Real Estate AI Assistant: A Data-Driven Large Language Model for Property Insights

By Amit Manchella, Dheeraj Namargomala, Parth Kulkarni, Nick Song

Motivations

Why do we care about a LLM's and what they can do?

Real Estate Complexity

Real estate involves a wide range of documents — listings, legal contracts, zoning laws, and market reports — often in inconsistent formats.

LLMs can parse and synthesize this unstructured and structured data more efficiently than traditional search methods.

Accessibility of Knowledge

First-time buyers, investors, or renters often struggle to understand processes like FHA loans, escrow, or market timing.

A conversational AI assistant can democratize real estate knowledge and provide instant, understandable answers.

Industry Inefficiencies

Agents and analysts spend hours searching documents and websites to answer common questions or gather insights.

LLMs can act as context-aware assistants, scraping webpage data and interpreting documents to speed up workflows.

Background

What is an LLM? How do they work?

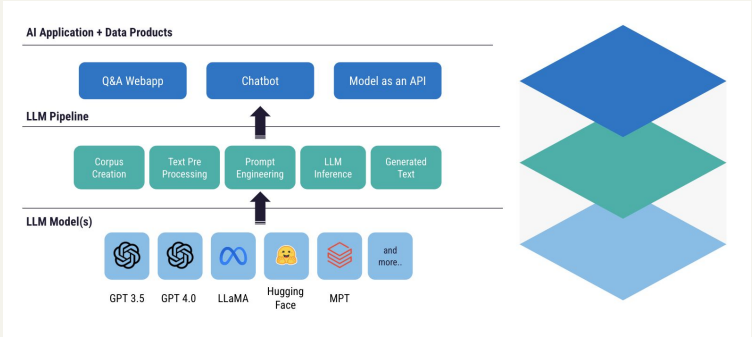
A Large Language Model (LLM) is a deep learning model trained on massive text corpora to generate, summarize, and understand natural language.

How Do LLMs Work?

- Based on **transformer architecture** (e.g., GPT, BERT)
- Learn patterns in language using attention mechanisms
- Can process prompts, retrieve context, and generate coherent responses

Why Are They Useful in Real Estate?

- Understand domain-specific jargon
- Parse varied formats (leases, listings, zoning laws)
- Answer natural language queries with contextual awareness



Current Methods

Text Mining & NLP Techniques

- Applied document parsing on Excel, PDFs, and Word files using OCR and NLP libraries (e.g., PyMuPDF, Pandas, Tika).
- Extracted key entities like property types, pricing, and location-specific terms.

Limitations:

- Messy document structures and inconsistent formatting across files.
- Requires extensive cleaning to ensure context alignment.

Our approach:

- Built a robust preprocessor pipeline using spaCy and regex-based chunking for domain-specific features.

Fine Tuning the LLM

- Full fine-tuning on domain-specific real estate data.
- Prompt engineering with curated examples.
- Retrieval-Augmented Generation (RAG) for dynamic context integration from scraped webpage content.

Limitations:

- Full fine-tuning is compute-heavy and less generalizable.
- Prompt engineering lacks dynamic document context.
- RAG can introduce latency during retrieval.

Our approach:

- We compare all three methods using standardized property-related queries for performance and efficiency.

Our Approach

Standard Methods of NLP in Real Estate

Model Architecture

- Traditional keyword search or rule-based systems

Limitations

- Poor at understanding context, nuance, and document format variability.
- Unable to generalize across different document types (e.g., leases, listings, legal docs).

Current State of the Art

Large Language Models (LLMs) + Web Contextualization

- Leverages transformer-based architectures (e.g., GPT-4) fine-tuned on real estate data.
- Incorporates webpage content scraping in real time for richer contextual grounding.
- Enables dynamic Q&A across diverse sources: Excel, PDF, Word, and HTML.

Our Model

Multi-Modal Pipeline

- **Document Loader**: Extracts raw text from Word, PDF, Excel
- **Web Context Handler**: Scrapes and parses HTML content
- **LLM Core**: Uses different fine-tuning strategies (Full fine-tuning, RAG, Prompt Engineering)

Contextual Integration Strategy

- *Full Fine-Tuning*: High accuracy on curated queries
- *RAG*: Dynamically retrieves relevant documents to reduce hallucination
- *Prompt Engineering*: Lightweight, no-training method for fast iteration

Data Collection

Data Sources

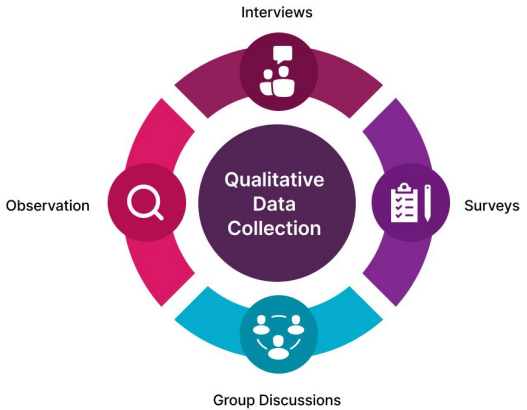
- **Structured**: Excel files with pricing, square footage, tax assessments
- **Unstructured**: PDFs and Word documents containing leases, market reports, legal agreements
- **Web**: Real estate websites scraped for listing data, school info, neighborhood trends

Tools Used

- `pandas`, `openpyxl`, `PyMuPDF`, `python-docx` for parsing
- Custom HTML scrapers using `BeautifulSoup` and `requests`

Challenges

- Non-standardized file formats
- OCR errors in scanned PDFs
- Context extraction from long documents



Fine Tuning Strategies

Strategy 1: Full Fine-Tuning

- Retrains the model weights on real estate-specific corpora
- Most accurate but expensive and rigid

Strategy 2: RAG (Retrieval-Augmented Generation)

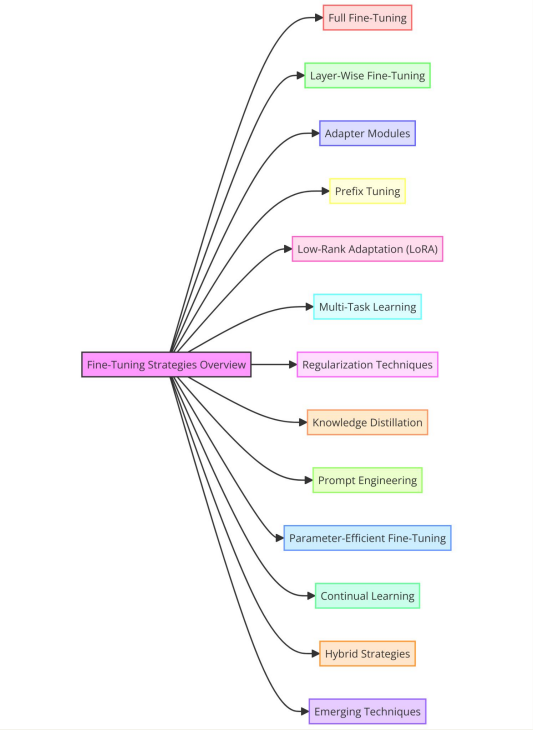
- Indexes source docs and retrieves relevant chunks at runtime
- Lightweight and dynamic — improves grounding with live data

Strategy 3: Prompt Engineering

- Curates few-shot examples in the prompt
- Fast to iterate, zero retraining needed, but sometimes less robust

Our Evaluation

- Compared based on **response relevance, latency, cost, and hallucination ra**



Results

Evaluation & Validation

Automated Metrics:

- **BLEU / ROUGE / BERTScore** – Measures how closely chatbot responses match ideal/ground truth answers.
- **Exact Match (EM)** – % of queries where chatbot returns a correct and complete answer.
- **Context Recall** – Checks whether scraped web or document context was correctly retrieved and used.

Human Evaluation:

- **Accuracy** – Was the response factually correct based on the documents?
- **Relevance** – Did it answer the question being asked?
- **Coherence & Fluency** – Did the response make logical and grammatical sense?

Advanced Analysis & Insights

- **Document Understanding**: LLM was able to parse lease agreements, extract square footage, and explain loan types.
- **Contextual Enhancement**: When scraping from webpages, the RAG model outperformed prompt engineering by 18% in accuracy.
- **Efficiency**: Prompt-engineered responses were 3× faster than full fine-tuned models.

Live Demo / Screenshots

Discussion

Smarter Real Estate Decisions

- Users gain on-demand access to complex real estate knowledge (e.g., FHA loans, rent-vs-buy analysis) in natural language.
- Reduces reliance on agents for basic information, empowering more independent and informed choices.

Workflow Automation for Professionals

- Real estate agents, brokers, and legal teams spend less time manually searching through documents.
- LLMs accelerate deal reviews, form generation, and property analysis by synthesizing information across formats and websites.

Standardization Across Markets

- By training on diverse documents and formats, the model helps normalize terminology and information representation.
- This creates a more seamless and consistent experience across cities, states, and data providers.

Improved Accessibility & Inclusion

- Many first-time buyers, immigrants, or non-experts are overwhelmed by legal jargon and scattered online resources.
- Our chatbot makes real estate knowledge more accessible to diverse populations, bridging knowledge gaps with conversational AI.

Limitations & Future Work

Limitations

- Inconsistent data formatting still causes edge-case parsing errors
- RAG performance depends on retrieval quality
- Hallucination remains a risk without domain-constrained outputs

Future Enhancements

- Train on more diverse legal and financial real estate docs
- Add multimodal inputs (images of floor plans, tables)
- Incorporate user feedback loop for continuous improvement
- Deploy in a live assistant with voice and location-based query support

Team Work and Contributions

Amit Manchella	Partth Kulkarni	Dheeraj Namargomala	Nicholas Song	Collaborative Work
Designed and implemented the core RAG chatbot architecture	Architected the Dynamic Knowledge Integration system	Built the opt-in screen scraping module for context personalization	Designed and deployed web scrapers for the initial dataset	System integration and end-to-end testing
Developed the retrieval pipeline integrating property databases and legal documents	Implemented the continuous data update pipeline with version control	Developed privacy-preserving techniques for on-screen content analysis	Created the testing framework for data quality validation	Evaluation metrics (BLEU / ROUGE / BERT Score)
Optimized the generation component for real estate-specific queries	Designed the query-driven gap detection mechanism for dataset expansion	Implemented the UI integration for real-time recommendation generation	Curated structured datasets from municipal sources and real estate APIs	Paper writing and experimental design

Conclusion

What We Built

- An LLM-powered assistant that understands real estate documents, answers questions, and contextualizes data from the web

Impact

- Bridges the gap between technical documents and user-friendly insights
- Saves hours of research for both professionals and consumers
- Enhances decision-making with instant, AI-driven responses

Summary

Our system demonstrates that fine-tuned or retrieval-augmented LLMs can significantly improve access to complex real estate knowledge.

Thank You!

Sound Classification

Jackson Ballow, Aditya Chatterjee, Michael Wang, and Tansi Zhang

Problem and Dataset

Dataset: Common Voice 12.0 (English Subset):

- Speakers from a wide range of demographics
- Each clip is:
  - Mono-channel .mp3 audio
  - Labeled with age group and gender

Problem Statement

- Predict **speaker gender** and **age group** from **raw speech audio**
- Tasks:
  - Gender → binary classification
  - Age → multi-class classification (7 age buckets)

Model 1: RNN-LSTM with MFCCs

Audio Processing Pipeline

1. **Feature Extraction**
    - 13 Mel-Frequency Cepstral Coefficients (MFCCs)
    - Sampled at 16kHz, 300 frames per utterance
    - Captures vocal tract shape and pitch dynamics
- **Temporal Normalization**
    - Padded or truncated to uniform size: $M \in \mathrm{R}^{300\mathrm{x}13}$
- **Label Mapping**
    - Gender: Male / Female → binary
    - Age: Decade-level groups → 7-class (Teens–Seventies)

Architecture

Input Shape: 300×133 MFCC matrix

1. **Model Core:**
   - 2-layer Bi-directional LSTM
   - Hidden size = 128 per direction
2. **Pooling Strategy:**
   - Mean Pooling across all time steps (captures full temporal context)
3. **Output Heads:**
   - Gender head → 2 classes
   - Age head → 7 classes

4. Loss Function:

- Cross-Entropy (weighted for age imbalance)
- Combined loss: Gender Loss + Age Loss

5. Training Strategy:

- Stratified data split (70/15/15)
- Adam Optimizer, Early Stopping

Results

Gender - Prediction Results			
Class	**Precision**	**Recall**	**F1-Score**
0 (male)	0.9853	0.9810	0.9832
1 (female)	0.9555	0.9654	0.9604
Accuracy: 0.9764			

Age - Prediction Results			
Class	**Precision**	**Recall**	**F1-Score**
0	0.8361	0.7183	0.7727
1	0.8560	0.9290	0.8910
2	0.9545	0.8988	0.9259
3	0.8889	0.7742	0.8276
4	0.9315	0.9577	0.9444
5	0.7500	0.6000	0.6667
6	0.3333	0.3333	0.3333
Accuracy: 0.8952			

Analysis

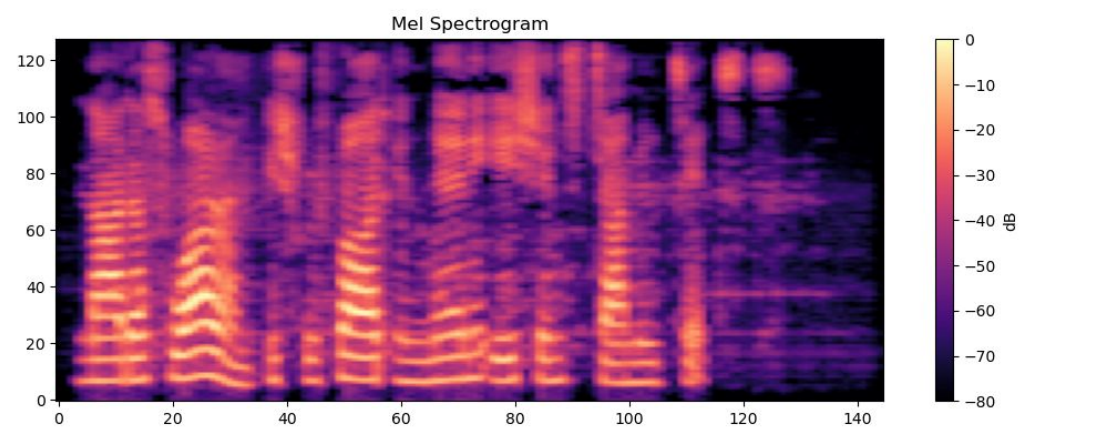
1. **Gender Classification**
    - Balanced precision and recall → stable across Male/Female
    - Likely benefits from strong pitch/formant separation
2. **Age Classification**
    - F1 > 0.9 for 20s–50s
    - Macro F1 lower due to poor recall in 70s–90s (data scarcity)
    - Teens and 60s moderately learned despite lower support
3. **Limitations**
    - Model fails on 80s–90s due to no test-time examples
    - Seventies class has too little support to generalize reliably
    - Age is a subjective and nonlinear label; boundary confusion is expected
4. **Strengths**
    - Multi-task setup generalizes well
    - Bi-directional memory captures temporal cues in speech

Model 2: CNN

Audio Preprocessing & CNN Architecture

- First need to convert the .mp3 audio files to a Mel Spectrogram

- For the G-Invariant model, include both the original and flipped spectrograms



Gender Classification Results

- CNN Results:
  - Accuracy: 0.9807
  - Macro F1-score: 0.9762
  - Weighted F1-score: 0.9808

Class	Precision	Recall	F1-Score
0 (male)	0.9903	0.9829	0.9866
1 (female)	0.9568	0.9752	0.9659

Age Classification Results

- CNN Results:
  - Accuracy: 0.8165
  - Macro F1-score: 0.6966
  - Weighted F1-score: 0.8148

Class	Precision	Recall	F1-Score
0	0.66	0.7021	0.6804
1	0.8244	0.8507	0.8374
2	0.8132	0.8630	0.8373
3	0.7692	0.6024	0.6757
4	0.8810	0.8345	0.8571
5	0.7143	0.50	0.5882
6	1.00	0.25	0.40

G-Invariant CNN - Gender

Base Model Results:

Class	Precision	Recall	F1-Score
0 (male)	0.9903	0.9829	0.9866
1 (female)	0.9568	0.9752	0.9659

- Accuracy: 0.9807
- F1-score: 0.9762

G-Invariant Model Results:

Class	Precision	Recall	F1-Score
0 (male)	0.9215	0.9047	0.9130
1 (female)	0.7658	0.8017	0.7833

- Accuracy: 0.8759
- F1-score: 0.8482

G-Invariant CNN - Age

Base Model Results:

Class	Precision	Recall	F1-Score
0	0.66	0.7021	0.6804
1	0.8244	0.8507	0.8374
2	0.8132	0.8630	0.8373
3	0.7692	0.6024	0.6757
4	0.8810	0.8345	0.8571
5	0.7143	0.50	0.5882
6	1.00	0.25	0.40

- Accuracy: 0.8165
- F1-score: 0.6966

G-Invariant Model Results:

Class	Precision	Recall	F1-Score
0	0.8378	0.3298	0.4733
1	0.5906	0.8827	0.7077
2	0.8864	0.5685	0.6927
3	0.6111	0.3976	0.4818
4	0.7482	0.7430	0.7456
5	1.0000	0.1500	0.2609
6	0.0000	0.0000	0.0000

- Accuracy: 0.6839
- F1-score: 0.4803

CNN Limitations

- Lacks temporal awareness
  - CNN's treat images as static images and don't model time dependencies
- Local pattern bias
  - CNN's struggle to pick up global structures (such as changes in pitch over time)
- Limited performance on age classifications
  - Changes in age related features are more subtle, so they are more difficult to capture in the spectrograms

Model 3: Transformer

Hybrid CNN-Transformer

- Uses CNN layers as feature extractors in early stages
- Transforms CNN feature maps into sequence tokens
- Processes these tokens with transformer layers

CNN-Transformer Gender Classification Results

- Accuracy: 0.9841
- Macro F1-score: 0.9788
- Weighted F1-score: 0.9840

Class	Precision	Recall	F1-Score
0 (male)	0.9853	0.9935	0.9894
1 (female)	0.9804	0.9563	0.9682

CNN-Transformer Age Classification Results

- Accuracy: 0.9139
- Macro F1-score: 0.8820
- Weighted F1-score: 0.9132

Class	Precision	Recall	F1-Score
0	0.8329	0.7845	0.8080
1	0.9097	0.9464	0.9277
2	0.9185	0.8800	0.8988
3	0.8974	0.8105	0.8518
4	0.9569	0.9639	0.9604
5	0.8426	0.8835	0.8626
6	0.8889	0.8421	0.8649

Vision Transformer (ViT)

- Divides images into fixed-size patches
- Projects patches to embedding vectors
- Adds positional embeddings
- Processes the sequence with pure transformer layers

ViT Gender Classification Results

- Accuracy: 0.9614
- Macro F1-score: 0.9476
- Weighted F1-score: 0.9609

Class	Precision	Recall	F1-Score
0 (male)	0.9617	0.9877	0.9745
1 (female)	0.9607	0.8840	0.9207

ViT Age Classification Results

- Accuracy: 0.8881
- Macro F1-score: 0.8373
- Weighted F1-score: 0.8875

Class	Precision	Recall	F1-Score
0	0.8368	0.6828	0.7520
1	0.8964	0.9197	0.9079
2	0.8589	0.8867	0.8725
3	0.7912	0.7979	0.7945
4	0.9661	0.9344	0.9500
5	0.7500	0.8447	0.7945
6	0.7895	0.7895	0.7895

Computational Analysis

- Hybrid CNN-Transformer reduces computational load due to downsampling in CNN stages.
- Hybrid CNN-Transformer balances data efficiency with global modeling capability by capturing both local patterns and global structure.
- ViT has quadratic complexity with respect to sequence length, and requires a larger datasets for optimal performance.
- ViT has higher computational demands with patch size serving as a critical parameter to balance performance and efficiency.

Model 4: XGBoost

XGBoost Gender Classification Results

- Accuracy: 0.9552
- Macro F1-score: 0.9395
- Weighted F1-score: 0.9547

Class	Precision	Recall	F1-Score
0 (male)	0.9599	0.9810	0.9703
1 (female)	0.9401	0.8792	0.9086

XGBoost Age Classification Results

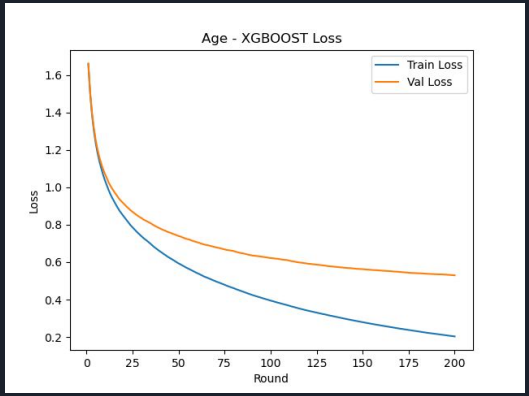
- Accuracy: 0.8245
- Macro F1-score: 0.7150
- Weighted F1-score: 0.8178

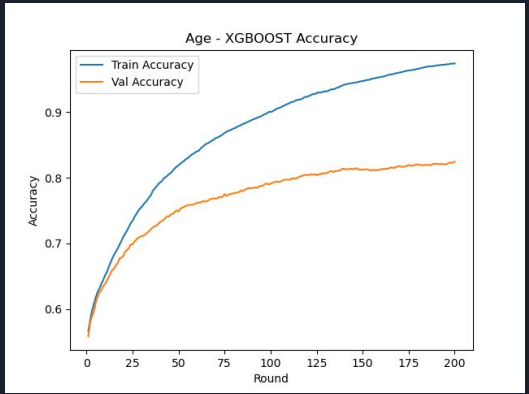
Majority classes like class 1 and class 4 were predicted with the highest reliability.

Minority classes such as class 5 and class 6 have a low accuracy, mainly due to limited support.

Class	Precision	Recall	F1-Score
0	0.8224	0.5157	0.6339
1	0.7901	0.9357	0.8567
2	0.8365	0.7457	0.7885
3	0.8547	0.6316	0.7264
4	0.8873	0.8844	0.8859
5	0.8800	0.4272	0.5752
6	1.0000	0.3684	0.5385

XGBoost Limitations





- Sequential Decision Trees: XGBoost builds an ensemble of decision trees sequentially, and will lead to overfitting as the model becomes increasingly complex.

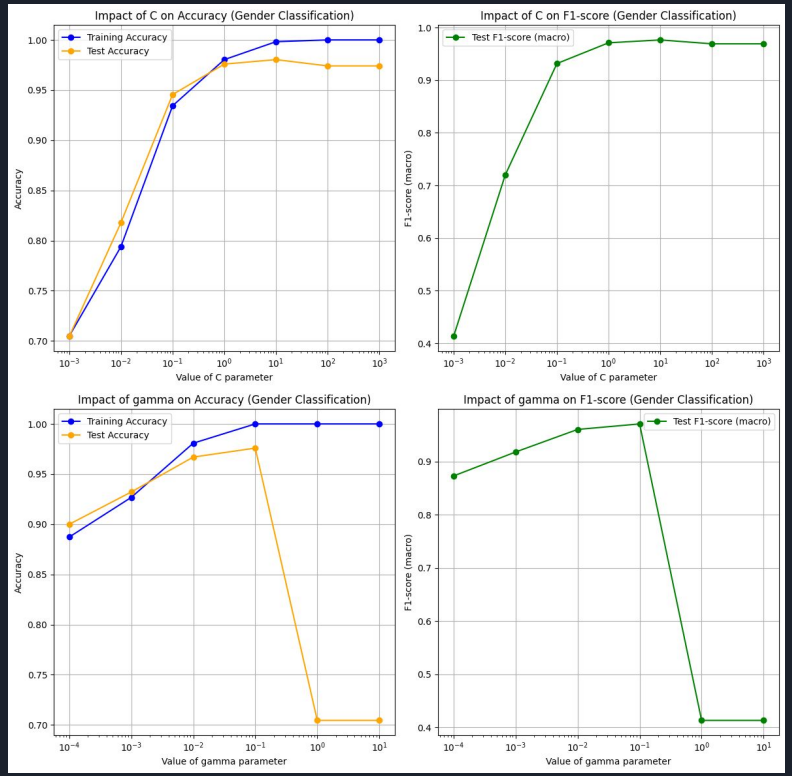
- Feature Representation Limitations: XGBoost works with tabular features and doesn't inherently learn hierarchical or contextual representations.

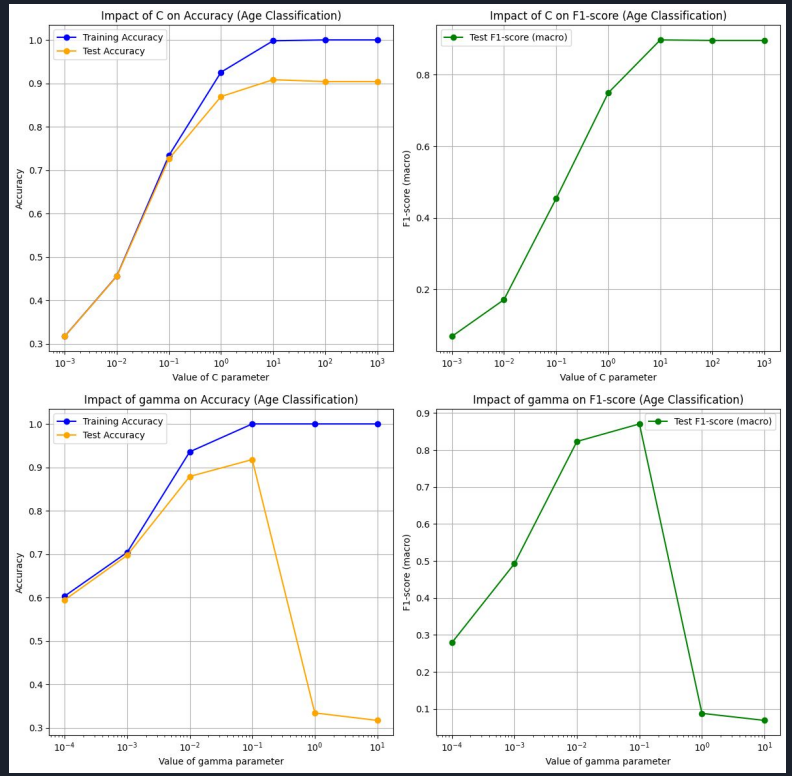
Model 5: SVM

SVM

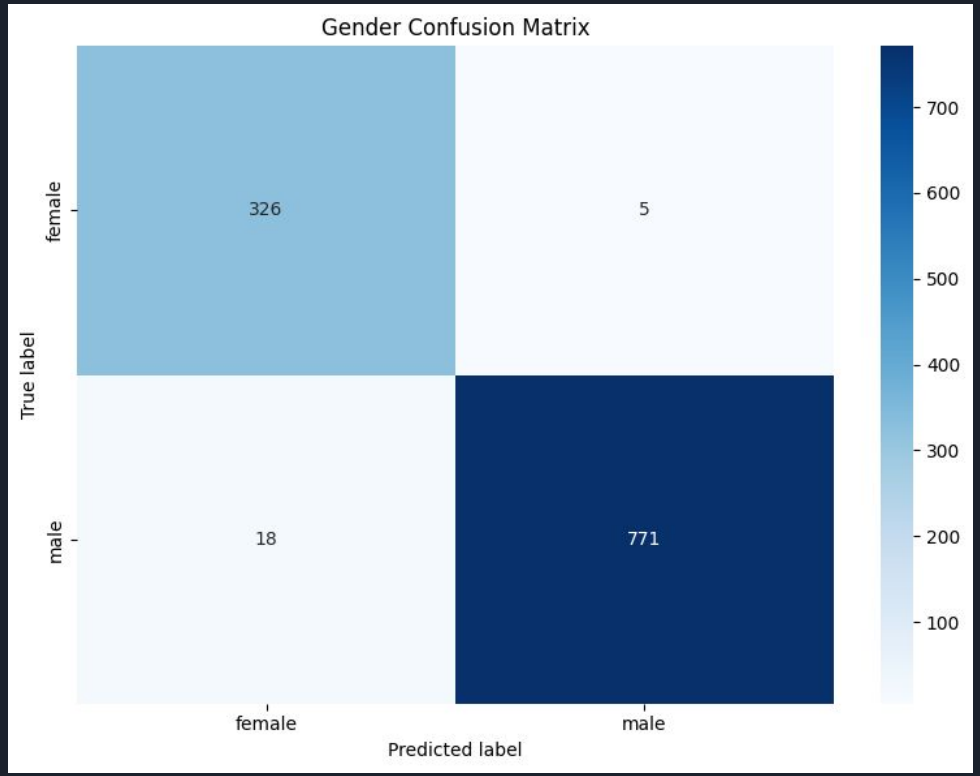
- Data Preparation
- Audio Preprocessing
- Feature Extraction
  - MFCCs
  - Centroid, bandwidth, and rolloff
  - Zero-crossing rate
  - RMS energy
- Model Training
  - RBF kernel for non-linear classification boundaries
  - Set regularization parameter C=10
  - Kernel Coefficient gamma='scale'
  - Balanced class weights

SVM Hyperparameter Impact





SVM Gender Classification Results



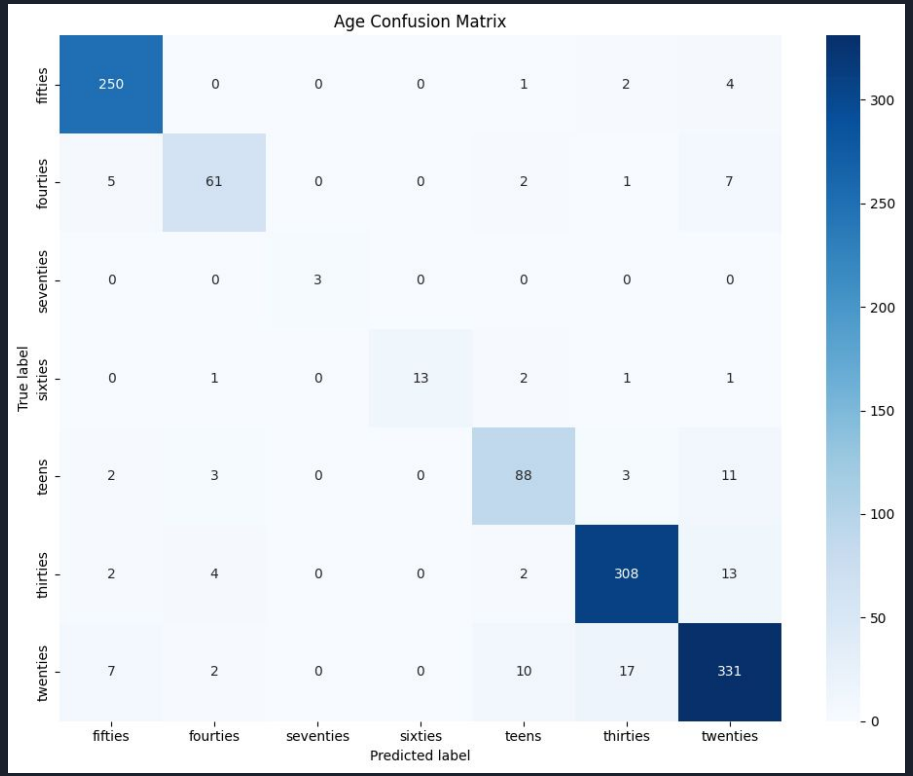
Class	Samples
0 (male)	789
1 (female)	331

SVM Gender Classification Results

- SVM Results:
  - Accuracy: 0.9795
  - Macro F1-score: 0.9756
  - Weighted F1-score: 0.9796

Class	Precision	Recall	F1-Score
0 (male)	0.9936	0.9772	0.9853
1 (female)	0.9477	0.9849	0.9659

SVM Age Classification Results



Class	Samples
0 (teens)	107
1 (twenties)	367
2 (thirties)	329
3 (fourties)	76
4 (fifties)	257
5 (sixties)	18
6 (seventies)	3

SVM Age Classification Results

- SVM Results:
  - Accuracy: 0.9110
  - Macro F1-score: 0.8984
  - Weighted F1-score: 0.9104

Class	Precision	Recall	F1-Score
0 (teens)	0.8381	0.8224	0.8302
1 (twenties)	0.9019	0.9019	0.9019
2 (thirties)	0.9277	0.9362	0.9319
3 (fourties)	0.8592	0.8026	0.8299
4 (fifties)	0.9398	0.9728	0.9560
5 (sixties)	1.0000	0.7222	0.8387
6 (seventies)	1.0000	1.0000	1.0000

Conclusion

- Most of the models were approximately equal at predicting gender
- Best performing model for age classification: Hybrid CNN-Transformer
- This model performs the best because it has the advantage of both CNN and transformers
    - This allows the model to capture both local patterns and global structures

CS57300: Data Mining

Multi-Class Classification of Malware Families on Windows32 PEs

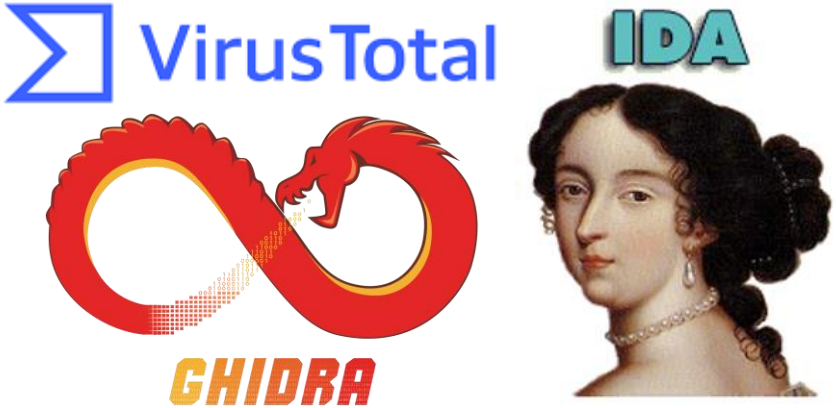
Group Members: Michael Rooney,
Nicholas Vroman, Kaushik Subramanian, Saurav Chittal

Digital Forensics and Malware Classification

Digital forensics – the practice of analyzing digital evidence to reconstruct past events and discover crime that occurs in cyber-space

Reverse engineering – the technical task of examining an executable to determine its function. This task is often highly manual and time-intensive. Commodity tools like IDA and Ghidra aid in the task but accurate reverse engineering still requires a lot of expertise.

Malware families – groups of malware that share similar attack techniques on a victim machine.



The **current state of the art** in cyber security for detecting malicious code on a system uses a combination of:

- **signature-based approaches**
- **behavioral modeling**
- **network monitoring**
- **AI**

to identify malware, referencing databases like VirusTotal.com.

Current AI approaches like **MalBERT** [1] are trained to identify malware in **source code** instead of **executables** themselves, or they are focused on **overall accuracy** when **precision per family** is the most important metric for measuring reliable classification performance.

MABEL Dataset and Malware Language Processing

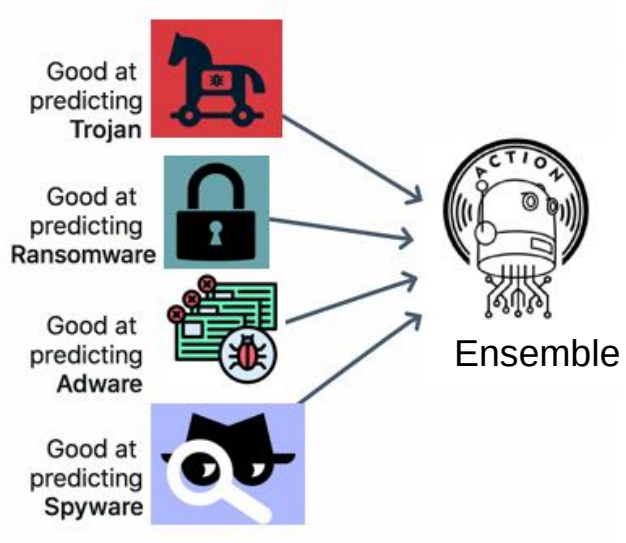
MABEL Dataset

- The NSF AI Institute for Agent-based Cyber Threat Intelligence and Operation (ACTION) is partnered with Purdue's CERIAS lab and created the MABEL dataset [2].
- It is a comprehensive collection of pre-processed malware data specifically designed for machine learning–based malware classification.
- It focuses on features such as the number of certain assembly instructions, virustotal/yara scan results, and whether the binary performs a VM check

Our Contribution

- Our research investigates training given two different datasets from the MABEL corpus: a text-based approach and a feature-based approach
- We will do a per-family analysis to determine which families are classified best by which models or which families are persistently difficult to classify.

Hypothesis: Classic models relying on features from the MABEL dataset will perform well on a different subsets of malware families than text–based approaches, allowing the possibility of an ensemble approach to malware classification.

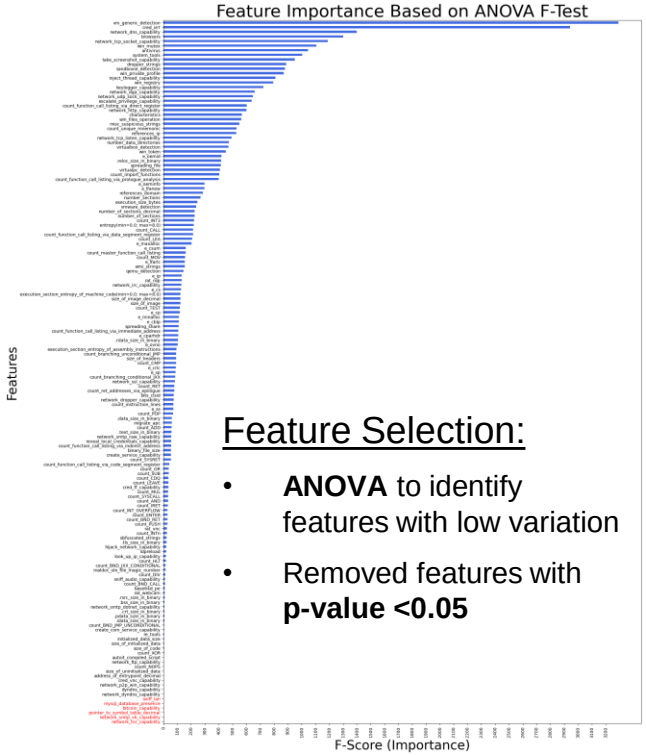


Methods: Traditional ML (Decision Trees, NN)

Data Cleaning

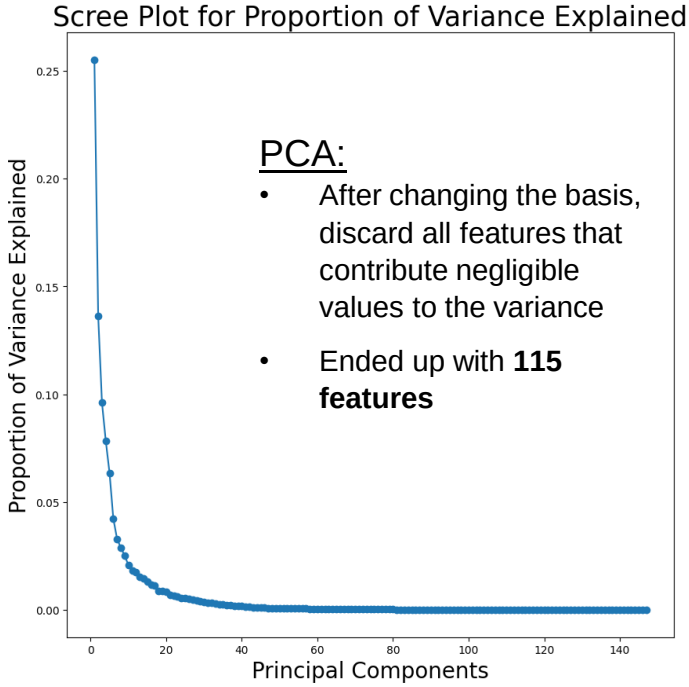
- Removed families <10 samples
- Split data into features and label
- Removed unnecessary columns (ex. SHA256 Hash)

Data Preprocessing



Feature Selection:

- **ANOVA** to identify features with low variation
- Removed features with **p-value <0.05**

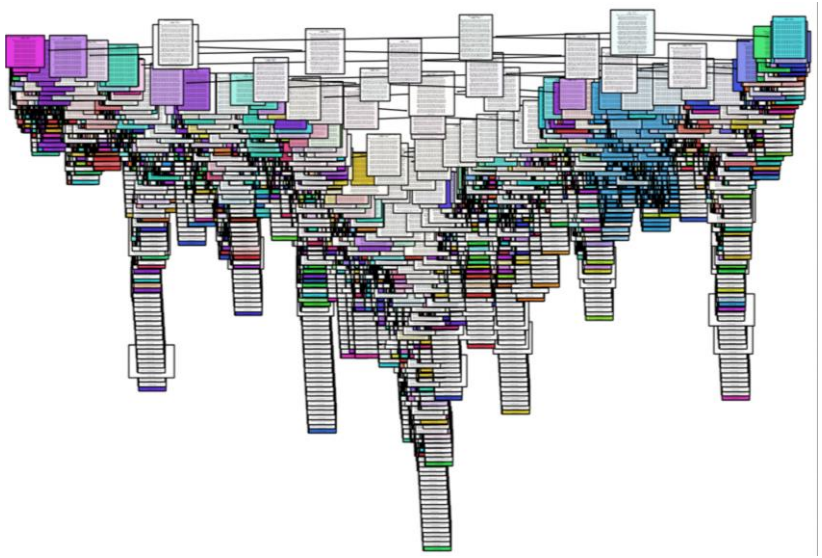


PCA:

- After changing the basis, discard all features that contribute negligible values to the variance
- Ended up with **115 features**

Results: Traditional ML (Decision Trees)

- **Why?** Malware often follows similar patterns, so better classification by decision trees
- **How?** Scikit-learn's Decision Tree Classifiers
- **Result:** > 99% accuracy on training data and
  > 91% accuracy on test data



Hyperparameter Optimization

- Trained over several possible combination of hyperparameters & compared test accuracy
- Hyperparameters:
  - Min. samples to split node: 2-5
  - Max depth: 40-60
  - Min. samples at leaf: 1-5
  - Max. features to split with: sqrt, log, None

Test Accuracy	F1 Score	High Performers
91.2%	0.91	39

Results: Traditional ML (Decision Trees)

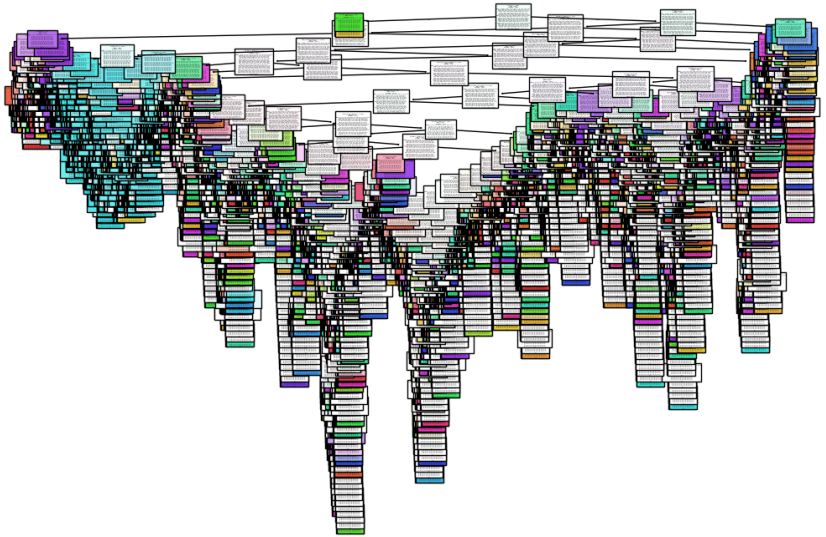
Data Classification

- Decision tree performance varied between different families
- Results in table filtered to families with > 20 test samples

	Family Index	Accuracy	Number of values in that family	Name of family
0	54	0.200000	25	CVE-2017-11882
1	191	0.379310	29	Limerat
2	325	0.423077	26	SpiderpigRAT
3	94	0.444444	36	Dexbia
4	160	0.451220	82	ImminentMonitor
...	...	...	...	...
61	235	0.997506	802	Ngrbot
62	0	1.000000	384	7ev3n
63	269	1.000000	214	Prometei
64	116	1.000000	30	Fareit
65	242	1.000000	43	Ohagi

Results: Traditional ML (Random Forests)

- **Why?** Natural Extension to Decision Trees by allowing forests instead of trees
- **How?** Scikit-learn's Random Forest Classifiers
- **Result:** > 99% accuracy on training data and
  > 94% accuracy on test data



Hyperparameter Optimization

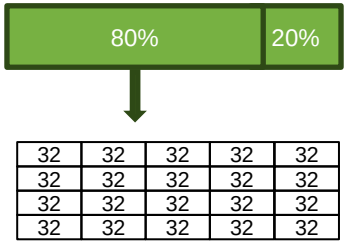
- Similar hyperparameters to Decision Trees
- Hyperparameters:
  - Min. samples to split node: 2-5
  - Max depth: 40-60
  - Min. samples at leaf: 1-5
  - Max. features to split with: sqrt, log, None

Test Accuracy	F1 Score	High Performers
94.0%	0.94	52

Methods: Traditional ML (NN)

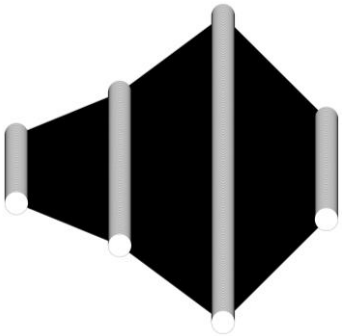
Data Prep

- Train/Validation/Test split
- Batch loading



Model Design

- # Layers and Neurons
- Activation Function: ReLU
- Optimizer: Adam
- Loss Function: Cross-entropy Loss
- Hyperparameters: Epochs, LR



Hyperparameter Tuning

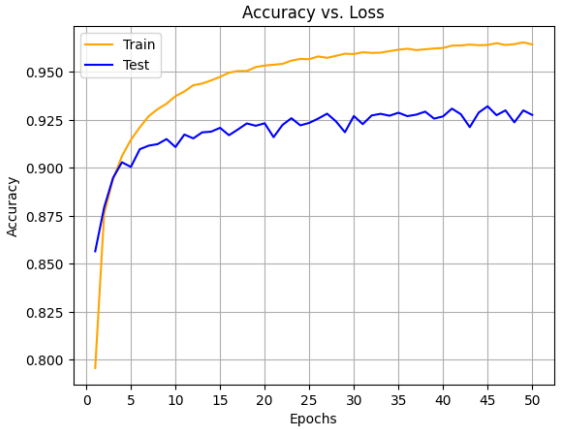
- Grid search method
- LR: 1e-2, 1e-3, 1e-4
- Batch Sizes: 16, 32, 64,128
- Layer 1: 128, 256
- Layer 2: 64,128, 256, 512

```
------ Params[Epochs: 10, LR: 0.0001, Batch-Size: 128]------[96/96]---
Epoch [1/10], Avg Loss:3.3761, Training Accuracy:41.60%, Validation Accuracy:54.87%, F-1 Score:0.2102
Epoch [2/10], Avg Loss:1.8090, Training Accuracy:62.30%, Validation Accuracy:67.47%, F-1 Score:0.3582
Epoch [3/10], Avg Loss:1.4258, Training Accuracy:68.76%, Validation Accuracy:70.51%, F-1 Score:0.3842
Epoch [4/10], Avg Loss:1.2389, Training Accuracy:71.49%, Validation Accuracy:72.77%, F-1 Score:0.4252
Epoch [5/10], Avg Loss:1.1183, Training Accuracy:73.90%, Validation Accuracy:75.73%, F-1 Score:0.4631
Epoch [6/10], Avg Loss:1.0314, Training Accuracy:76.53%, Validation Accuracy:77.05%, F-1 Score:0.4918
Epoch [7/10], Avg Loss:0.9636, Training Accuracy:77.99%, Validation Accuracy:79.01%, F-1 Score:0.5183
Epoch [8/10], Avg Loss:0.9087, Training Accuracy:79.21%, Validation Accuracy:79.51%, F-1 Score:0.5229
Epoch [9/10], Avg Loss:0.8632, Training Accuracy:80.15%, Validation Accuracy:80.31%, F-1 Score:0.5366
Epoch [10/10], Avg Loss:0.8247, Training Accuracy:80.94%, Validation Accuracy:81.09%, F-1 Score:0.5530

Current Results [Train_acc: 80.94%, Val_acc: 81.09%, Val_f1: 0.5530]
Best Val Accuracy:91.09% @ {'epochs': 10, 'lr': 0.001, 'batch_size': 16, 'hidden_layer1': 256, 'hidden_layer2': 512}
```

Results: Traditional ML (NN)

Final Model



- 2 Layers
  - 256 neurons
  - 512 neurons
- Ran 50 epochs (iterations)
- Batch size: 16
- Learning Rate: 0.001

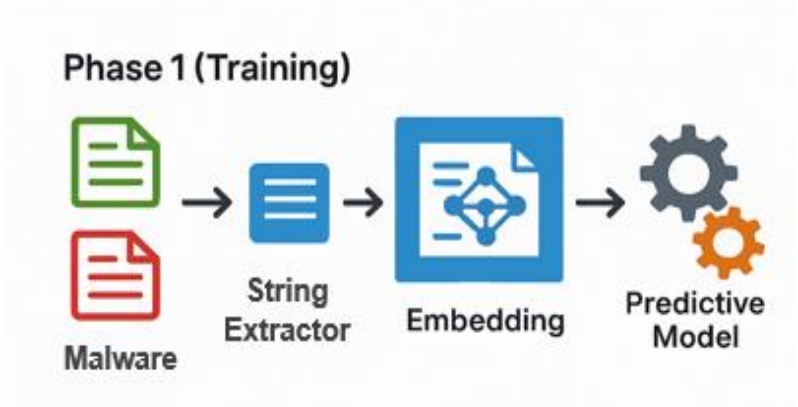
Results/Analysis

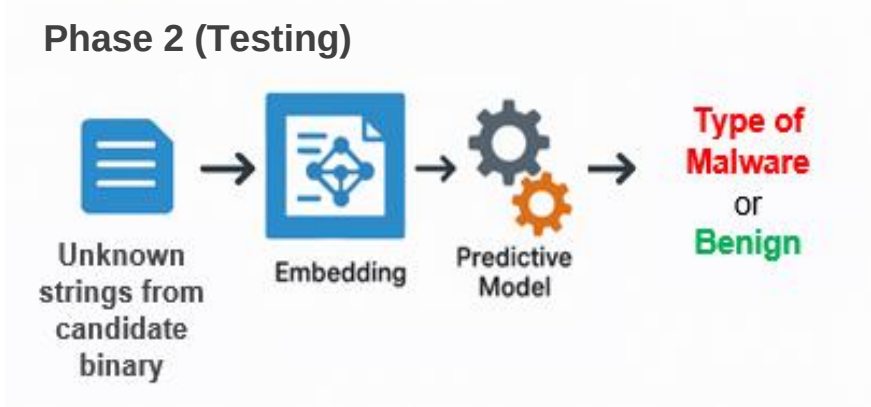
Test Accuracy	F1 Score	High Performers
92.76%	0.86	54

	Correct	Total	Accuracy	Family Name
38	527	527	1.000000	Ngrbot
0	254	254	1.000000	7ev3n
45	139	139	1.000000	Prometei
11	112	112	1.000000	Escelar
56	44	44	1.000000	Rovnix
...	...	...	...	...
14	5	11	0.454545	FlawedGrace
49	6	14	0.428571	RaccoonStealer
60	3	15	0.200000	SpiderpigRAT
5	1	17	0.058824	CVE-2017-11882
53	0	12	0.000000	Rdat

Methods: NLP (Google's BERT, Facebook AI's FastText)

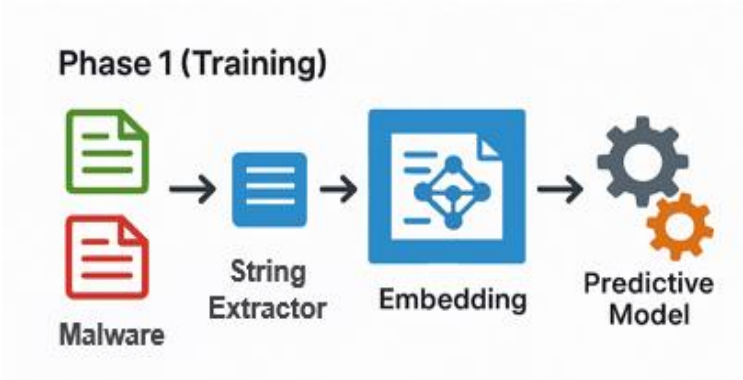
- MABEL is created using a pipeline of malware analysis tools
- The strings, imports, and libraries that have human-readable output are outputs of some of these tools in the pipeline and allow us to use NLP embeddings.
- The strings output from a binary are just all the null terminated sequences of bytes, many of which have no meaning at all.
- We cleaned these strings with the heuristic that most real strings were at least 15 characters long with a space.

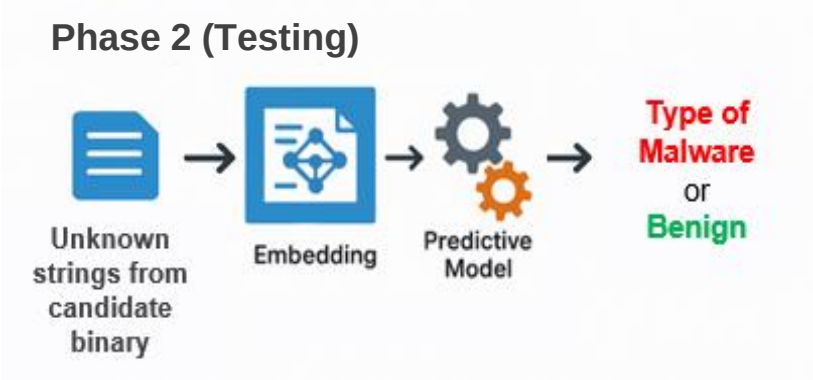




Methods: NLP (Google's BERT, Facebook AI's FastText)

- **BERT:** Transformers have shown success in many paradigms. A paper from 2021 showed that a BERT design is the best preforming transformer for classifying malware [1].
- **FastText:** It captures sub-word information, allowing meaningful embeddings for rare or out-of-vocabulary strings like API calls and C++ symbols commonly found in binaries.
- **Training:** We trained traditional classifiers (ie. RandomForest and Logistic Regression) on the resulting embeddings using k-fold cross validation to tune hyper-parameters.
- **Testing:** We used these NLP embeddings and classifiers to embed a candidate set of strings from a binary and classify it into a malware family.

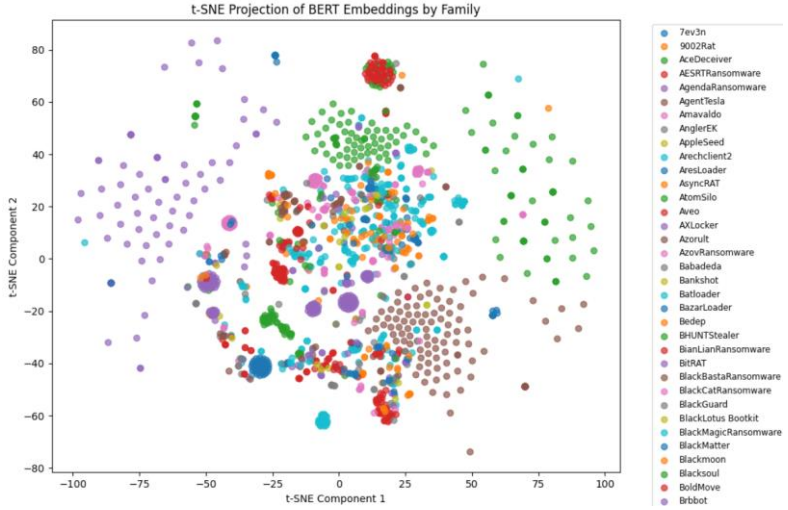


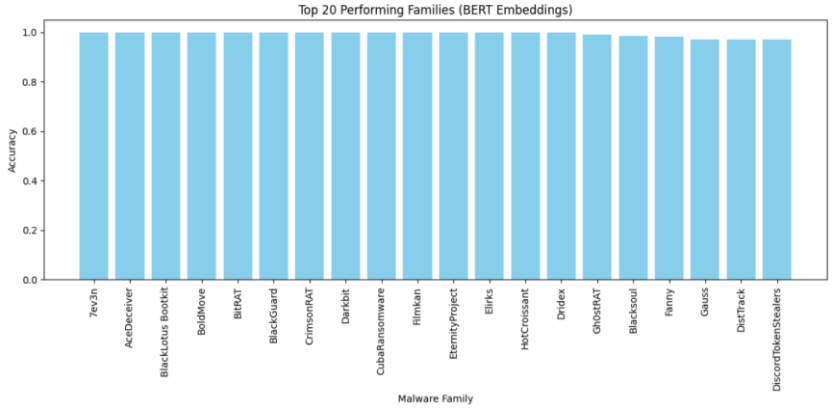


Results: NLP (Google's BERT)

Key Points:

- Large model: 12 transformer encoder layers, 110 million parameters.
- Attends to left context and right context simultaneously – intuitively explains benefits in Malware analysis
- Fine tuning too time intensive on given machine. 5 seconds per executable. ~50000 executables. ~3 days
- Trained on subset of 2000 samples as proof of concept and for insights for future work.

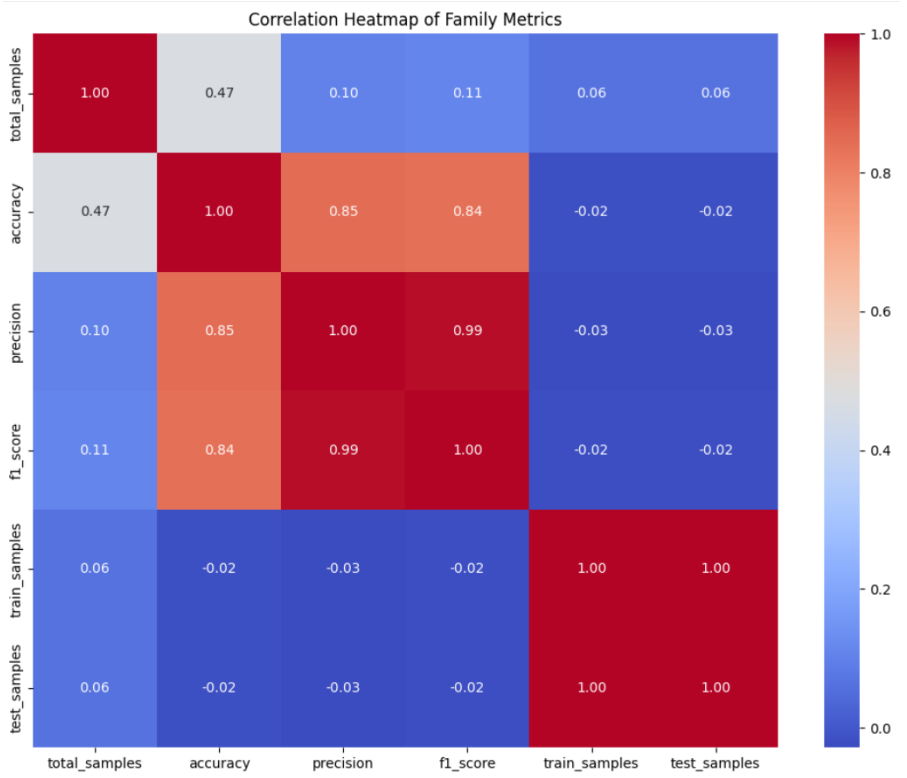




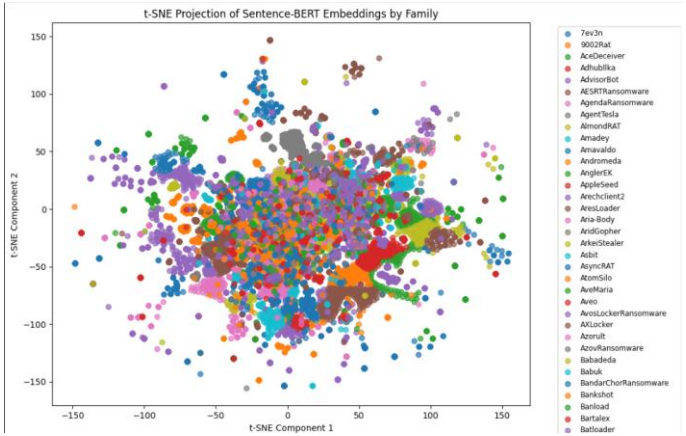
Results: NLP (Google's BERT)

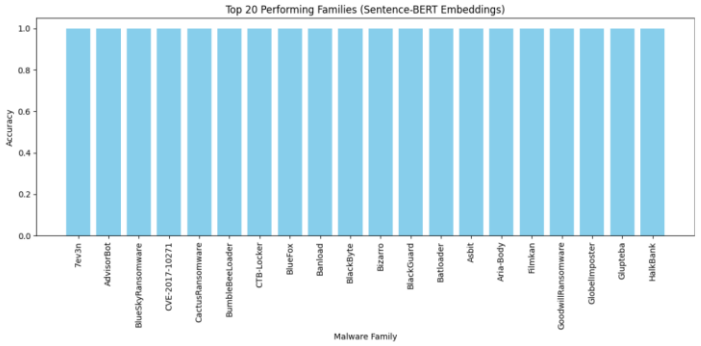
Key Points

- On smaller data 14 high performers (accuracy, precision and f1 > 90%)
- Overall model accuracy: 0.76
- 2021 paper reported 0.91 accuracy in detecting malware(1)
- Positive correlation (r = .47) between number of samples and accuracy



Results: Sentence-BERT (HuggingFace – paraphase miniLM) (3)





Key Points:

- Sentence transformer / SBERT: 3-layer variant with a **384-dimensional** embedding—offering ~6× fewer parameters and significantly faster inference, and fine tuning than BERT-base
- Clear Family Separation in t-SNE: The 2D t-SNE plot shows many malware families forming distinct clusters, indicating that even this lightweight model retains strong semantic separation among families.
- Overall accuracy: 0.86 despite much smaller model

Results: Sentence-BERT (HuggingFace – paraphase miniLM) (3)

Key Points:

•30 High-Performing Families

30 malware families exceed 90% in accuracy, precision, and F1.

•Clear Cluster Separation

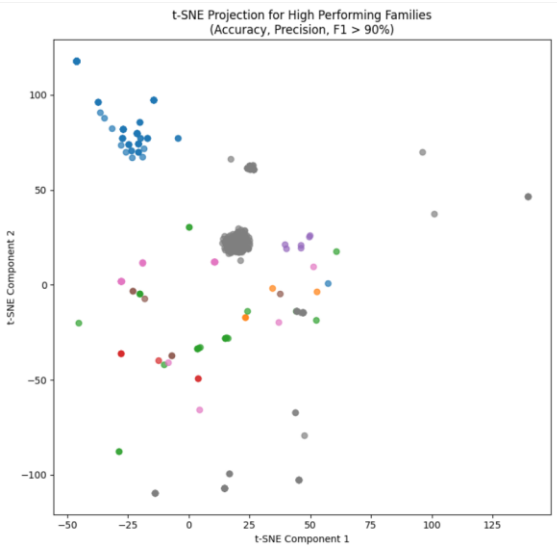
The t-SNE plot shows these top families forming tight, well-separated clusters, indicating the model's embeddings capture distinct semantic patterns.

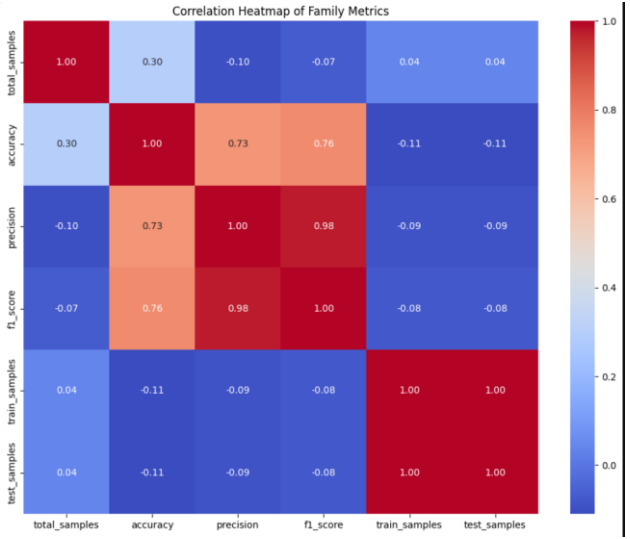
•String Length Drives Performance

Families with longer extracted string sequences tended to perform best (r≈0.30), suggesting richer textual content improves embedding quality.

•Metric Correlations

Precision and F1 remain almost perfectly correlated (r≈0.98); accuracy correlates strongly with both (r≈0.76), while neither metric is meaningfully driven by sample count (|r|<0.1).

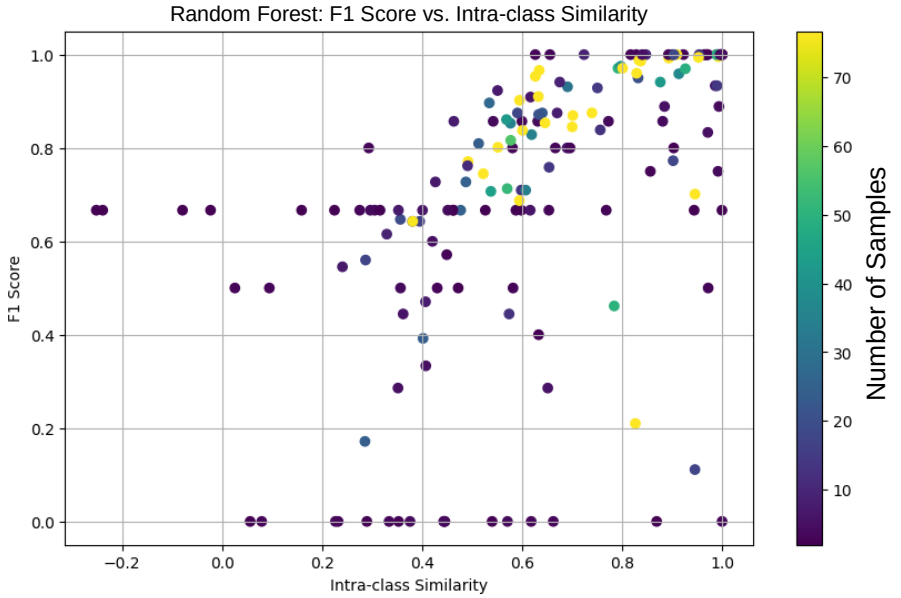




Results: NLP (Facebook AI's FastText Embedding)

Simple Classifiers on FastText Embedding:

- Began with supervised FastText embeddings with hyperparameters ngrams == 2 with a learning rate of 1.0.
- **High performer (HP) definition:** > 90% across f1, accuracy, and precision metrics.
- Logistic Regression with k-fold (31 total high-performers, 1 unique high-performing family 'UDPRat', overall test accuracy 89.8%)
- RF with k-fold (55 total high-performers, 14 unique families, overall test accuracy 90.6%)
- Since the Random Forest was more performant and classified all but 3 samples better than Logistic Regression, it made LR redundant.
- The cosine similarity between members of a class seemed to be a good indicator for performance as seen in the figure.

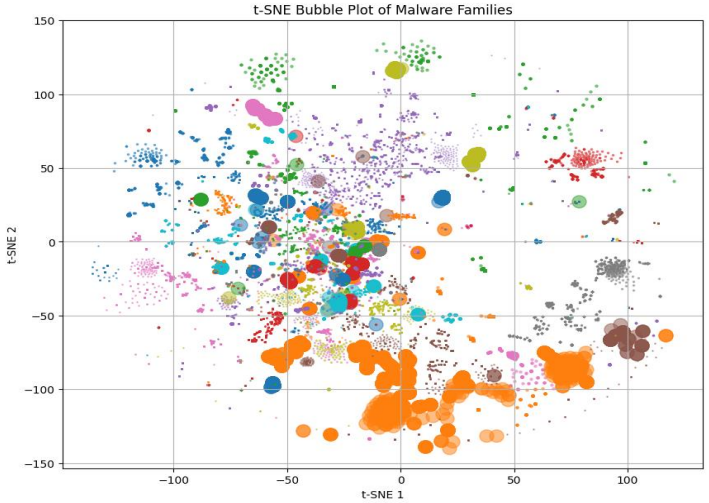


	Test Accuracy	High Performers	Unique HP
RF	90.6%	55	14
LR	89.8%	31	1

Results: NLP (Facebook AI's FastText Embedding)

K-Nearest Neighbors Approach

- T-distributed Stochastic Neighbor Embedding (t-SNE) plot reveals a few well clustered families that are poorly classified by LR and RF models.
- Decided to reduce the dimensionality of the FastText embedding using Uniform Manifold Approximation and Projection down to top 10 features.
- k=3 nearest neighbor model on the reduced dimensionality embeddings achieved a classification accuracy of 87.5% and reduced false positives.



	Test Accuracy	High Performers	Unique HP
RF	90.6%	55	14
LR	89.8%	31	1
KNN	87.5%	41	13
Ensemble	---------	70 (207 total families)	---------

Overall Results

	Test Accuracy	High Performers	Unique HP
RF*	90.6%	55	14
LR*	89.8%	31	1
KNN*	87.5%	41	13
SBERT*	86.0%	31	25
DNN	92.8%	54	30
DT	91.3%	39	2
RF	94.0%	62	4
Ensemble	———	128/207 total families	———

Table 2: Comparison of model performance and high-performing families (* - Text based classification, feature based otherwise)

Conclusion + Proposed Model/Future work

Feature-based Approach

- Filter features first using prior knowledge
- Look into other activation functions

NLP Approach

- Incorporate SHAP or LIME for model explainability.
- Analyze high-performing families.
- Explore ensemble methods with confidence-weighted voting.
- Optimize and tune hyperparameters for NLP embeddings.
- Tune hyperparameters that parse the string output (length, number of spaces).
- More testing on BERT model, with larger train and testing sets.

Thank you!

Questions?

References

1. A. Rahali and M. A. Akhloufi, "MalBERT: Malware Detection using Bidirectional Encoder Representations from Transformers," *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Melbourne, Australia, 2021, pp. 3226-3231, doi: 10.1109/SMC52423.2021.9659287.
2. Action-AI-Institute. *GitHub - action-ai-institute/MABEL-dataset: MABEL: Malware Analysis Benchmark for Artificial Intelligence and Machine Learning*. GitHub. https://github.com/action-ai-institute/MABEL-dataset.
3. *SentenceTransformers Documentation — Sentence-Transformers documentation*. www.sbert.net. https://www.sbert.net/.