	0.39	577
Weighted avg	0.40	0.64	0.49	577

Table 2: Performance of Majority Class Baseline Model

The Random Guessing Baseline, which randomly predicts "Affirm" or "Reverse" with equal probability, performs slightly worse than

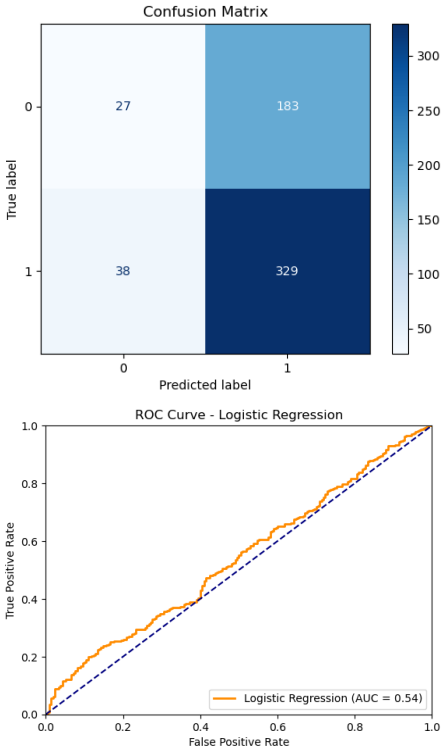

Figure 3: Confusion Matrix and ROC Curve for LR Model

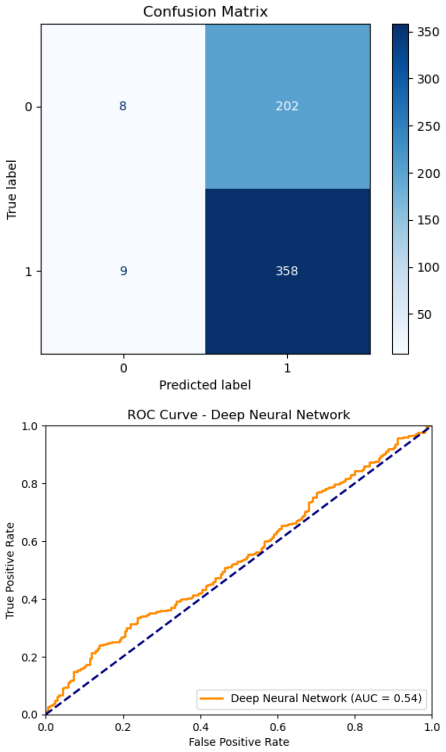

Figure 4: Confusion Matrix and ROC Curve for DNN Model

the Majority Class Baseline. It achieves an accuracy of 0.51 and a ROC-AUC score of 0.5156. The performance metrics are shown in Table 3.

Class	Precision	Recall	F1-Score	Support
Affirm	0.38	0.54	0.44	210
Reverse	0.65	0.49	0.56	367
Accuracy	0.51			
Macro avg	0.51	0.52	0.50	577
Weighted avg	0.55	0.51	0.52	577

Table 3: Performance of Random Guessing Baseline Model

The Logistic Regression model, after hyperparameter tuning and cross-validation, outperforms both baseline models. It achieves an accuracy of 0.62 and an ROC-AUC score of 0.5402. The performance metrics are shown in Table 4.

Class	Precision	Recall	F1-Score	Support
Affirm	0.42	0.13	0.20	210
Reverse	0.64	0.90	0.75	367
Accuracy	0.62			
Macro avg	0.53	0.51	0.47	577
Weighted avg	0.56	0.62	0.55	577

Table 4: Performance of Logistic Regression Model

The DNN model, which uses multiple hidden layers and regularization techniques, performed pretty well among the four models. As shown in Figure 4, the DNN model achieved a validation accuracy of 0.6343 and a validation loss of 0.6544.

Table 5 summarizes the performance of the four models based on accuracy and ROC-AUC score on dev-set.

Model	Accuracy	ROC-AUC
Majority Class Baseline	0.64	N/A
Random Guessing Baseline	0.51	0.5156
Logistic Regression	0.62	0.5402
Deep Neural Network (DNN)	0.63	0.6544

Table 5: Model Comparison on Accuracy and ROC-AUC

4.2 RQ2: Incorporating Decision-Related Features

The model achieved a validation accuracy of 0.6360 with a ROC-AUC of 0.59, as shown in Figure 5.

4.3 RQ3: Impact of Political Orientation of Justices

This time, the model achieved a validation accuracy of 0.6291, with a ROC-AUC of 0.55, as illustrated in Figure 6.

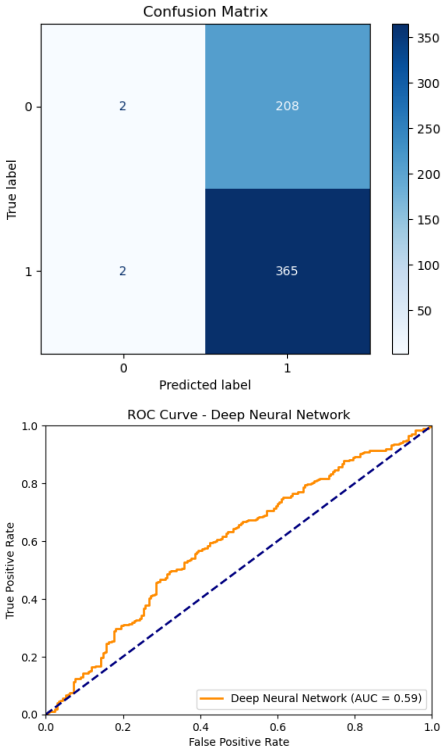

Figure 5: Confusion Matrix and ROC Curve for DNN Model with Decision-Related Features (RQ2)

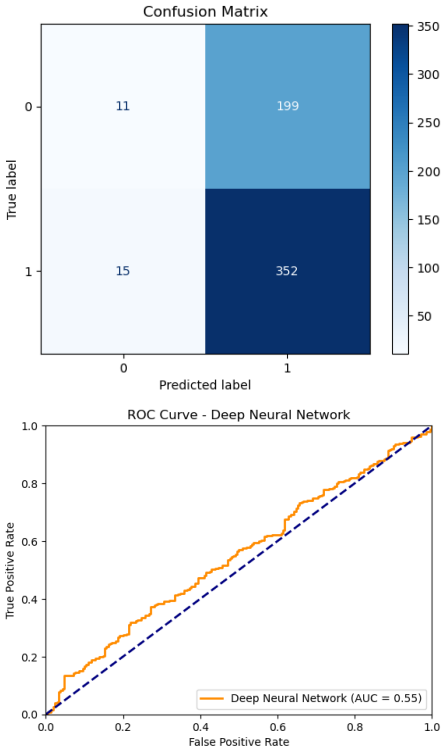

Figure 6: Confusion Matrix and ROC Curve for DNN Model with Political Orientation Features (RQ3)

5 Critical Analysis

5.1 Contextualizing Results

The performance of the models across research questions highlights key insights into their strengths and limitations, beyond the raw metrics.

For **RQ1**, the Deep Neural Network (DNN) outperformed Logistic Regression by capturing complex, non-linear relationships. However, the model struggled with class imbalance, particularly underpredicting "Affirm" decisions. This reflects the challenge of learning from minority classes in imbalanced datasets. The DNN's performance indicates that model complexity alone cannot overcome limitations in data representation.

For **RQ2**, adding decision-related features (`decision_date` and `majority_ratio`) moderately improved the DNN's generalization. These features provided valuable context, but the challenge in predicting "Affirm" decisions persisted, suggesting that case-specific factors not captured by these features might influence judicial rulings. While decision metadata enhances model performance, it is not sufficient to fully capture the intricacies of Supreme Court decisions.

For **RQ3**, using the `liberal_ratio` feature only achieved basic good result in predictive accuracy and ROC-AUC. This suggests that political orientation is a factor in decision-making, but its influence is complex and interacts with many other case-specific variables. This result aligns with existing literature, reinforcing the notion that judicial decisions are shaped by a multitude of interrelated factors beyond political leanings.

5.2 Optimizing the Deep Neural Network

The optimization of the DNN was a crucial aspect of the study. Initially, the model suffered from under-fitting, as evidenced by high training loss and poor performance on both training and validation sets. To address this, the model complexity was gradually increased by adding layers and neurons. However, this led to over-fitting, as the validation loss began to diverge from the training loss. Basically, I follow the optimization logic which is shown in Table 6

To mitigate over-fitting, several techniques were implemented:

- **Dropout** (rate of 0.3) was applied to randomly deactivate neurons during training, which helped the network generalize better by preventing reliance on specific neurons.
- **Batch Normalization** was added after each layer, which stabilized training and allowed for faster convergence.
- **Early Stopping** was used to halt training when validation loss plateaued, preventing the model from over-fitting beyond the point of optimal generalization.
- The **Adam Optimizer** was selected for its ability to adapt learning rates for individual parameters, speeding up convergence compared to SGD. A **Learning Rate Scheduler** was employed to reduce the learning rate when progress stalled, further fine-tuning the model.

Additionally, the use of **K-Fold Cross-Validation** ensured that the model's performance was evaluated across different data splits, providing a more robust understanding of its generalization capabilities. This cross-validation was critical for mitigating the impact of potential biases in the dataset and ensuring that the performance metrics reflected the model's true ability to generalize to unseen data.

Step	Condition	Action	Explanation
Evaluate loss on training data	Large	Model Bias	The model is under-fitting. Make the model more complex.
	Small	Proceed to next step	The model fits the training data well. Check performance on testing data.
Evaluate loss on testing data	Large	Over-fitting	The model has over-fitted the training data.
		Solution 1: Add more training data	Introduce more data to generalize better.
		Solution 2: Data augmentation	Create artificial data variation to improve generalization.
		Solution 3: Simplify the model	Reduce model complexity (e.g., fewer layers or neurons).
	Small	Good Model	The model performs well on both training and testing data.

Table 6: Model Optimization Process: Evaluating Loss on Training and Testing Data

5.3 Challenges and Limitations

Despite improvements through careful tuning, several challenges persisted. The primary issue was class imbalance, with more "Reverse" than "Affirm" decisions, causing the model to favor the majority class and resulting in low recall for the minority class. Techniques like Dropout and Batch Normalization helped reduce over-fitting but could not fully address the imbalance. Future work could explore data augmentation or weighted loss functions to improve minority class recall.

The inherent complexity of predicting Supreme Court rulings posed another challenge. Judicial decisions are influenced by numerous factors, many of which are not easily captured by the dataset. While the Deep Neural Network identified patterns, it could not account for the nuanced legal reasoning and socio-political dynamics driving court outcomes, limiting the potential gains machine learning models could achieve.

Feature engineering improved model performance, but the overall impact of decision-related (`hearing_length_months`, `majority_ratio`) and political orientation features (`chief_justice`, `justices`) was modest. More refined features, such as those from oral argument content, may better capture the complexity of judicial decisions.

Finally, balancing model complexity and generalization was an ongoing challenge, with over-fitting in more complex models and under-fitting in simpler ones.

6 Conclusion

This project highlighted the complexities of predicting Supreme Court decisions using machine learning. Despite the DNN outperforming baseline models (very close to Majority voting model), challenges such as class imbalance, particularly in predicting "Affirm" decisions, persisted. The modest gains from incorporating decision-related and political orientation features emphasize the nuanced, multi-faceted nature of judicial decision-making. Through this assignment, we identified the importance of model stabilization techniques like Batch Normalization, Dropout, and Early Stopping. However, the intrinsic difficulty of the task indicates that future improvements will require advanced techniques, such as data augmentation and more refined, context-specific features to better capture judicial reasoning.

References

Fang, B., Cohn, T., Baldwin, T., and Frermann, L. (2023a). More than Votes? Voting and Language based Partisanship in the US Supreme Court. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4604–4614, Singapore. Association for Computational Linguistics.

Fang, B., Cohn, T., Baldwin, T., and Frermann, L. (2023b). Super-SCOTUS: A multi-sourced dataset for the Supreme Court of the US. In Preo\textcommabelowtiuc-Pietro, D., Goanta, C., Chalkidis, I., Bar-

rett, L., Spanakis, G., and Aletras, N., editors, *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 202–214, Singapore. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385v1.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. https://arxiv.org/abs/1712.09913v3.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.