Can we predict which patents will be used in litigation?

Deepak Krishnamurthy, Kolby Devery, Deanna Emery, Matt Pitz

Berkeley
UNIVERSITY OF CALIFORNIA

Why Patent Litigation?

At a glance

- Our project focuses on patent litigation filed in the US
- Patent law determines who "owns" innovation
- Each year 2,000+ patent infringement cases are filed

Pay the bridge troll

- "Patent trolls" or "Non-Practicing Entities" account for 60%+ of patent litigation
- These companies' monetize patents rather than produce products or services
- Courts awarded $4.67B in damages in 2020

Source: Why Patent Litigation Was on the Upswing in 2012 (infotoday.com); Patent Litigation Statistics - Feuds about Intellectual Property (legaljobs.io)

Our Goal

Identify which patents will be used in litigation

- There may be certain features of a patent filing that make it likely to be used in ligation
- Patent data and patent litigation filings are readily available
- A machine learning model may be able to detect those features and offer useful predictions

Challenges

- Data for patent filings may exceed our hardware limits
- Patent litigation is partially driven by economic factors not present in our data and not publicly available at scale
- Patent trolls are not easily identifiable
- Small percentage of patents are litigated

Data

Sources

Patent Applications

- https://patentsview.org/download/data-download-tables

Patent Litigations

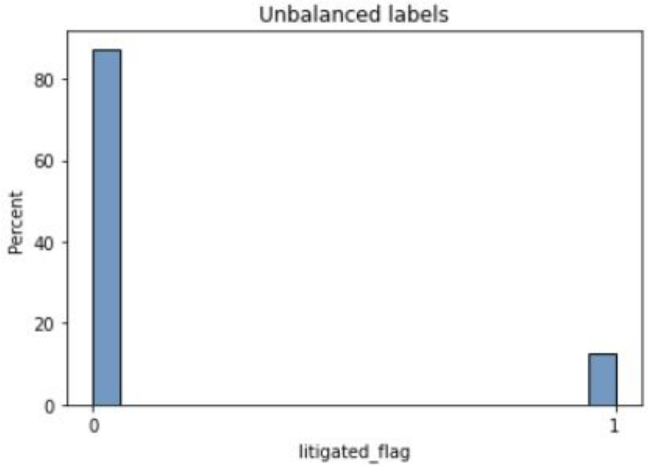
- https://npe.law.stanford.edu/

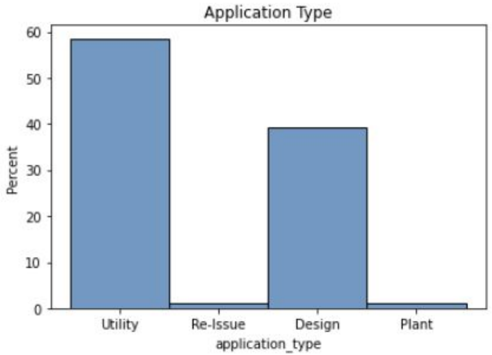
Patent Claims

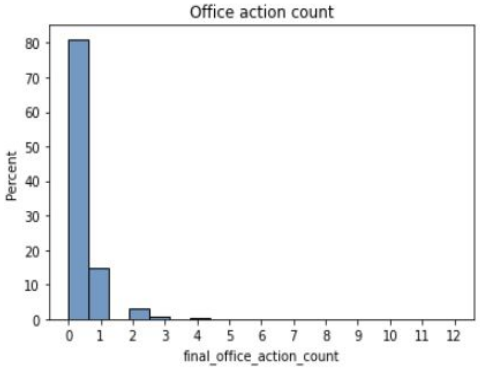
- Google BigQuery - patents-public-data.patents.publications table from Google

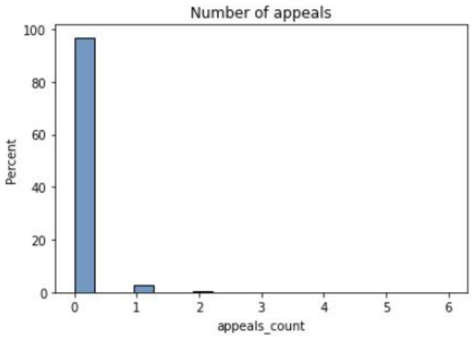
Data

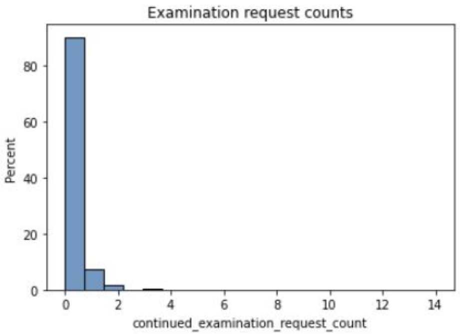
Distributions











BERT and Feature Engineering

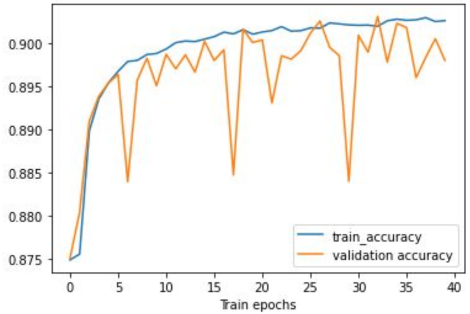
- Claims data extracted using Patent ID and Google BigQuery
    - Claims data merged with patent data
- Data Feature Engineered for final analysis
    - First 3 characters of CPC code and replace special characters with NaN
    - Mean values for Art Unit Number, Office action count, Interview count, Appeals Count
    - One Hot Encoding → CPC code, entity status, art unit number
- BERT Claims Encoding
    - “*all-MiniLM-L6-v2*” model used for BERT embedding
    - Model trained on Wikipedia, Yahoo Answers etc
    - Claims data vectorized using all-MiniLM-L6-v2 model
    - Model trainability enabled during modeling phase

	application_number	claims	claims_vector
0	US13902838	i : 1. a passageway barrier for a passag...	[-0.014044481, -0.018757526, -0.03600968, 0.00...
1	US13902838	i : 1. a passageway barrier for a passag...	[-0.014044481, -0.018757526, -0.03600968, 0.00...
2	US29686959	the ornamental design for a mask shell,...	[0.030425595, 0.18532415, 0.030899696, -0.0182...
3	US29686959	the ornamental design for a mask shell,...	[0.030425595, 0.18532415, 0.030899696, -0.0182...
4	US29643639	the ornamental design for a blade, as s...	[-0.03666717, 0.119967185, -0.02815474, -0.006...

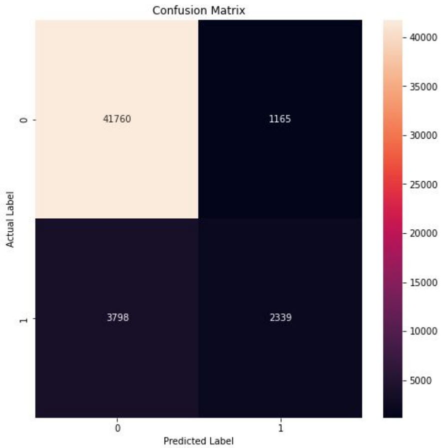
Approach 1: Logistic Regression

- Binary classification model used
- Data passed to model:
    - Vectorized Claims data
    - One-hot encoded categorical data → CPC code, art unit number, application type, entity status
- Parameters used:
    - Iterate through combinations of optimizers, learning_rate, batch size and output bias
    - Parameters chosen: SGD, learning rate = 0.01, batch_size = 32 and no output bias
- Imbalanced data → 12.5% litigated claims in total
- Weights added to minority class
- Fine-tuning and transfer learning improved by setting model training to True

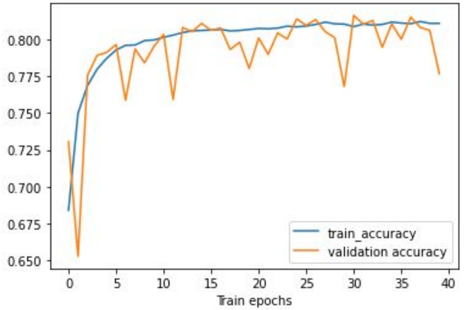
Approach 1: Logistic Regression



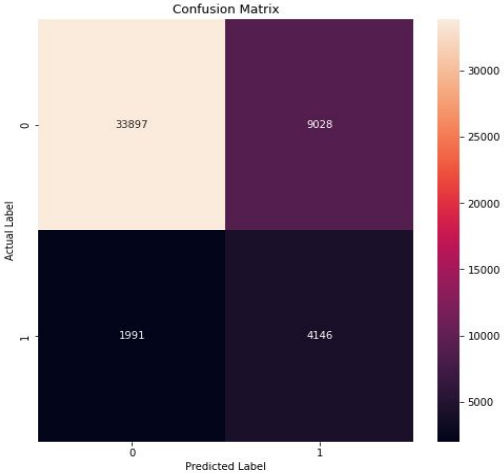

```
Training accuracy is: 90.26%
Validation accuracy is: 89.8%
```



Model without weights




```
Training accuracy is: 81.11%
Validation accuracy is: 77.69%
```



Model with minority weights

Approach 1: Logistic Regression Results

Metric	Unweighted	Weights Halved	Weighted
True Pos	2339	2398	4146
False Pos	1165	1258	9028
True Neg	41760	41667	33897
False Neg	3798	3739	1991
Accuray	89.9%	89.81%	77.54%
Precision	66.75%	65.59%	31.47%
Recall	38.11%	39.74%	67.55%
AUC	78.75%	76.95%	79.67%
PRC	53.5%	52.84%	45.03%
F1	48.52%	48.97%	42.93%
F2	41.69%	42.51%	54.95%

Approach 2: Random Forest

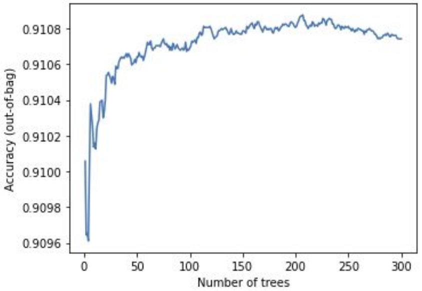
- Non-parametric decision forest model
- Ensemble of decision trees
- Draws subsamples from the training data to construct each tree in the forest
- Limited risk of overfitting

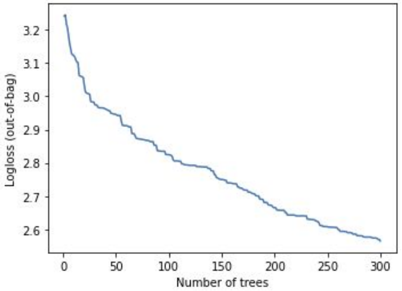
Available from “tensorflow_decision_forests” package

- Used top ranking hyper-parameter with reasonable performance and reduced max depth reduced from 16 to 8
- Run time was still close to 21 hours per model iteration
- Ran the simulation a various weights assigned to address label imbalance

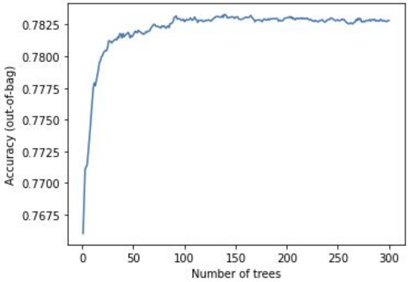
Random Forest Training Results

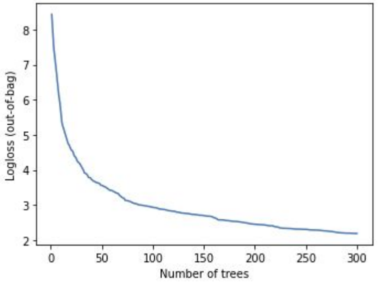
Unweighted



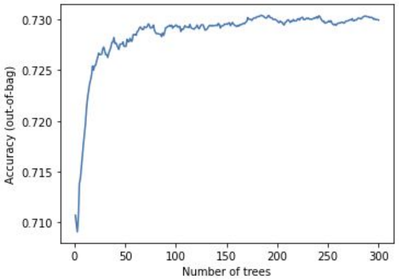


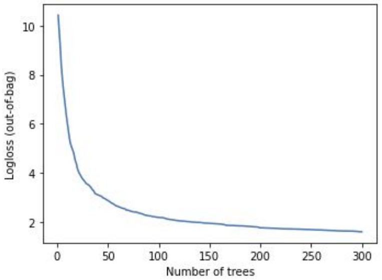
Half weights





Full weights





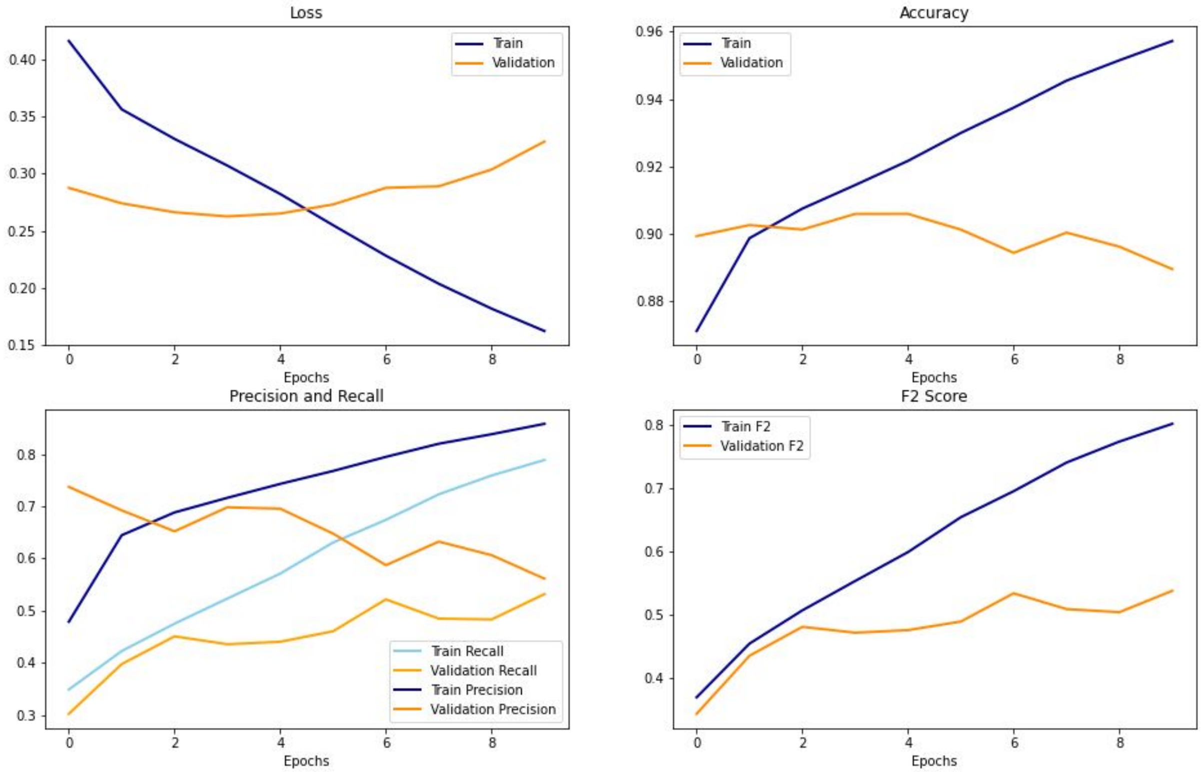
Approach 2: Random Forest Training Results

Metric	Unweighted	Weights Halved	Weighted
True Pos	1,524	2,057	4,167
False Pos	32	859	9,378
True Neg	34,307	33,480	33,547
False Neg	3,386	2,853	1,970
Accuray	91.3%	90.5%	76.9%
Precision	97.9%	70.5%	30.8%
Recall	31.0%	41.9%	67.9%
AUC	71.1%	81.4%	80.5%
PRC	53.1%	59.8%	56.1%
F1	47.2%	39.6%	22.0%
F2	29.5%	32.9%	26.5%

Approach 3: Neural Network

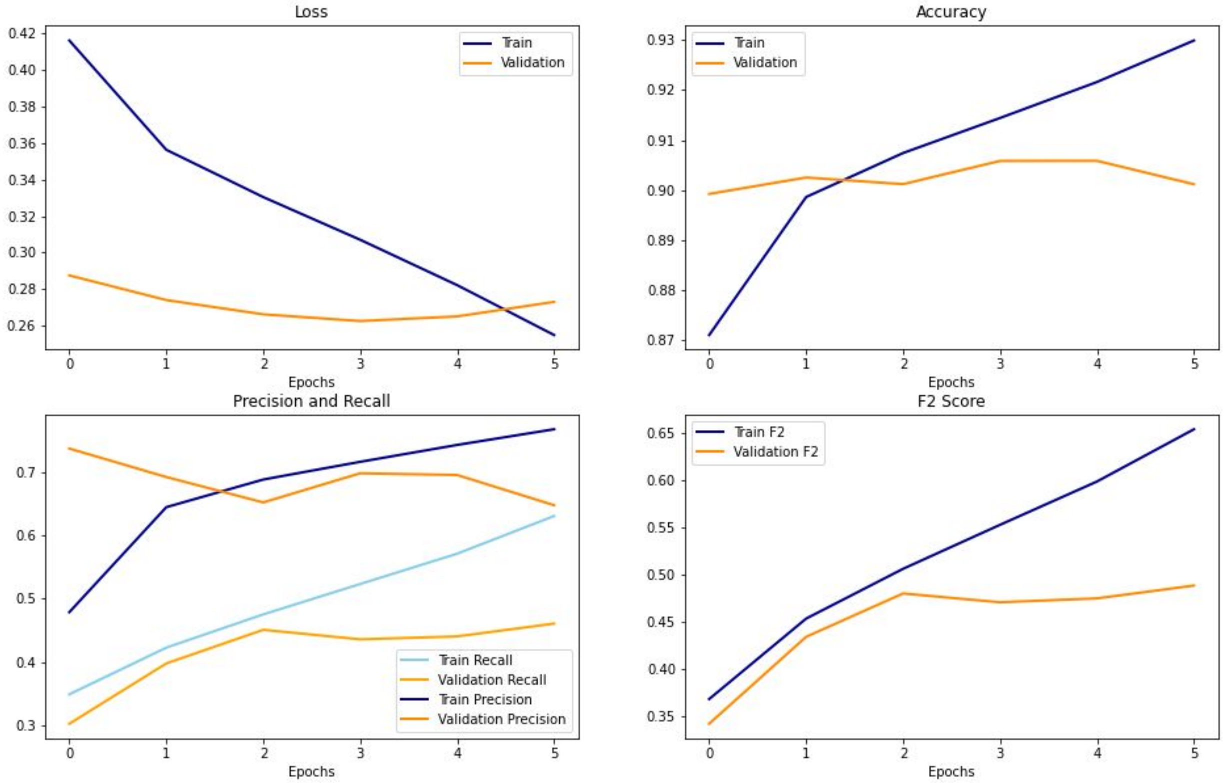
- Focused primarily on feedforward neural network
- Iterated through a number of hyperparameters
  - Ran ~4,000 model iterations
  - Hidden layer architecture: [[256,128,64], [256,128,32], [256,128,64,32], [128,64,32], [400,200,50], [400,200,100,50,25]]
  - Dropout rate: [0.25, 0.5, 0.75, 0.8]
  - Weight added to minority class: [None, 1.1, 1.2, 1.3, 1.4, 1.5]
  - Learning rate: [0.001, 0.0001]
  - Batch size: [32, 64, 128]
  - Activation: [tanh, relu, sigmoid]
  - Optimizer: [Adam, SGD]
  - Epochs: [10]

Neural Network Training: Recall-Optimized



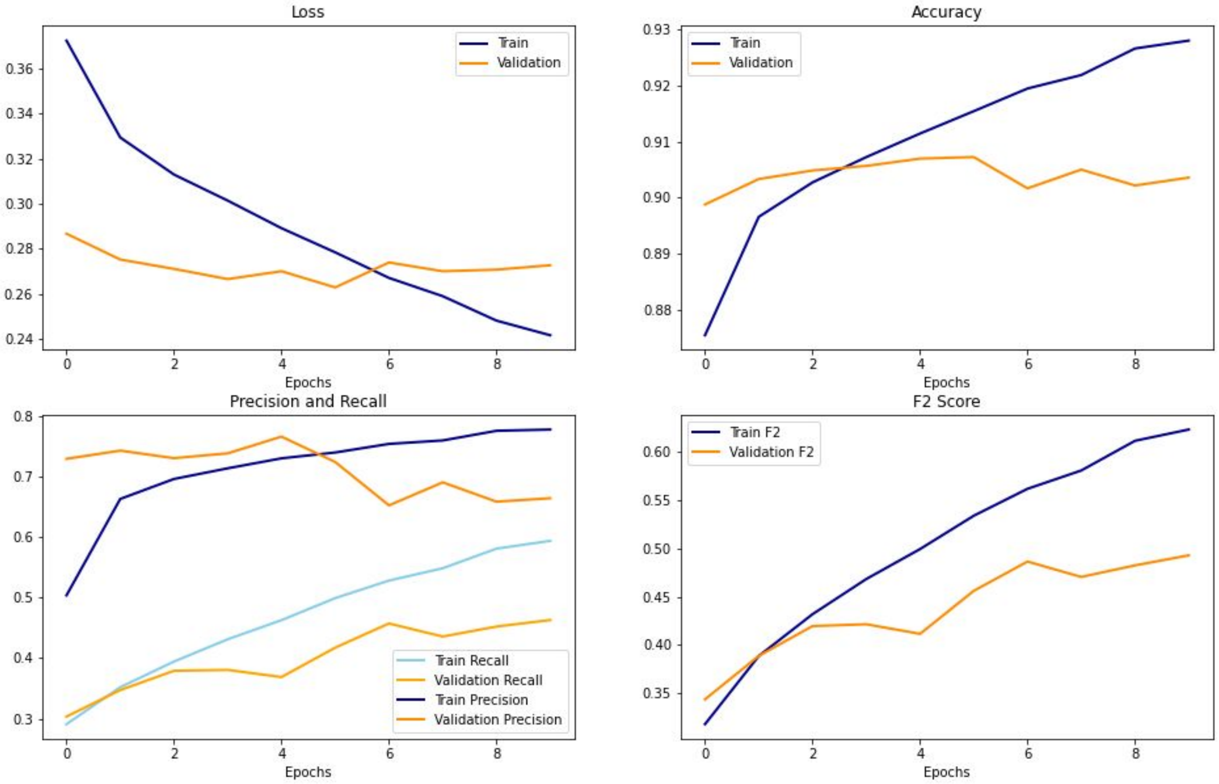
Hyperparameter	Value
Hidden Layer Sizes	[400,200,50]
Weighting	1.5
Learning Rate	0.001
Batch Size	128
Dropout Rate	0.5
Epochs	10
Activation	tanh
Optimizer	Adam

Neural Network Training: Recall-Optimized



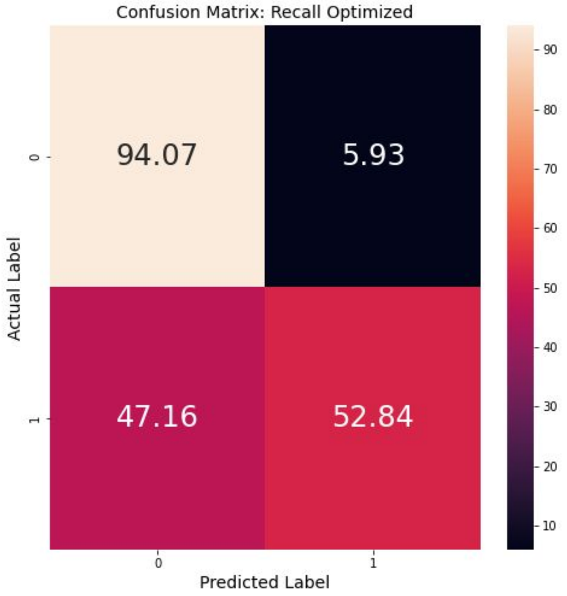
Hyperparameter	Value
Hidden Layer Sizes	[400,200,50]
Weighting	1.5
Learning Rate	0.001
Batch Size	128
Dropout Rate	0.5
Epochs	**6**
Activation	tanh
Optimizer	Adam

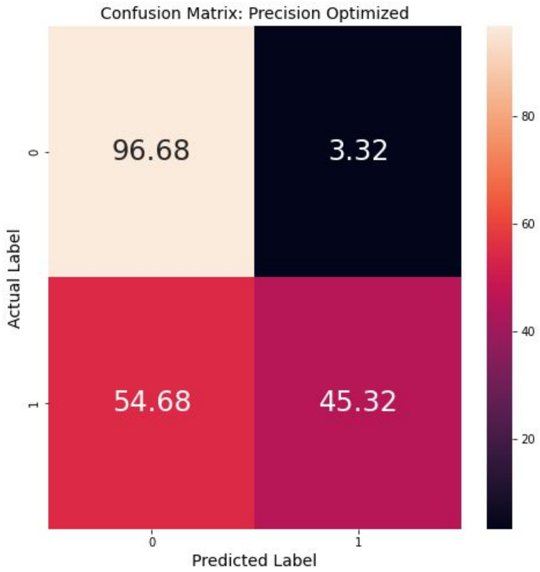
Neural Network Training: Precision-Optimized



Hyperparameter	Value
Hidden Layer Sizes	[400,200,50]
Weighting	1.2
Learning Rate	0.001
Batch Size	32
Dropout Rate	0.75
Epochs	10
Activation	tanh
Optimizer	Adam

Neural Network Test Results





Metric	Recall-Optimized	Precision-Optimized
True Pos	3243	2781
False Pos	2546	1423
True Neg	40379	41502
False Neg	2894	3356
Accuray	89.0%	90.0%
Precision	56.0%	66.0%
Recall	53.0%	45.0%
AUC	.83	.83
PRC	.58	.59
F1	54.0%	54.0%
F2	53.0%	48.0%
F.5	55.0%	61.0%

Neural Net Performance by Category

A: Human Necessities

B: Operations and Transport

C: Chemistry and Metallurgy

D: Textiles

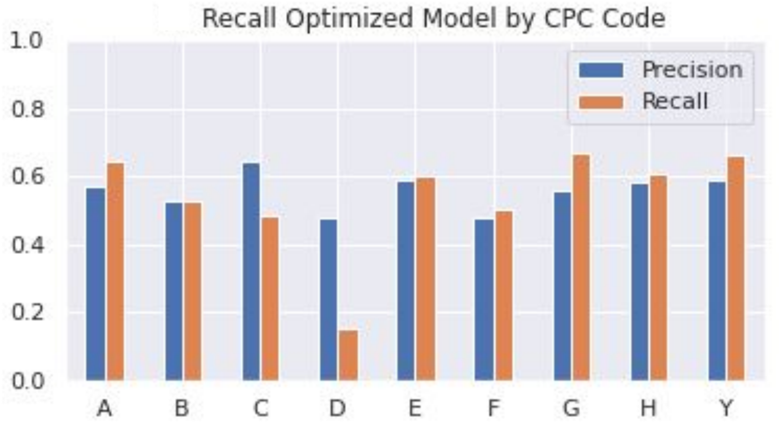
E: Fixed Constructions

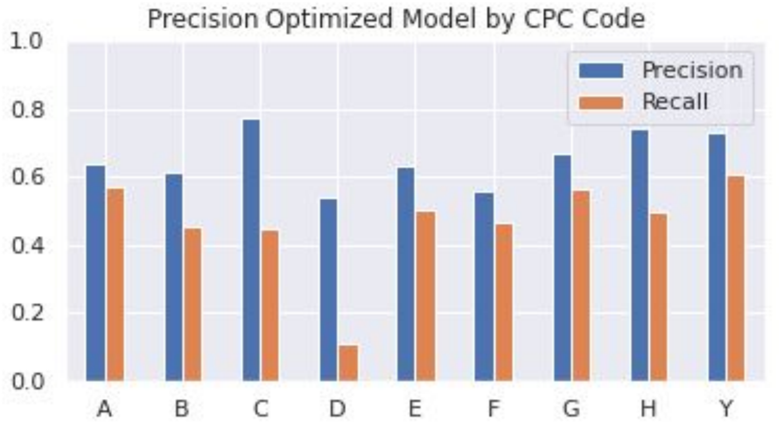
F: Mechanical Engineering

G: Physics

H: Electricity

Y: Emerging Cross-Sectional Technologies





Conclusions

- No single iteration of any model provides well balanced results
- Different models excel at different metrics
- Model selection depends on business case and level of risk/reward

Maximum observed metric by any model

Metric	Logistic Regression	Random Forest	Neural Network
Accuracy	90.0%	91.3%	90.0%
Precision	70.0%	97.9%	66.0%
Recall	69.2%	67.9%	53.0%
F1 score	24.3%	47.2%	54.0%

Future Focus

- BERT embeddings
  - Tuning
  - Patent trained models
- Dimensionality reduction: PCA, t-SNE, UMAP
- Threshold selection for Logistic Regression and Neural Networks
- K-fold validation
- Ensemble modeling
- Look into feature importances

Fin

Contributions

	Deanna	Deepak	Kolby	Matt
Preprocessing	✔	✔	✔	✔
BERT Embeddings	✔	✔	✔	✔
Logistic Regression		✔		
Random Forest				✔
Neural Network	✔		✔	
Presentation Slides	✔	✔	✔	✔