Sentence Prediction Using Ensemble Learning and Deep learning approach

Rishabh Sharma
College of Computing and Informatics
Drexel University
Philadelphia, USA
ORCID: 0000-0002-2986-9186

Divyanshu Kumar
College of Computing and Informatics
Drexel University
Philadelphia, USA
e-mail: dk984@drexel.edu

Shaitun Ma
College of Computing and Informatics
Drexel University
Philadelphia, USA
e-mail: bm3336@drexel.edu

***Abstract*—In the area of law and justice, sentencing prediction is a crucial instrument. The goal of this study is to develop a system for predicting the kind of sentencing from the provided docket data. While most studies solely focus on the docket description, this study along with the description also tries to concentrate on additional demographic parameters such as race, sex, judges, court name, court type, and title of the disposing authority in addition to the docket description. This study is based on the docket data collected for the cases in the Philadelphia Court of Common Pleas. This work aims to predict the kind of punishment for a particular criminal case using a decision tree classifier, ensemble learning techniques like AdaBoost and XGBoost, and deep learning-based models.**

***Index Terms*—Deep learning, machine learning, Ensemble Learning, Sentencing prediction, Law, justice.**

I. Introduction

The number of filings grew by 5 percent in the U.S. courts of appeals, according to the Federal Judicial Caseload Statistics 2020. In 2020, there were 50,258 filings in the 12 regional courts of appeals, up 2,281 appeals or 5 percent. This increase was caused by more criminal appeals and other private civil appeals being filed, which more than compensated declines in petitions from U.S. prisoners and private prisoners. It is impossible to fundamentally resolve the paradox of having too many cases and too few judges by merely boosting trial resources and increasing the number of judges since the court is overburdened by the sheer volume of cases, and because it takes time to train highly skilled judges. As a result, we can lighten the load on judges by using sentencing prediction algorithms. The goal of intelligent justice trials is to employ computer technology to anticipate penalties from semi-structured adjudication papers. Sentencing prediction is a key component of these trials. Additionally, lawyers can employ sentence prediction models to foresee the statistical outcome of a case before accepting it. By this model, they can appropriately counsel their clients on the wisdom of plea negotiating choices, guilty pleas, and appeals against punishment, attorneys must be able to foresee sentencing judgments themselves.

Identify applicable funding agency here. If none, delete this.

The application of boosting approaches in sentence prediction is covered in the study. Additionally, deep learning and a decision tree method for decision-making will also be emphasized. XGBoost and AdaBoosted decision trees (ADTs). ADTs and XGB capture advances in prediction when there are numerous variables, the majority of which have no additional predictive value. By predicting Supreme Court rulings using a cutting-edge dataset that incorporates textual data, other demographics, and case-level information, we show their value. XGBoost Classifier has an accuracy of 79 % for all models created, compared to 65 % for AdaBoost, 73 % for deep learning, and 69 % for decision tree classifiers,

II. Related Works

Sentence prediction has been a study subject carried out by various researchers, and Attorneys[1-3]. Previous studies solely looked at the docket's text description[5]. While this study focuses on a variety of additional characteristics, including sex, race, age, and others, in addition to the description data. Some other papers have followed various methods to predict the sentence type for a docket. Bunn and Wright[5], focus on the technique of bootstrapping like a Random forest algorithm. Other researchers such as Ouyang L, Huang R, Chen Y, and Qin Y researched sentencing prediction by developing logic for the judicial system using abductive learning. Some researchers have examined the problem of sentence prediction study on the theory of utilizing text classification [7] by extracting some features since the emergence of deep learning techniques. Both the Legal AI Challenge 2021 (LAIC2021) and the Challenge of AI in Law 2018 (CAIL2018) opened sentencing prediction tracks, with the traditional text classification technique serving as the baseline model. By introducing pertinent legal provisions, Luo et al. [8] established a neural network model based on an attention mechanism to forecast case charges based on the factual description of the case.

For this research, we will discover methods like boosting, deep learning models, and decision trees to draw a model to classify different types of sentencing. Our study will use both textual as well as demographic data to draw suitable conclusions from our models.

III. Methods

In this section, we will describe our proposed model in terms of the logic of the trial: it is divided into four modules, namely knowledge base preparation, model building, sentencing calculation and hyper-parameter tuning.

A. Knowledge Base Preparation

In order to build our model. The data can be collected from the 2021- Datathon Philadelphia. This contains datasets like defendant docket details, offense disposition, bails, and many more. For this project, we have chosen the most pertinent datasets, i.e. defendant docket details and offense disposition. These datasets also contain the description data for the various dockets.

To generate the data frame, After data cleaning. The categorical variables are turned into numeric features by applying label encoding since models cannot understand categorical data because those categories have no significance for them. This information must be prepared if we want a computer to be able to process it. The bag of words approach is used to transform the description data into the appropriate vector representation. By doing the feature indices mapping, we produce a sparse matrix, which is our encoded data. Since there are five classes, we can forecast the type of sentence. In order to train the classification model using the data, we transformed them into single hot encoded values. After analyzing the imbalance in the dataset, SMOTE(Synthetic Minority Over-sampling) was to be done to get an equal number of data points for each class.

B. Model Building

This is a classification problem since our data contains 5 different types of punishment. The process of categorization consists of identifying, comprehending, and classifying concepts or things into specified groupings or "sub-populations." Probation, confinement, No Further Penalty, Merged, and IPP are the 5 categories. We categorize the sentencing type for a docket as Probation, Probation, Confinement, No Further Penalty, Merged, and IPP using our processed data. We have used 4 categorization models to solve the issue. Specifically, a deep learning-based classifier, a decision tree classifier, an AdaBoost classifier, and an XGBoost classifier.

1) . Decision Tree Classifier: The trees begin with a d = X, y, from which we must extract the tree structure and decision-making processes for each node. Each node will divide the data set into two or more disjoint subsets, each designated by its own layer number, or d(m,i)*, where m is the current layer number. The subset is said to be pure and the node will be designated as a leaf node, indicating that this branch of the tree has terminated if all of our labels in it belong to the same class. The splitting criteria will be applied again if it is determined that the node is impure. The impurity factor for the above model will be calculated by using the Gini impurity which is given by the formula.

$$GiniIndex = 1 - \sum_{n=1}^{n} p(x_i)^2$$

It is necessary to conduct an experiment using data and the splitting criterion since the Gini index favors larger partitions (distributions) and is relatively simple to apply, whereas information gain supports smaller partitions (distributions) with a range of distinct values. Thus, we will use gini index for the model

2) . Deep Learning Classifier: The foundation of conventional statistical classification techniques like discriminant analysis is the Bayesian decision theory. To calculate the posterior probability that serves as the basis for the classification decision in these procedures, an underlying probability model must be created. The fact that statistical models only perform well when the underlying assumptions are met is one of their main limitations. The numerous criteria or assumptions that are used to build the models have a significant impact on how effective these methods are. Before the models can be properly applied, users must have a solid understanding of both the data attributes and the capabilities of the models.
The input data is sent through the network in the forward direction during the forward propagation step. The data is accepted by each hidden layer, processed in accordance with the activation function, and then passed on to the following layer. A pre-activation weighted sum of inputs, or the linear transformation of weights with respect to inputs available, is used for this stage. The neuron decides whether or not to transmit this information on based on this aggregated total and activation function. The activation function receives the computed weighted sum of inputs. A mathematical function known as an activation function gives the network more non-linearity. Relu activation.

$$R(x_i) = max(0, x_i)$$

The Softmax activation have been utilized for this model's hidden and output layers, respectively. Since it gives more flexibility when dealing with a multi-class classification problem

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

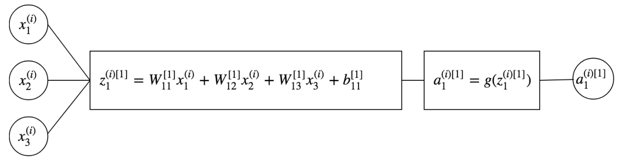
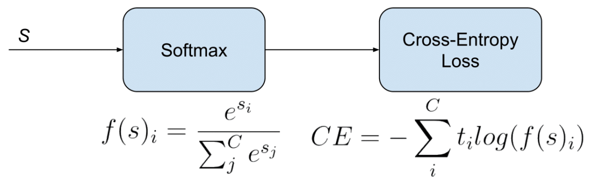

$x_1^{(i)}$
$x_2^{(i)}$
$x_3^{(i)}$
$z_1^{(i)[1]} = W_{11}^{[1]}x_1^{(i)} + W_{12}^{[1]}x_2^{(i)} + W_{13}^{[1]}x_3^{(i)} + b_{11}^{[1]}$
$a_1^{(i)[1]} = g(z_1^{(i)[1]})$
$a_1^{(i)[1]}$

Fig. 1. forward propagation with pre-activation and activation applied

Back-propagation is required to discover the optimal weights and biases for the neural network. To adjust the weights in the neural network layers, the error rate of a forward propagation is fed backward through the layers. This error function, also known as the loss function, is selected to be the categorical cross-entropy for multi-class issues. To calculate



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \quad CE = -\sum_i^C t_i log(f(s)_i)$$

Fig. 2. Categorical Cross entropy

the values of all the weights and biases the generalised rule can be given by Optimizers are programs or techniques that

$$\frac{\partial L}{\partial w_{ij}} = \delta_j x_i$$

$$\delta_j = \sum_k \delta_k w_{jk} y_j (1 - y_j) x_i$$

modify the weights and learning rates of your neural network in order to minimize losses. Optimizers facilitate quicker results. Stochastic gradient descent is extended by the Adam optimization technique. For all weight updates, stochastic gradient descent maintains a constant learning rate (referred to as alpha), which does not change throughout training. As learning progresses, a learning rate is maintained and independently adjusted for each network weight (parameter). A stochastic gradient descent method called Adam optimization, on the other hand, is based on adaptive estimation of first- and second-order moments.

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Fig. 3. Adam optimization algorithm

3) . Random Forest Classifier: A huge number of ensemble-operating decision trees are used in random forest. Each tree provides a categorization result, and the classification result with the most votes wins. A low correlation between individual models is crucial. The random forest model's strength comes from this. Multiple classifiers are useless if they all make errors that point in the same direction. An ensemble model will move in the proper direction even though some trees may be making mistakes while others may be correct. Low correlation in the trees must be maintained since decision trees are highly data-sensitive. The same fact is utilized by random forest

The data set may be sampled at random by each tree and used to build the trees. This procedure is referred to as bagging. The fact that we are not breaking down the data set into smaller data sets is crucial here. Instead, we use replacement random samples of size N. In addition, a decision tree can only select from a random subset of characteristics, as opposed to the decision tree method, which considers all feasible features and selects the one that provides the greatest separation between nodes in order to increase unpredictability.

4) . AdaBoost Classifier: During the training phase, n trees are required. First, priority is given to the improperly classified document. Only them are sent to the second model as input. The procedure continues until the number of base learners, which we specify at the beginning, is reached. The random forest and AdaBoost vary in that the random forest does not specify the depth of the tree. In contrast to Adaboost, where it only creates two nodes, some trees can be larger than the others. They are referred to as stumps.

The Adaboost algorithm favours this since the stumps are poor learners. The arrangement of the stumps is crucial because it establishes the framework for the development of subsequent stumps.

$$SampleWeightinitialization(T) = 1/N$$
$$Performanceofastump = log((1 - T)/T)$$
$$NewWeight = SampleWeight * e^-(Performance)$$

The frequency of entries that are incorrectly classified will be higher in the new data set. Incorrectly classified submissions are placed in larger buckets, increasing their likelihood of selection.It will build new decision trees or stumps based on the new data set until it determines that they have less error than the sample weight that was initialized in the first step.

5) . XGBoost Classifier: The category of boosting strategies in ensemble learning includes XGboost. For a model to be more accurate, ensemble learning typically uses numerous predictors. By adding some weight to the previous models' errors, we can correct them using the boosting technique.
In difference to other boosting algorithms where weights of wrong classifications are increased, in Gradient boosting the loss function is optimized. XGBoost stands for Extreme gradient boosting which means it inherits the properties of Gradient boosting with some regularization factors.

The dependent feature will first be averaged out by the base model (M0), which will then provide the predictions. We are aware that this model has a loss, which will be improved in the following model (M1). The aim in Model M1 will be the loss incurred (variances) in Model M0. Up until this point, the gradient boosting method has been used. $\lambda- >$

regularization parameter
$\gamma->$ for auto tree pruning
$\eta->$ how many models will converge

Similarity Score$(S.S.) = (S.R^2)/(N + \lambda)$

Here, S.R is the sum of residuals and N is the number of Residuals
Gain = $S.Sbeforesplit - S.Safterthesplit.$

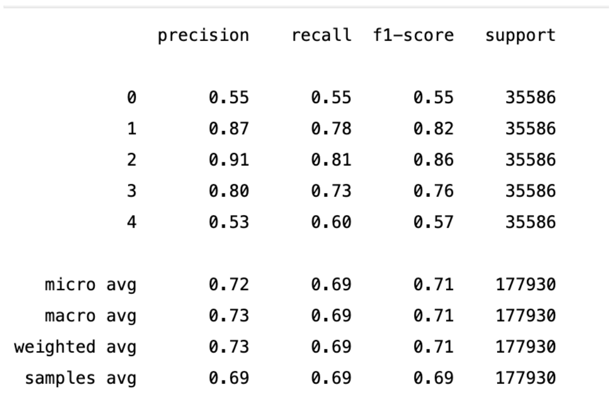
Now calculate γ which we provide at the start and is used in making the splits. If Gain¿ γ only then splitting will happen. This enables auto tree pruning. The greater the γ, the more the pruning. For regularization and preventing over-fitting, we must increase the λ which we initially set to 0. But one should always be careful about the fact that greater the λ value lesser is the similarity score, the lesser the Gain value, and the more the pruning.

IV. Results and Conclusions

This research examines the legal sentencing prediction based on docket data. The following four models were selected as the reference models to compare the experimental outcomes. Outcomes of the models can be measured by their accuracy, f1-score, precision and recall for the model. These measures can be calculated by:

A. Decision Tree Classifier

For the decision tree classifier the model was able to classify each label with an accuracy of 60% The classification report shows us that the model classified 91% of data accurately for class 2.

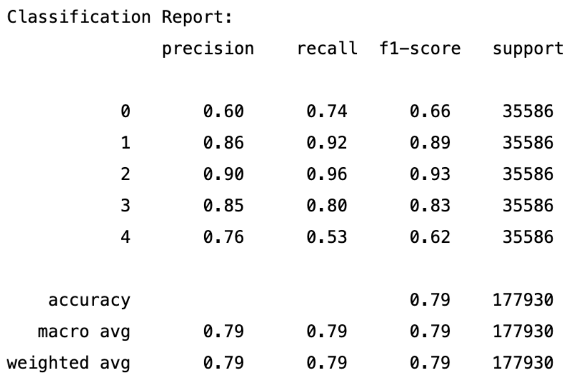


```
              precision    recall  f1-score   support

           0       0.55      0.55      0.55     35586
           1       0.87      0.78      0.82     35586
           2       0.91      0.81      0.86     35586
           3       0.80      0.73      0.76     35586
           4       0.53      0.60      0.57     35586

   micro avg       0.72      0.69      0.71    177930
   macro avg       0.73      0.69      0.71    177930
weighted avg       0.73      0.69      0.71    177930
 samples avg       0.69      0.69      0.69    177930
```

Fig. 4. Decision Tree Classifier

B. XG Boost Classifier

For the XGBoost Classifier the accuracy score turns out to be 79% giving us one of the best results out of all the trained models. for classification of the labels 1, 2 and 3. The precision was 86%, 90% and 85%.

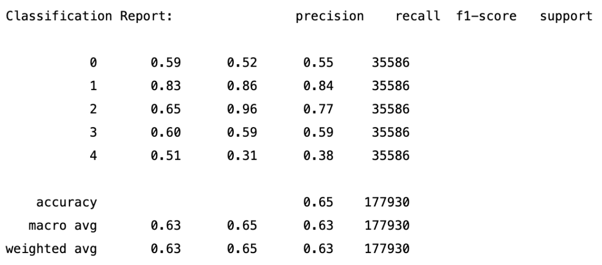


```
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.74      0.66     35586
           1       0.86      0.92      0.89     35586
           2       0.90      0.96      0.93     35586
           3       0.85      0.80      0.83     35586
           4       0.76      0.53      0.62     35586

    accuracy                           0.79    177930
   macro avg       0.79      0.79      0.79    177930
weighted avg       0.79      0.79      0.79    177930
```

Fig. 5. XGBoost Classifier

C. Ada Boost Classifier

AdaBoost Classifier was able to give out results by classifying the labels correctly 65% of the time. This model performed the worst out of all the classification models used. These results showcased are after the hyper-parameter tuning.

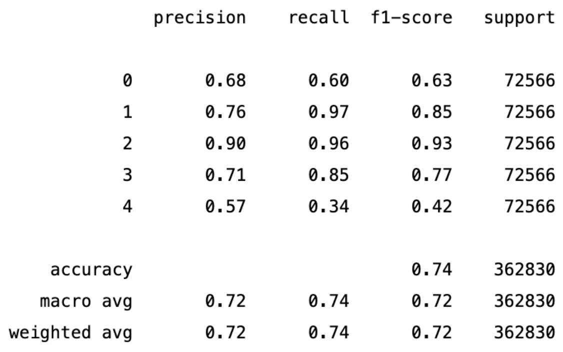


```
Classification Report:              precision    recall  f1-score   support

           0       0.59      0.52      0.55     35586
           1       0.83      0.86      0.84     35586
           2       0.65      0.96      0.77     35586
           3       0.60      0.59      0.59     35586
           4       0.51      0.31      0.38     35586

    accuracy                           0.65    177930
   macro avg       0.63      0.65      0.63    177930
weighted avg       0.63      0.65      0.63    177930
```

Fig. 6. Adam optimization algorithm

D. Deep Learning Classifier

deep learning model was the second best classification with an accuracy of 74% with good accuracy, precision and recall scores for all the labels.

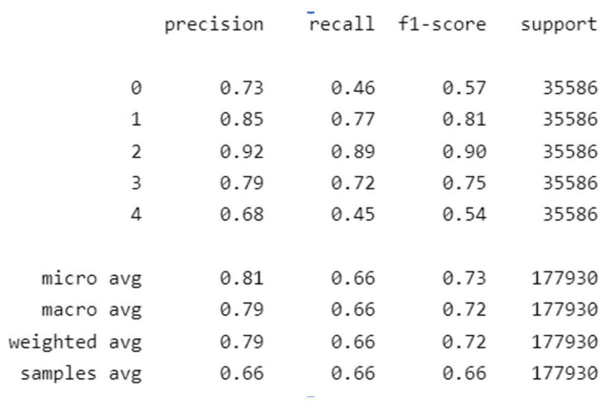


```
              precision    recall  f1-score   support

           0       0.68      0.60      0.63     72566
           1       0.76      0.97      0.85     72566
           2       0.90      0.96      0.93     72566
           3       0.71      0.85      0.77     72566
           4       0.57      0.34      0.42     72566

    accuracy                           0.74    362830
   macro avg       0.72      0.74      0.72    362830
weighted avg       0.72      0.74      0.72    362830
```

Fig. 7. Adam optimization algorithm

E. Random Forest Classifier

The Random Forest Classifier, the accuracy score turns out to be 60% with good precision, recall and f1 scores for all the class labels. This score was generated after hyper-parameter tuning on the model.



```
              precision    recall  f1-score   support

           0       0.73      0.46      0.57     35586
           1       0.85      0.77      0.81     35586
           2       0.92      0.89      0.90     35586
           3       0.79      0.72      0.75     35586
           4       0.68      0.45      0.54     35586

   micro avg       0.81      0.66      0.73    177930
   macro avg       0.79      0.66      0.72    177930
weighted avg       0.79      0.66      0.72    177930
 samples avg       0.66      0.66      0.66    177930
```

Fig. 8. XGBoost Classifier

V. Future Work and Extensions

There are various other methods that could have been used for the classification purpose. Some of the researches highlight the use of Advanced neural networks like CNN, GRU and LSTM models for the classification. Other techniques like Abductive learning will also be used in the future work for the research. The current data deals with limited information from the dockets of each filing in Philadelphia. The data spectrum can be broadened by adding more data from other cities across the states and focus on more wholesome data other than docket details.

VI. References

The above research is referenced from various researches over the topic of sentencing and punishment prediction in the jurisdiction area. Various researches and articles highlight the use of various algorithms and methods to draw judge-like predictions from the case data. This research is concentrated in the courts of Philadelphia and is an extension of using 2022-Social Justice Datathon data, Philadelphia.

References

[1] Kaufman, A., Kraft, P., Sen, M. (2019). Improving Supreme Court Forecasting Using Boosted Decision Trees. Political Analysis, 27(3), 381-387. doi:10.1017/pan.2018.59

[2] Benjamin, L.; Tom, C. The Supreme Court's Many Median Justices. Am. Political Sci. Rev. 2012, 106, 847–866. [CrossRef]

[3] Huang, Y.X.; Dai, W.Z.; Yang, J.; Cai, L.W.; Cheng, S.F.; Huang, R.Z.; Li, Y.F.; Zhou, Z.H. Semi-Supervised Abductive Learning and Its Application to Theft Judicial Sentencing. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 1070–1075.

[4] Katz, D.M.; Bommarito, M.J., II; Blackman, J. A General Approach for Predicting the Behavior of the Supreme Court of the United States. PLoS ONE 2016, 12, e0174698. [CrossRef]

[5] Bunn, Derek Wright, George. (1991). Interaction of Judgemental and Statistical Forecasting Methods: Issues Analysis. Management Science. 37. 501-518. 10.1287/mnsc.37.5.501.

[6] Ouyang, L.; Huang, R.; Chen, Y.; Qin, Y. A Sentence Prediction Approach Incorporating Trial Logic Based on Abductive Learning. Appl. Sci. 2022, 12, 7982. https://doi.org/10.3390/app12167982

[7] https://doi.org/10.48550/arXiv.1807.02478

[8] Luo, B.F.; Feng, Y.S.; Xu, J.B.; Zhang, X.; Zhao, D.Y. Learning to Predict Charges for Criminal Cases with Legal Basis. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), Copenhagen, Denmark, 7–11 September 2017; pp. 2727–2736.