Using Time Series Models to Predict G-Research Crypto Returns

cs 57300 Data Mining Course Project

PURDUE UNIVERSITY

Ruoling Fan, Donghun (James), Vinit, and Parth

Content

PURDUE UNIVERSITY

Part 1: Introduction & Related work

PURDUE UNIVERSITY

Introduction & Related Work

Introduction

- Time series data is a special data form in academia and industry
- G-research provided long-period returns of different cryptocurrencies with some relative features
- Cryptocurrency plays an essential roles in the world financial systems with some advantages.

Related Work

- ARIMA model
  - statistical model
  - without Feature
  - ignore relationships
- Machine Learning
  - independent data
  - ignore the sequence
- Deep Learning + ARIMA
  - further implement

Motivations

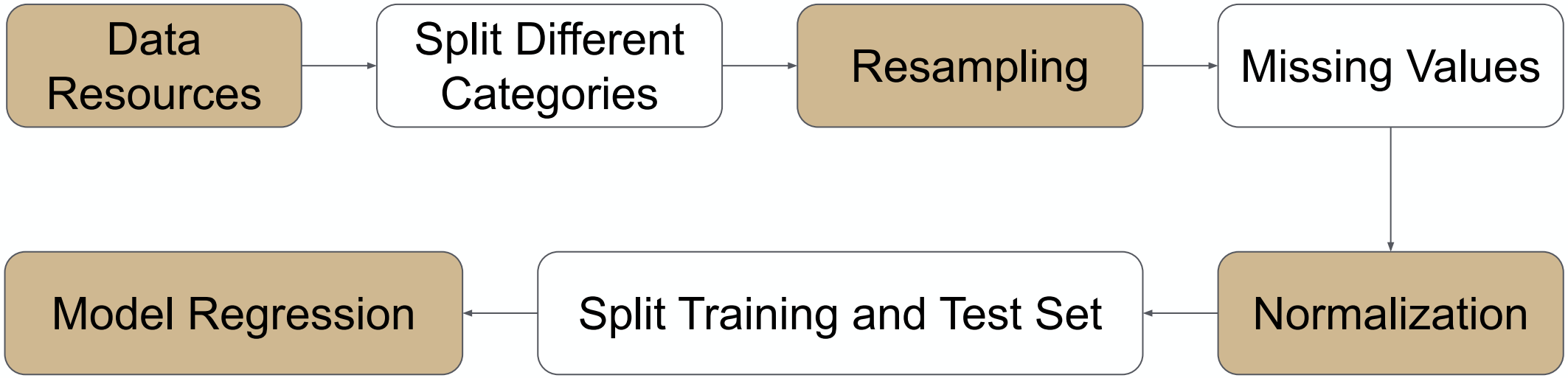
- New data form
  - different with class
- Resource easy acquired
  - open-resource
  - difference categories
- Widely applications
  - predict returns
  - risk management
  - asset allocation
- Academic and industrial combination

PURDUE UNIVERSITY

Part 2: Data Pre-processing

PURDUE UNIVERSITY

Data Preprocessing & Road Map



- It contains long period and more data.
- bItcoin: maintained a market dominance (approximately 51%, accounting for over half of the total global cryptocurrency market capitalization, which is valued around $2.43 trillion [1])

PURDUE UNIVERSITY

https://www.coingecko.com/en/global-charts

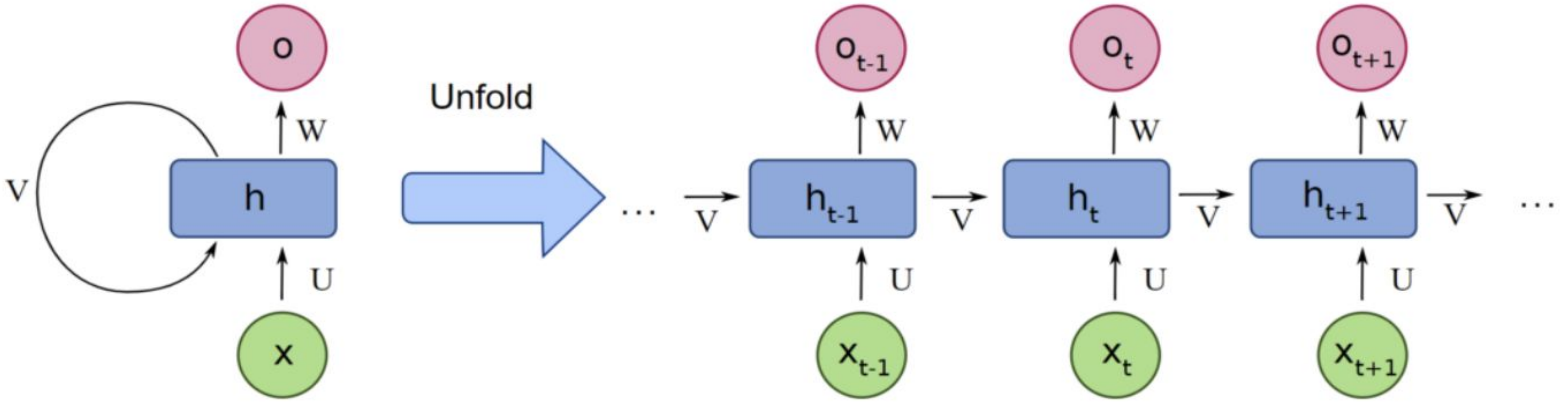
Data Preprocessing & Road Map

timestamp	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
2018-01-01 00:15:00	1.0	0.002846	0.171156	0.173630	0.168723	0.170734	0.004728	0.170846	0.003995
2018-01-01 00:30:00	1.0	0.001863	0.170120	0.171542	0.168314	0.169888	0.002540	0.169997	-0.001220
2018-01-01 00:45:00	1.0	0.002492	0.170083	0.172107	0.167830	0.170101	0.003934	0.169951	0.000087
2018-01-01 01:00:00	1.0	0.002904	0.170027	0.171970	0.169095	0.170024	0.002659	0.169829	-0.004592
2018-01-01 01:15:00	1.0	0.001543	0.169098	0.170589	0.167776	0.168976	0.001165	0.168930	0.004491

Part 3: Recurrent NN

PURDUE UNIVERSITY

RNN Architecture



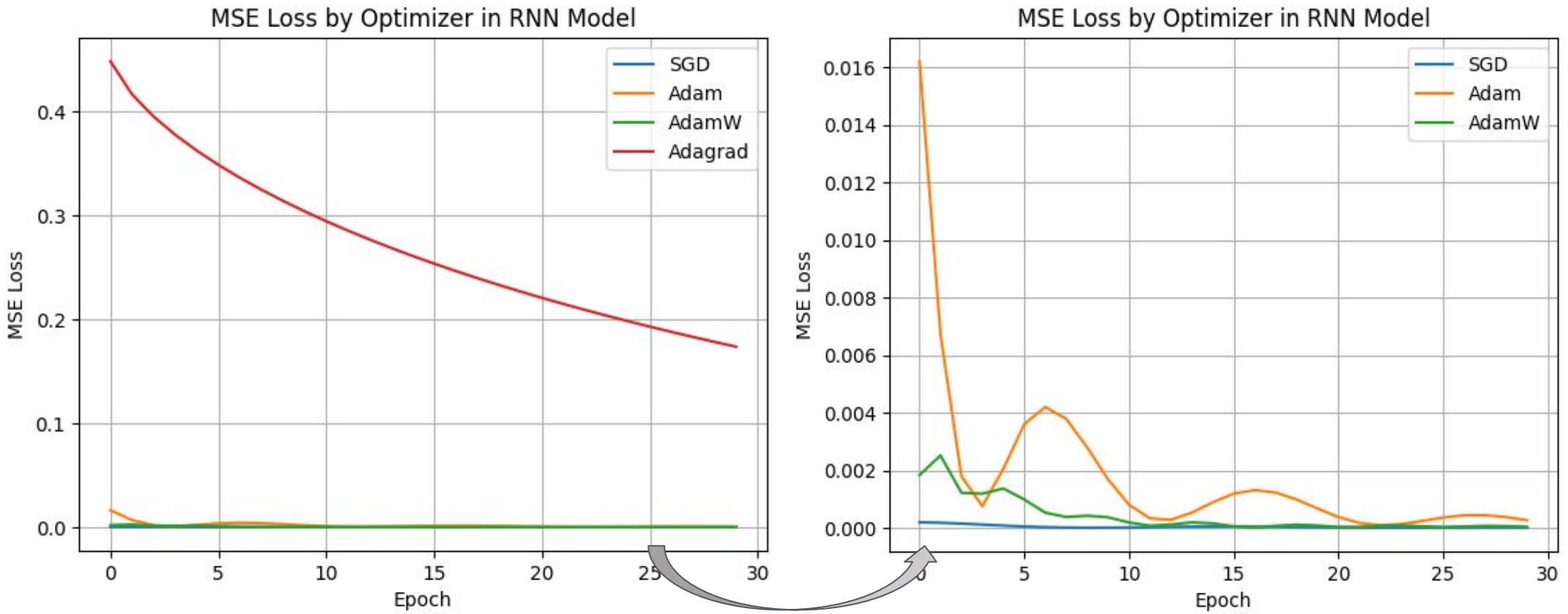
- **Core Concept:**
  - sequential data processing, suitable for time series data or natural language texts.
- **Structural Components:**
  - Input Layer:
  - Hidden Layer: $\mathbf{h_t} = f(\mathbf{W_1 h_{t-1}} + \mathbf{W_2 x_t} + \mathbf{b})$
  - Output Layer: $\mathbf{o_t} = g(\mathbf{W_3 h_t} + \mathbf{b})$
- **Characteristics:** Parameter Sharing; Long-term Dependencies Problem

PURDUE UNIVERSITY

Citation: https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/

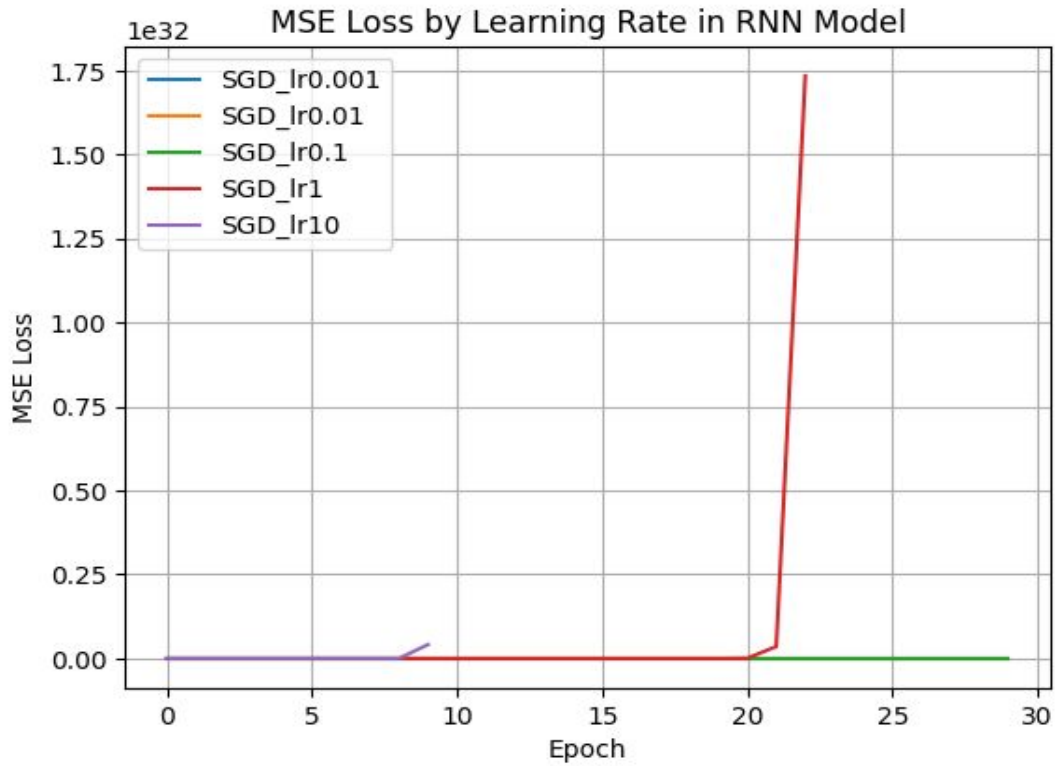
RNN Fine Tuning - Optimizer

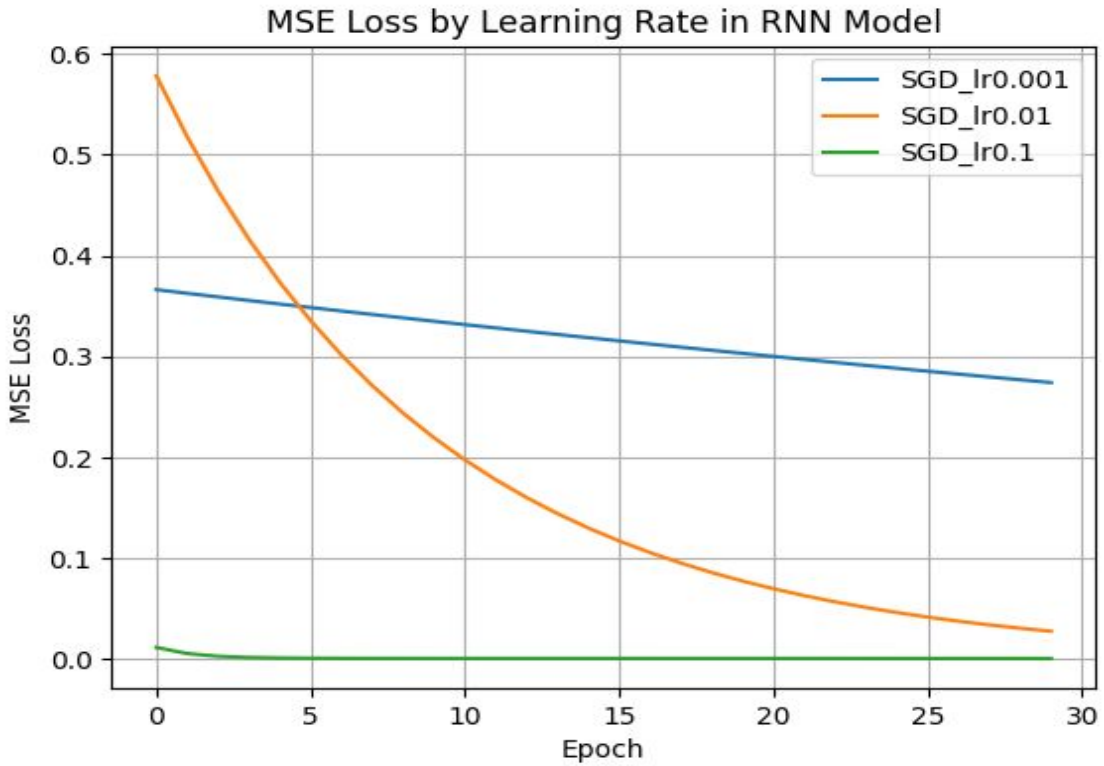
- SGD: Stochastic Gradient Descent
- Adam: Computing the first and second derivative
- AdamW: Improvement on Adam, adopting decay weights to make regularization
- Adagrad: adopts fluctuating learning rate according to the historical gradient size



RNN Fine Tuning - Learning Rate

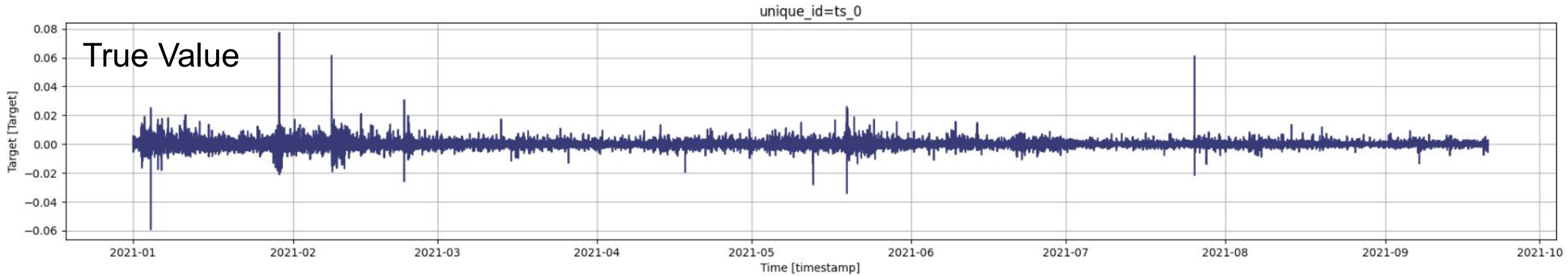
- Learning rate is 1 or 10, there was a huge jump in the loss
- Best hyperparameters: Optimizer = ‘SGD’, Learning_rate = 0.1

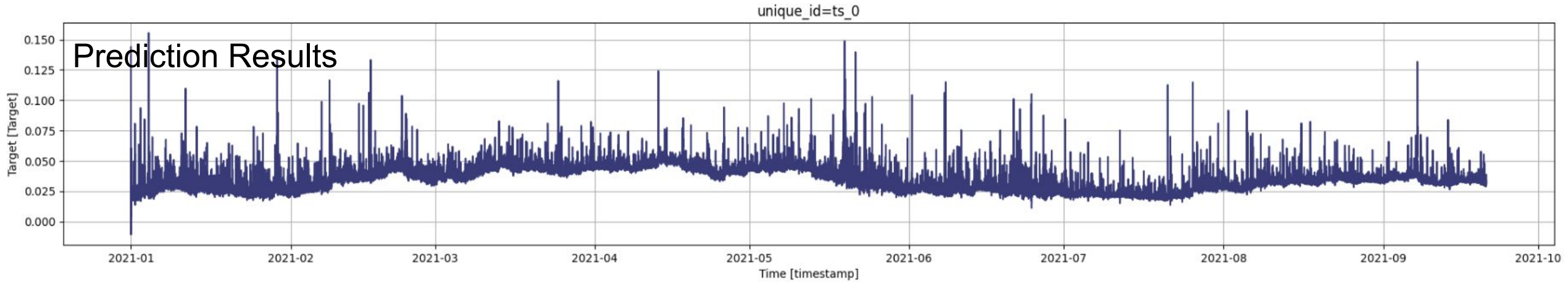




PURDUE UNIVERSITY

RNN Prediction Results





MSE = 0.002996540444

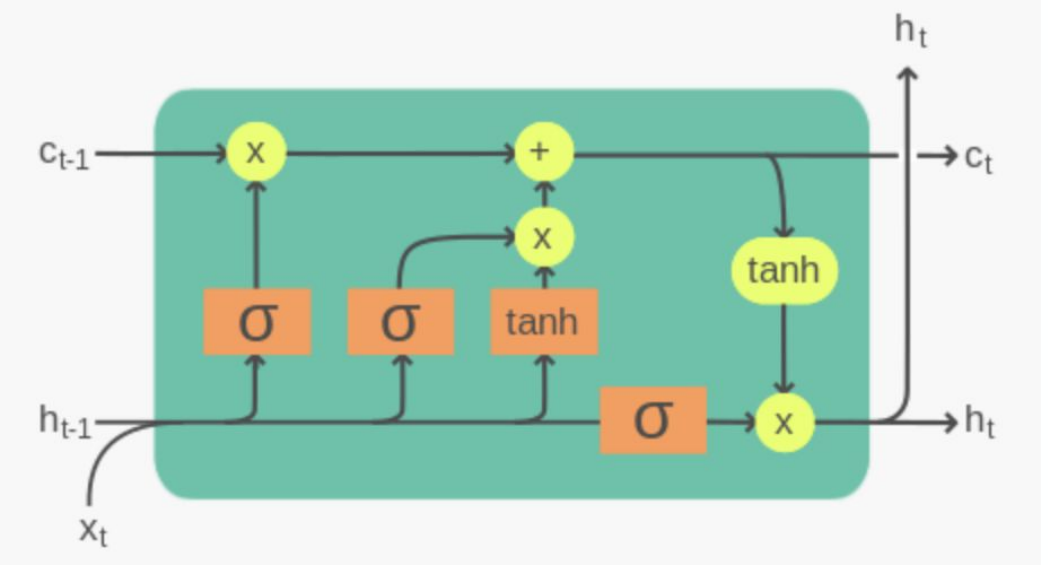
PURDUE UNIVERSITY

Part 4: LSTM Model

PURDUE UNIVERSITY

LSTM Architecture

- **Core Concept:**
  - handle long-term dependencies, addressing the vanishing and exploding gradient problems common in standard RNNs.
- **Structural Components and Mathematical Formulas:**
  - Input Layer: Receives sequence elements
  - LSTM Cell:
    - Forget Gate $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
    - Input Gate $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
    - Cell Candidate $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
    - Cell State Update $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
    - Output Gate $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
    - Hidden State Update $h_t = o_t * \tanh(C_t)$
  - Output Layer:
    - Outputs $y_t = W_{hy} \cdot h_t + b_y$

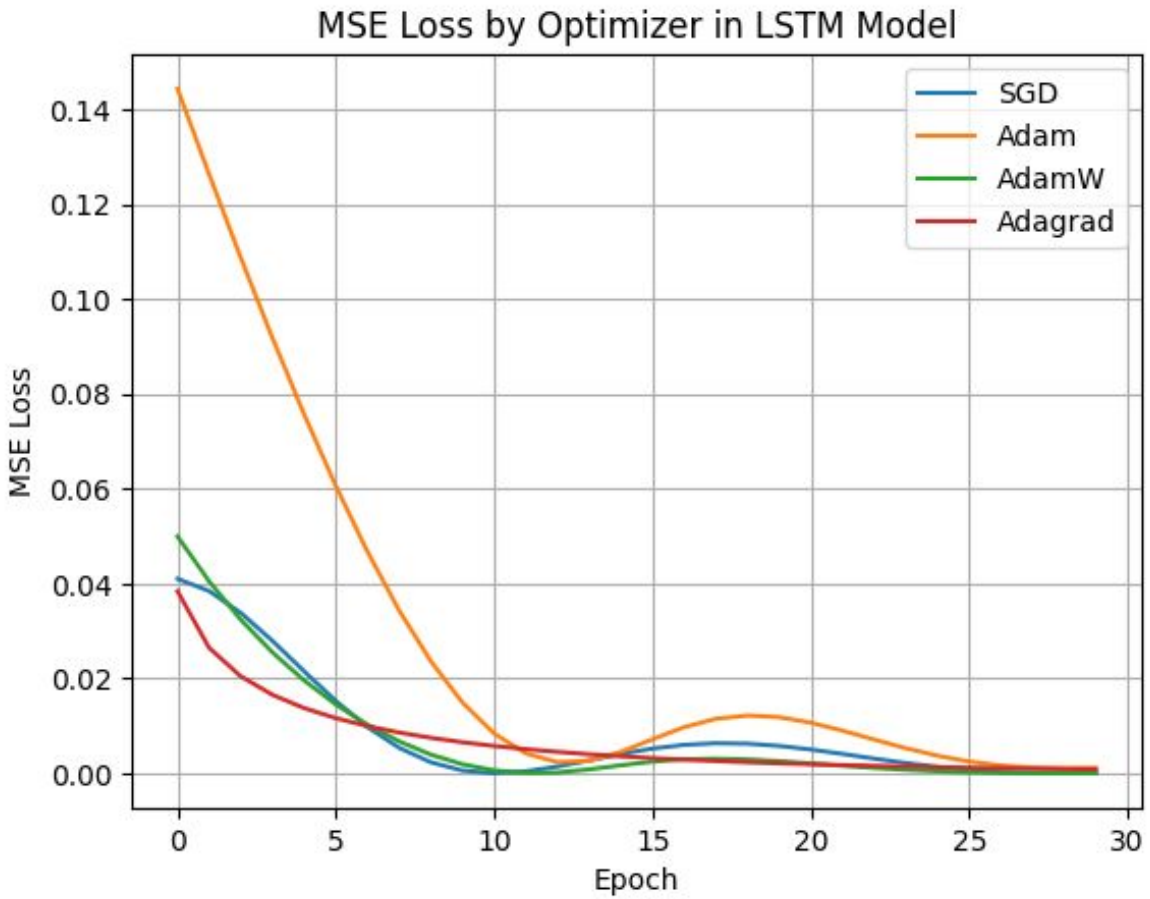


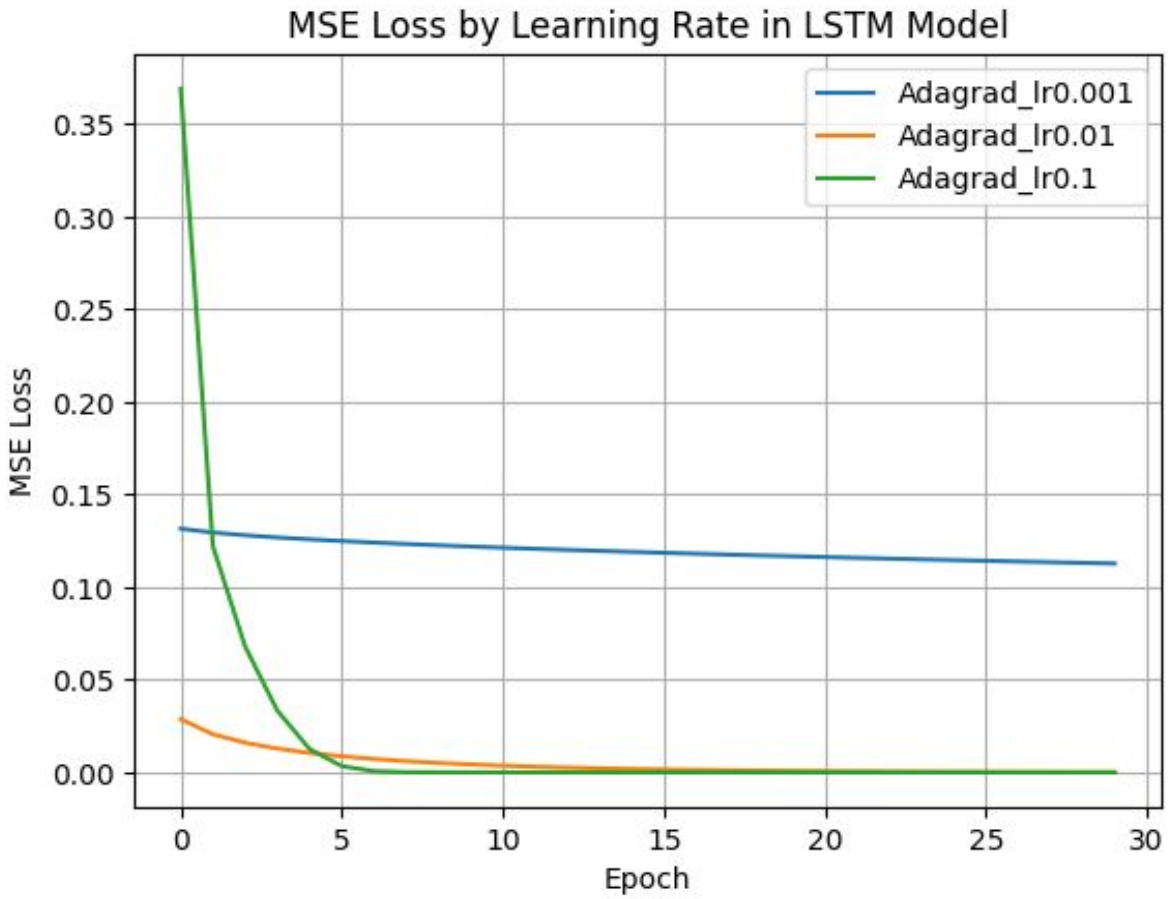
- **Characteristics:**
  - Advanced gating mechanisms to control the flow of information.
  - Handel noise efficiently since the gating mechanisms.

Citation: https://machinelearningmastery.com/lstm-for-time-series-prediction-in-pytorch/

LSTM Fine Tuning

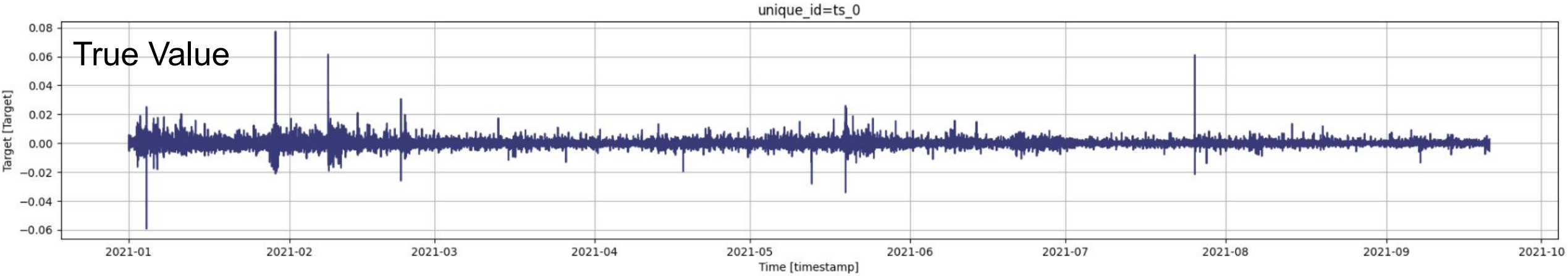
- Best hyperparameters: Optimizer = ‘Adagram’, Learning_rate = 0.01

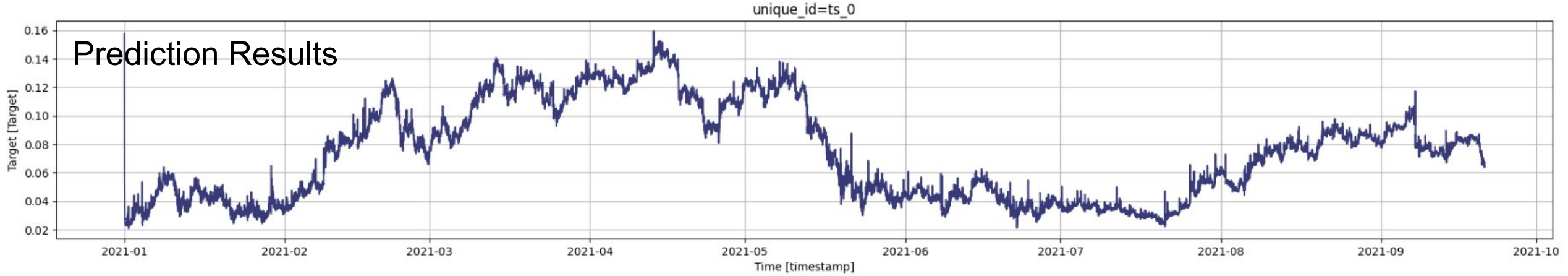




PURDUE UNIVERSITY

LSTM Prediction Results





MSE = 0.0037925077

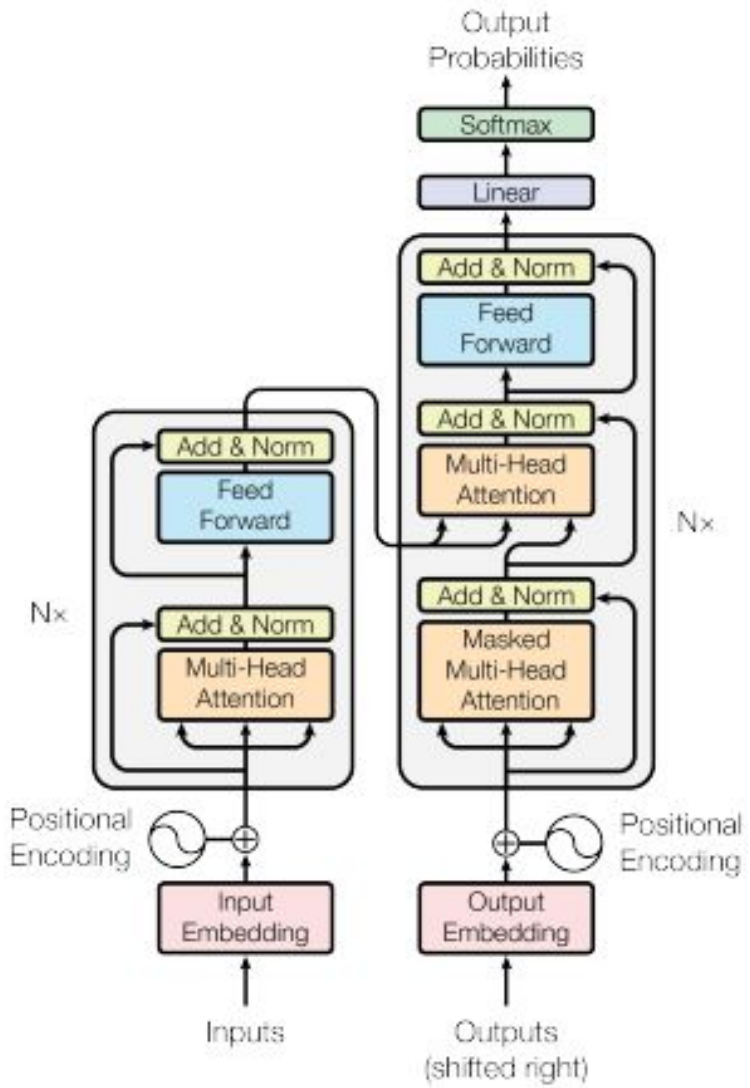
PURDUE UNIVERSITY

Part 5: Transformer

PURDUE UNIVERSITY

Transformer Architecture

- Is a deep learning model that eliminates recurrence and instead uses an attention mechanism.
- Attention assigns different degrees of importance to different elements in specific parts of the input.
- Encoder maps an input sequence to a continuous representation.
- Decoder uses encoded input and previous timestep decoder output to create an output sequence.



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

PURDUE UNIVERSITY

Adjusted Transformer Architecture

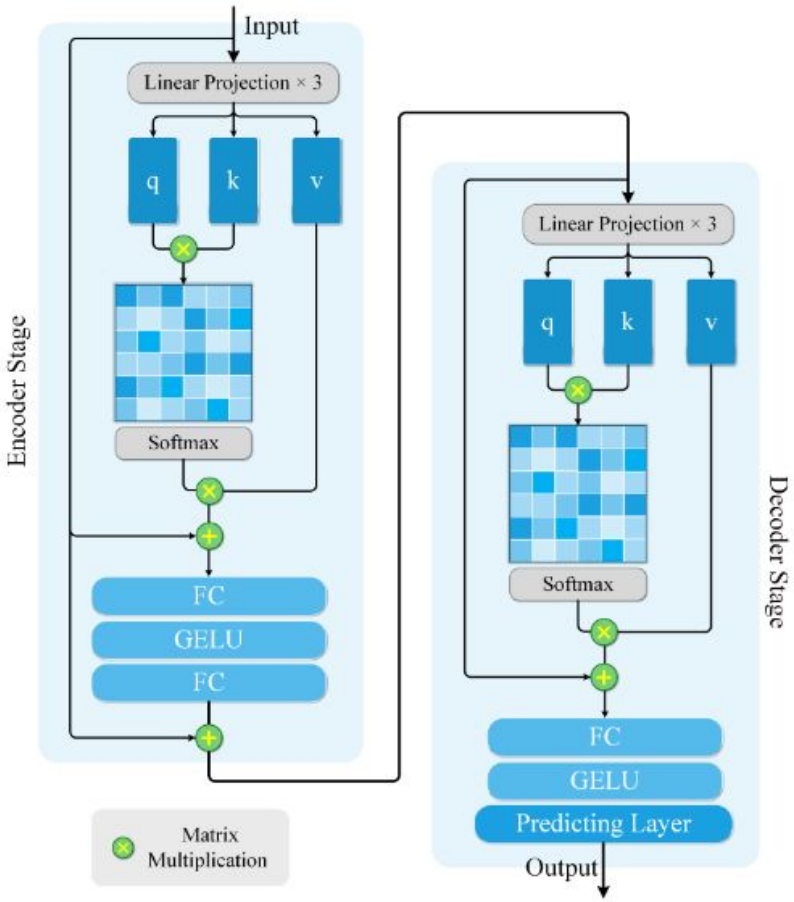
- Feasible to train transformer without a GPU.
- Same architecture as the original but now we only have the encoded input as an input into the decoder.
- 64 hidden units for all the fully connected layers.
- Gaussian Error Linear Units (GELU) improves performance in transformer architectures.
- Mean Squared Error (MSE) is loss function.

$$Softmax\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

$$GELU\left(x\right) = xP\left(X \leq x\right) = 0.5x\left(1 + erf\left(\frac{x}{\sqrt{2}}\right)\right)$$

$$\text{where } P(X) \sim N(0,1)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \tilde{y}_i\right)^2$$

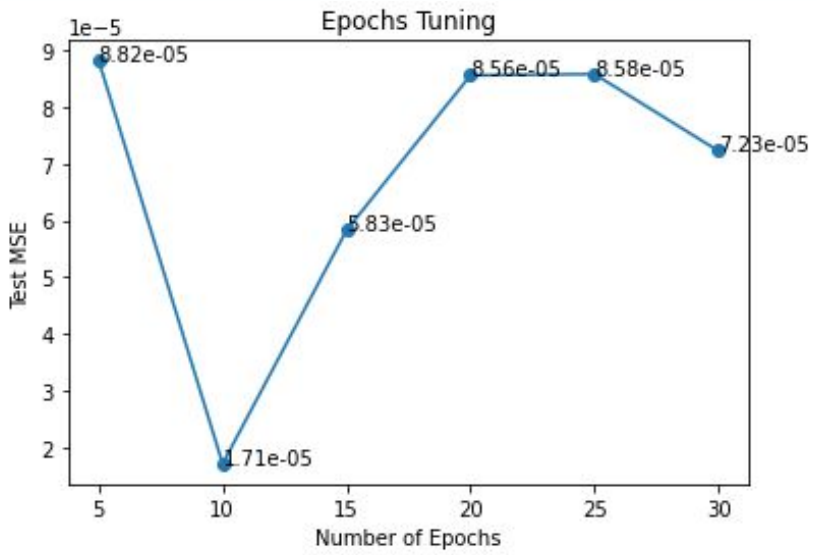


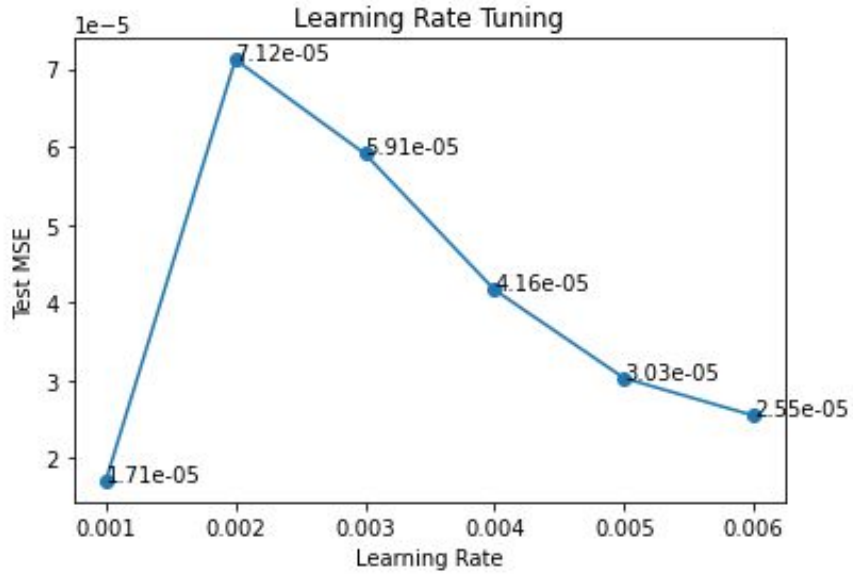
Evergreen0929. A light-weight transformer model for kaggle house prices regression competition.

PURDUE UNIVERSITY

Hyperparameters

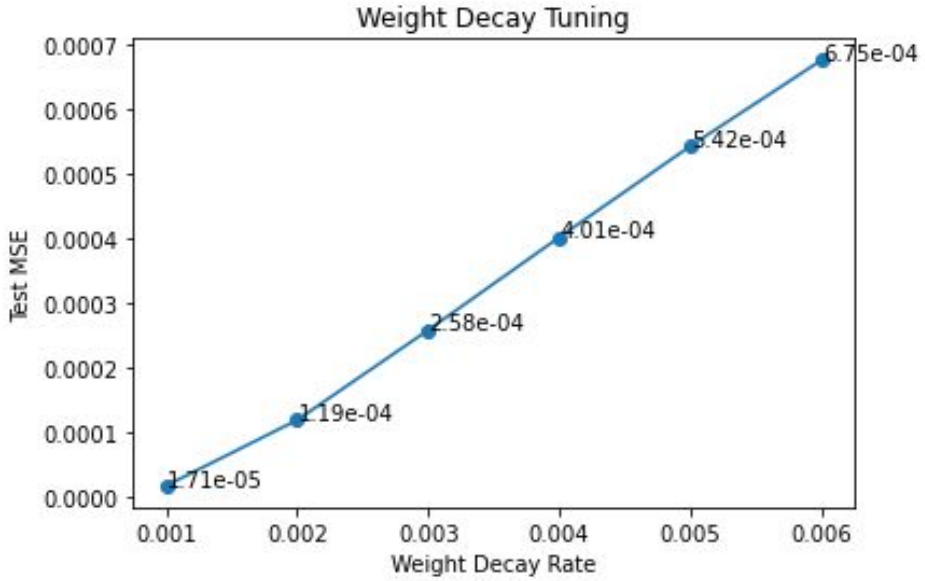
- number of epochs = number of iterations we want the transformer to run
- learning rate = size of the step in optimization process
- weight decay = regularization technique which penalizes large weights
- batch size = number of training examples used in one iteration of optimization process.

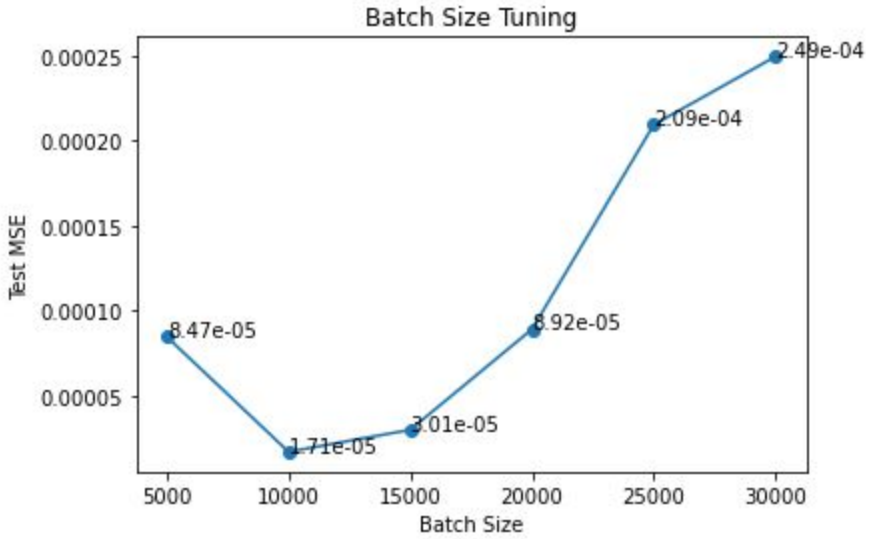




PURDUE UNIVERSITY

Hyperparameters





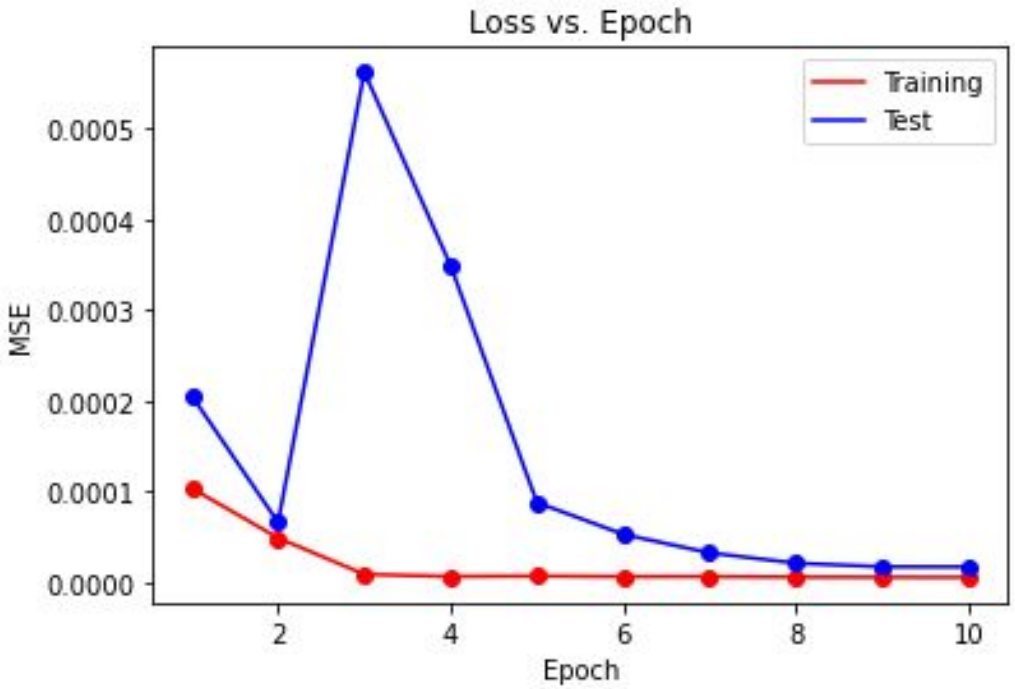
Final Hyperparameters:

- number of epochs = 10
- learning rate = 0.001
- weight decay = 0.001
- batch size = 10,000

PURDUE UNIVERSITY

Model Evaluation

- As the epochs increase the training loss becomes smaller.
- Around the third epoch test loss suddenly increases, but then starts to decrease probably due to the model guessing well early on for the wrong reasons.
- Model fits well since there isn't a huge gap between our training and test loss and the MSE is low.
- MSE of 0.0000055610 on the training set
- MSE of 0.0000171145 on the test set

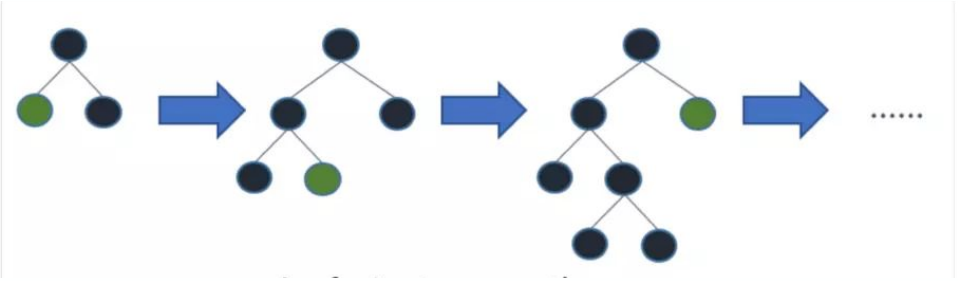


Part 6:
Machine Learning Models

PURDUE UNIVERSITY

Light GBM

LightGBM (Light Gradient Boosting Machine) is a powerful gradient boosting framework designed for efficiency and scalability.

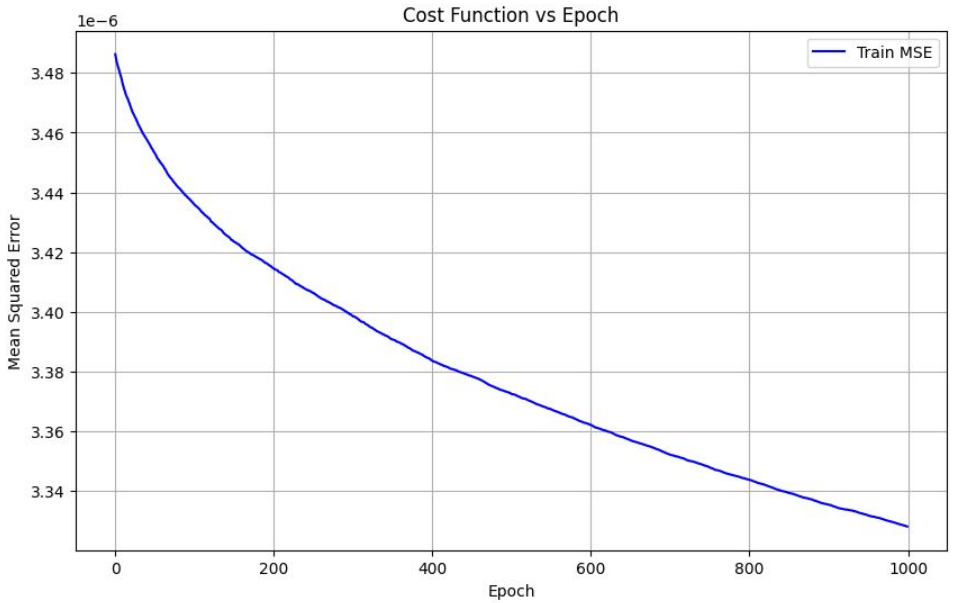


Key Features:

- Efficient tree-based learning algorithms.
- Leaf-wise tree growth strategy for faster convergence.
- Histogram-based algorithms for optimized memory usage.
- Support for large datasets and complex problems.

Working Principle:

- LightGBM builds decision trees sequentially, focusing on nodes with the maximum delta loss during each iteration.
- Uses a leaf-wise approach for tree growth, prioritizing nodes that yield the highest information gain.



Advantages:

- Faster training and prediction times.
- Improved accuracy and model performance.
- Scalability for handling large datasets.

PURDUE UNIVERSITY

Citation: https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters

Neural Network

Neural Network (Feedforward Neural Network)

Key Features:

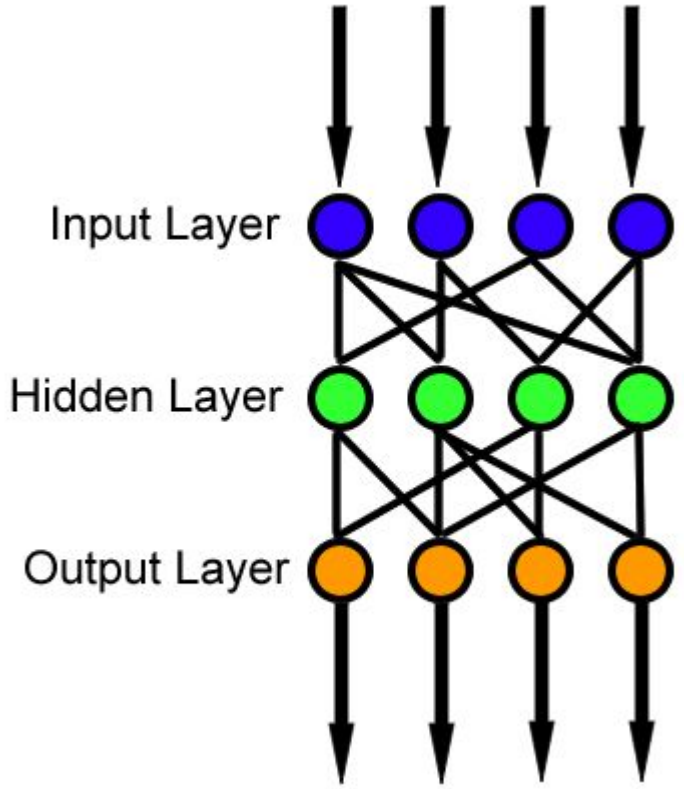
- Stacked Layers Architecture
- Uni-directional Information Flow
- Trainable Weights

Working Principle:

- Each layer consists of interconnected artificial neurons that perform calculations on the data they receive. Feedforward neural networks learn through Backpropagation. By iteratively adjusting the weights, the network learns to map the input data to the desired output.

Advantages:

- Non-linear Modeling
- Automatic Feature Extraction
- Improved accuracy and model performance.
- Scalability for handling large datasets.



Training Error	Test Error
3.6×10^{-6}	6.7×10^{-6}

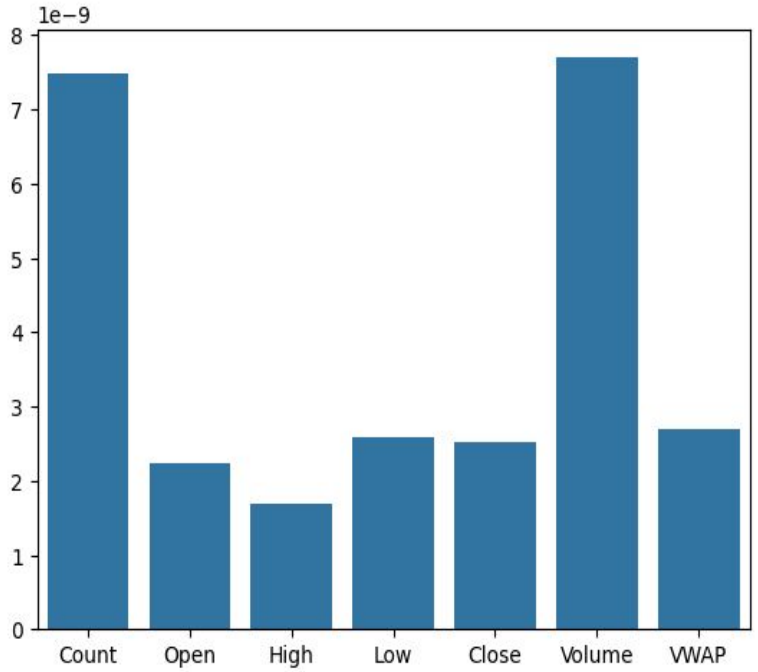
Citation: https://en.wikipedia.org/wiki/Feedforward_neural_network

PURDUE UNIVERSITY

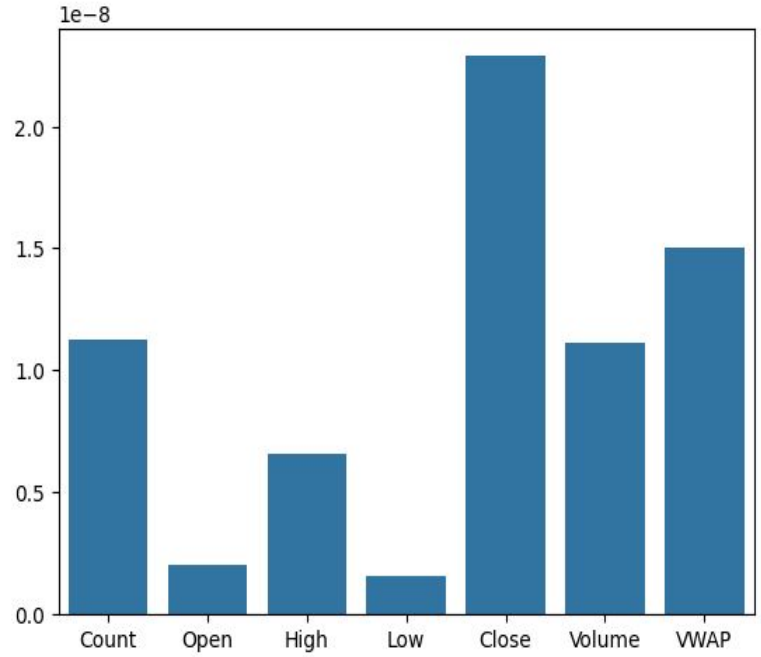
Feature Importance

Permutation Importance

- works by randomly shuffling the values of a single feature and then measuring the effect on the model's accuracy
- If the accuracy of the model drops significantly after shuffling a particular feature, it suggests that the feature is important



LightGBM



Neural Network

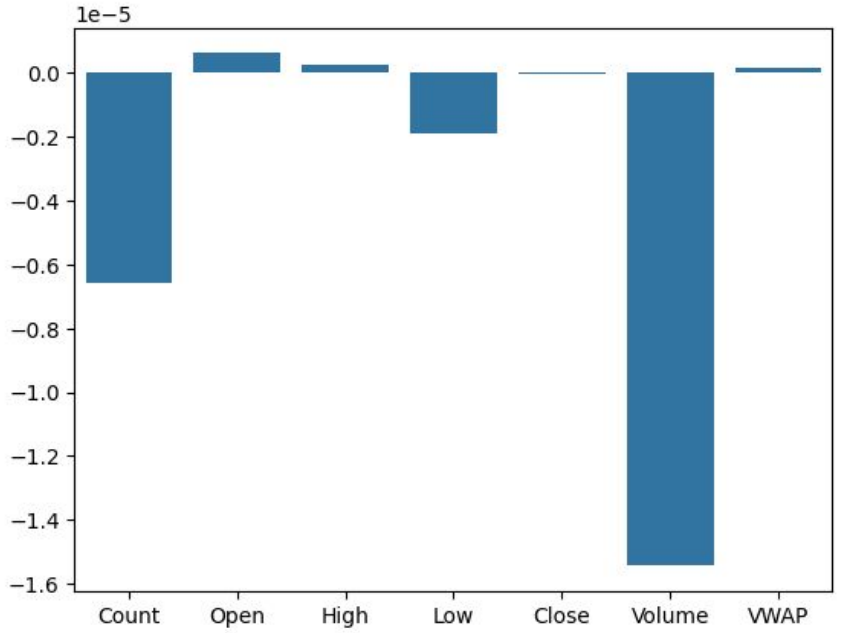
PURDUE UNIVERSITY

Feature Importance

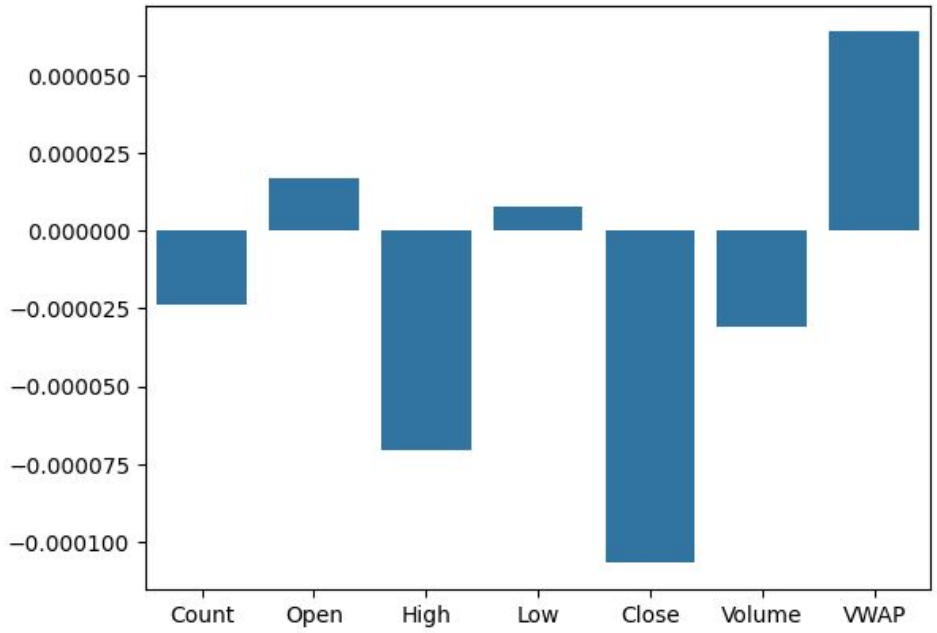
Donghun (James) Lee Ends

Shap (SHapley Additive exPlanations)

- a game theoretic approach to explain the output of any machine learning model
  - Considering all Coalitions of features
  - Marginal Contribution: calculate the additional value a feature brings when joining
  - Averaging Across Coalitions: average these marginal contributions across all possible coalitions



LightGBM



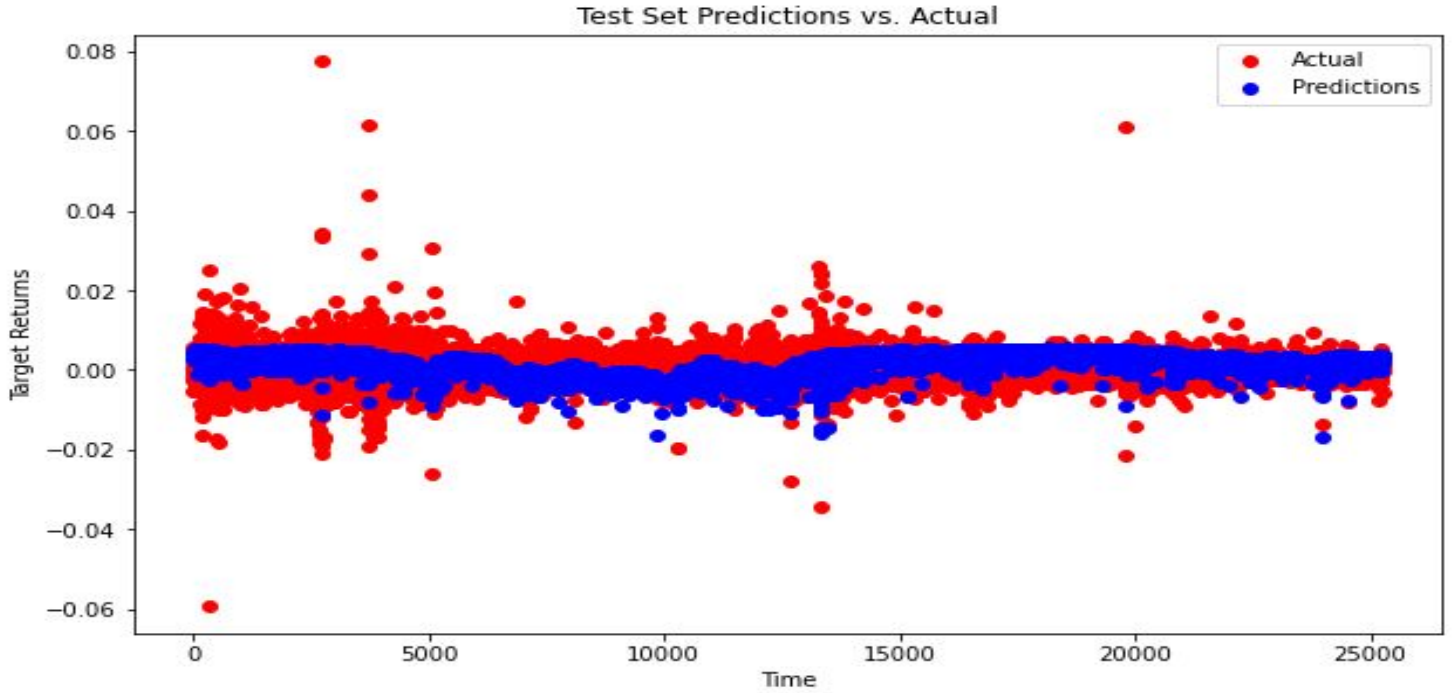
Neural Network

Part 7: Conclusion

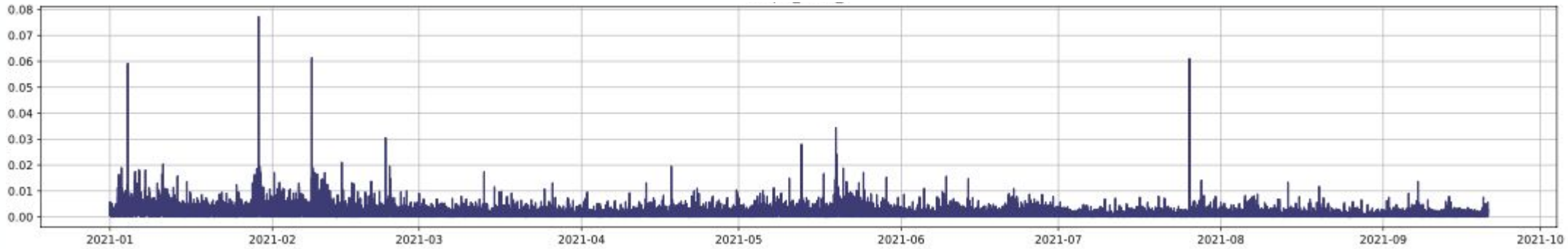
PURDUE UNIVERSITY

Model Evaluation

Results from Transformer



Absolute error from Transformer



PURDUE UNIVERSITY

Comparison of Models

Metric: Mean Squared Error

Observations:

- Machine Learning models in general perform better than RNN / LSTM.
- Among the deep learning models Transformer performs the best.
- Among the ML models - Light GBM has the least test error, which signifies that it learns to predict the target better. Other models are overfitting.

Model	Training Error	Test Error
RNN	0.001251	0.002997
LSTM	0.003793	0.003792
Transformer	5.561×10^{-6}	1.711×10^{-5}

Model	Training Error	Test Error
LightGBM	**3.3×10^{-6}**	**6.4×10^{-6}**
CatBoost	3.6×10^{-6}	6.7×10^{-6}
XGBoost	3.6×10^{-6}	6.7×10^{-6}
Neural Network	3.6×10^{-6}	6.7×10^{-6}
KNeighbors	2.9×10^{-6}	7.1×10^{-6}
Extra Trees	1.3×10^{-6}	7.1×10^{-6}
Weighted Ensemble	3.5×10^{-6}	7.2×10^{-6}
Random Forest	9.0×10^{-7}	7.9×10^{-6}

PURDUE UNIVERSITY

Insights

Why simple ML models work better?

- **Less Data Requirement**: Machine learning models need less data compared to deep learning.
- **Simplicity and Efficiency**: Traditional models like ARIMA are simpler and more resource-efficient.
- **Interpretability**: Machine learning models offer clearer insights into predictions.
- **Noise Robustness**: Machine learning models handle noise and outliers better.
- **Feature Engineering**: Manual feature engineering improves model performance.

Model	Training Error	Test Error
RNN	0.001251	0.002997
LSTM	0.003793	0.003792
Transformer	5.561×10^{-6}	1.711×10^{-5}

Model	Training Error	Test Error
LightGBM	**3.3×10^{-6}**	**6.4×10^{-6}**
CatBoost	3.6×10^{-6}	6.7×10^{-6}
XGBoost	3.6×10^{-6}	6.7×10^{-6}
Neural Network	3.6×10^{-6}	6.7×10^{-6}
KNeighbors	2.9×10^{-6}	7.1×10^{-6}
Extra Trees	1.3×10^{-6}	7.1×10^{-6}
Weighted Ensemble	3.5×10^{-6}	7.2×10^{-6}
Random Forest	9.0×10^{-7}	7.9×10^{-6}

PURDUE UNIVERSITY

Part 8: Future Research Directions

PURDUE UNIVERSITY

Future Research Directions

Vinit Singh End

- We stated that we would use the models timeGPT, iTransformer, and Mamba for this task, but couldn't due to the need for GPUs and not free.
    - timeGPT, iTransformer, and Mamba improve Transformer architecture and address its weaknesses
- Using the best performance model in financial market to make predictions in the real-world.
- We will also test the deep learning + ARIMA model (combine deep learning and statistical model).
- We hope to get a high Pearson Correlation Coefficient in the Kaggle Competition with our model. Top PCC was 0.0326.
- Integrate and embed the prediction models into financial trading platform to ensure timely predictions (with backend techniques).

Thank You

04/18/2024

PURDUE UNIVERSITY®

March Madness Entry Prediction

By: Akhil Thata, Aryan Jain, Fenil Gala, and Maheep Brar

Context

Figure 1: NCAA March Madness Logo [5]

- **March Madness**
  - NCAA Division I Men's **Basketball Knockout Tournament**
  - One of the most popular annual sporting event in the nation
  - UConn vs. Purdue Finals had **14.82** million viewers [8]
- **Selection Process**
  - Automatic Bids: **32 teams** qualify by winning their conference tournaments
  - At-Large Bids: **36 teams** are selected by the committee based on the criteria

Selection Criteria

Quantitative Data	Qualitative Data
Season Record	Quality Wins/Losses
Key Metrics	Injuries and Suspensions
Recent Performance	Eye Test (Style of Play)
Scoring Margins	Reputation

Problem Statement

Find a data-driven method of predicting who will qualify for March Madness using Data Mining

Dataset

- **Kaggle College Basketball Dataset** [3]
  - By Andrew Sunberg
  - 23 columns of information
  - Consists of 363 different teams
- **Trained on seasons 2014 to 2023, tested on 2024**
  - Except 2020: Due to COVID-19
- **Important Columns**

Power Rating	Win Percent	Defensive Efficiency
ThreePointShooting	TwoPointShooting	FreeThrowRate
Wins	NumberGamesPlayed	TurnoverRate

Preprocessing

Preprocessing

- Converted conference names to codes
- Following map for ‘Qualified for NCAA’ column
  - 1 if ‘TournamentSeed’ is not null
  - 0 if ‘TournamentSeed’ is null
- Dropped …
  - ‘Wins Above Bubble’ - calculated after March Madness
  - ‘Conference’ - Already have Conferences in form of code
  - ‘NCAATournamentSeed’ - Don’t need this anymore due to ‘Qualified for NCAA’

Major Conference	*Mid-major Conference*	*Small Conference*
Atlantic Coast Conference	Atlantic 10 Conference	America East Conference
Big 12 Conference	Colonial Athletic Association	Atlantic Sun Conference
Big East Conference	Conference USA	Big Sky Conference
Big Ten Conference	Horizon League	Big South Conference
Pacific-10 Conference	Mid-American Conference	Big West Conference
Southeastern Conference	Missouri Valley Conference	Ivy League
	Mountain West Conference	Metro Atlantic Athletic Conference
	West Coast Conference	Mid-Eastern Athletic Conference
	Western Athletic Conference	Northeast Conference
		Ohio Valley Conference
		Patriot League
		Southern Conference
		Southland Conference
		Southwestern Athletic Conference
		Summit League
		Sun Belt Conference

Figure 2: List of NCAA Division 1 Conferences[7]

Feature Engineering

- Team Heritage Column
  - Number of times a team has made it to March Madness
- Transformed Wins and Games Played to Win%
  - Number of games Played is not standardized

	TeamName	SeasonYear	TeamHeritage	QualifiedForNCAA
1820	Purdue	2013	0	0
1658	Purdue	2014	0	0
2179	Purdue	2015	1	1
2182	Purdue	2016	2	1
2413	Purdue	2017	3	1
2415	Purdue	2018	4	1
21	Purdue	2019	5	1
2469	Purdue	2021	6	1
2812	Purdue	2022	7	1
3161	Purdue	2023	8	1

Figure 3: Team Heritage for Purdue from 2013 - 2023

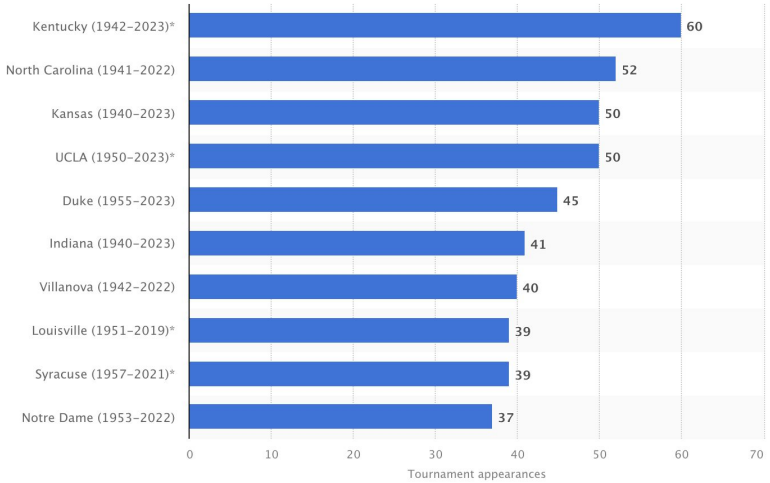


Figure 4: Top 10 appearances by team from 1939 to 2023 [6]

Exploratory Analysis

Offensive Efficiency

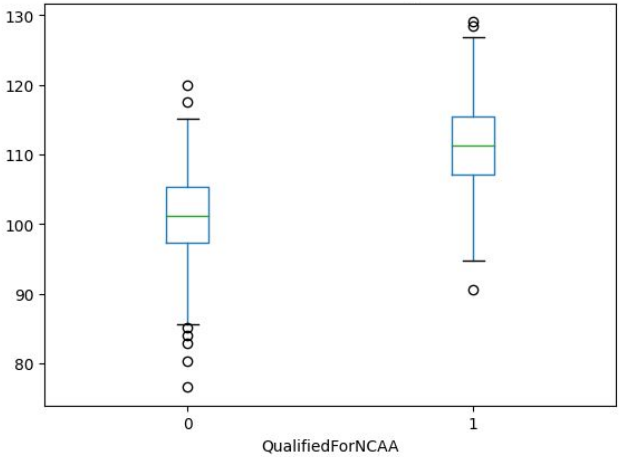


Figure 5: Boxplot of Offensive Efficiency

Free Throw Rate

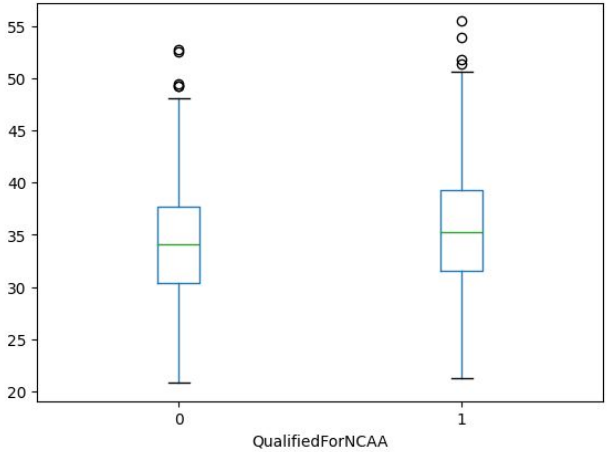


Figure 6: Boxplot of Free Throw Rate

Win Percent

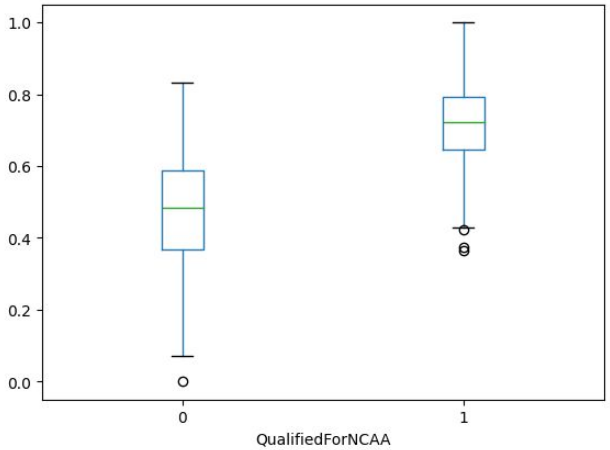


Figure 7: Boxplot of Win Percent

Exploratory Analysis - PCA

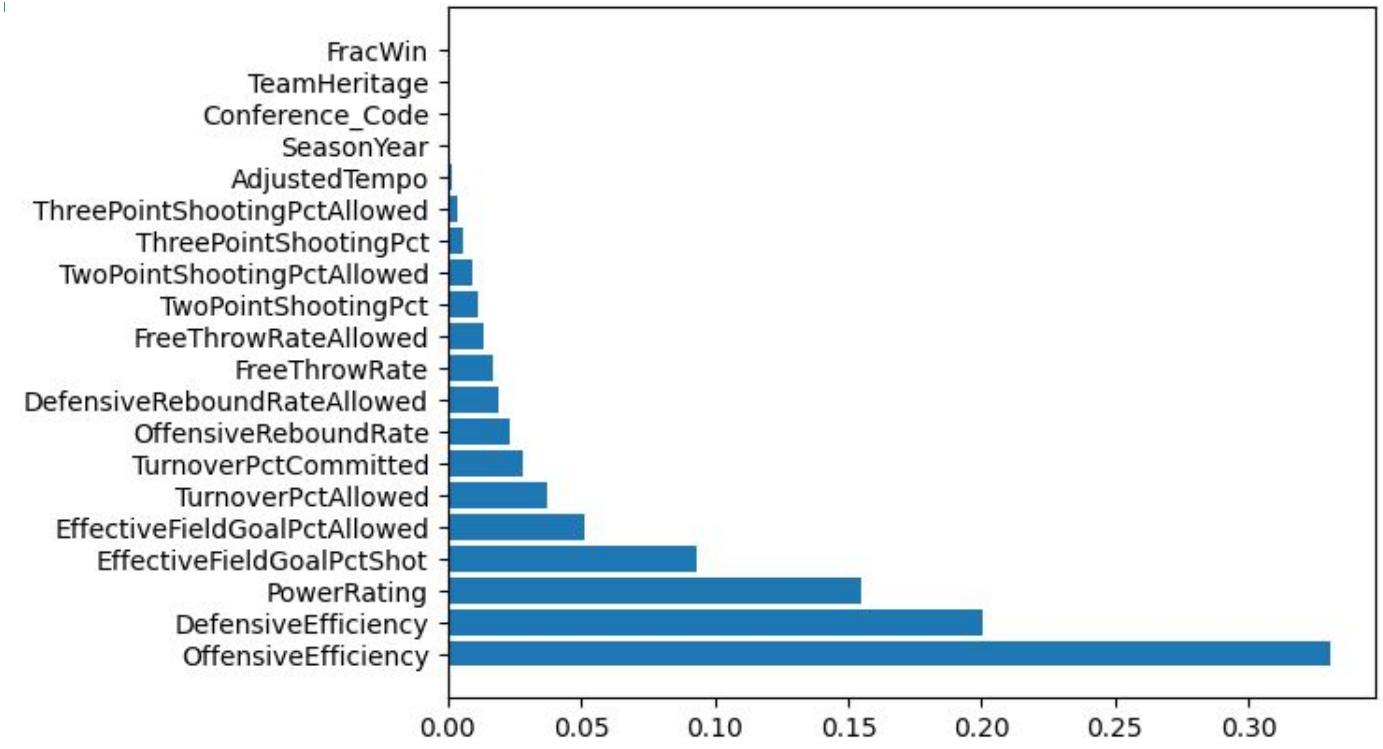


Figure 8: PCA of dataset

Exploratory Analysis
HeatMap

- Offensive/Defensive/Power Rating heavily correlated
- Power rating and Fractional Win Correlated
- Effective Field Goal Percent Allowed and Two Point Shooting Allowed

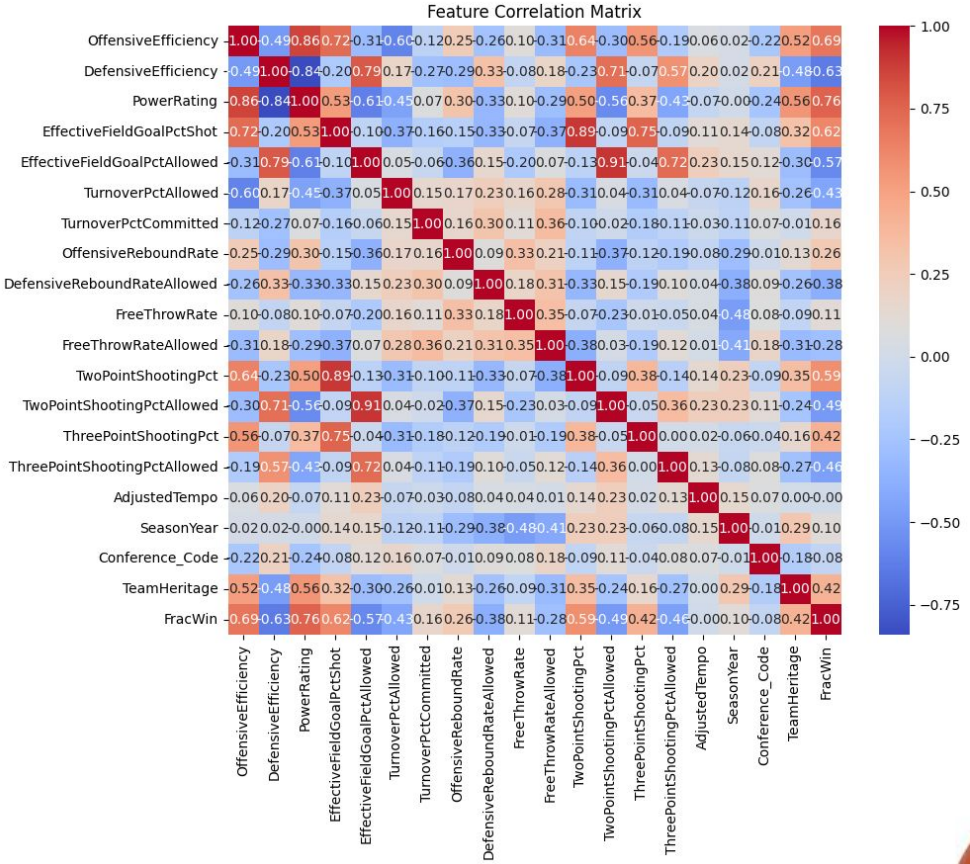


Figure 9: HeatMap of dataset

Models + Results

First Approach: K-nearest-neighbors

- **Reason for using KNN**
  - Simple model compared to other statistical models
  - Problem was a one-time analysis compared to real-time analysis
  - Highly interpretable due to nearest-neighbors being extractable.
- **Concerns**
  - As data increases, so does computation time
  - Feature importance is not reflected in the KNN model
- **Hyperparameters**
  - k = amount of neighbors
  - d = distance metric

Distance Metric 1: Minkowski

- **Added Hyperparameter**
  - p = coefficient related to the distance metric
- **Generalization of multiple distance metrics**
  - When p = 1, Manhattan distance
  - When p = 2, Euclidean distance
  - When lim p → ∞, Chebyshev distance
- **Results**
  - Best hyperparameters: k = 12, p = 1
    - Manhattan on k = 12
  - Prediction on 2024 data: **91.99%** accuracy
  - 10 - fold cross validation: **90.34%** accuracy
  - ROC curve: AUC = **0.92**
  - F1 score = **0.72**

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Figure 10: Minkowski Distance [1]

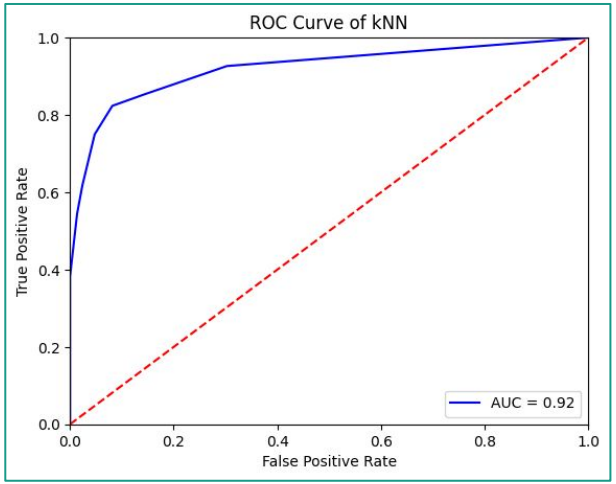


Figure 11: ROC Curve for Minkowski

Distance Metric 2: Hassanat [2]

- **Alternative to normalization**
  - Range [0,1] → invariant to scale
  - Asymptotic → resistant to noise and outliers
- **Many diverse applications**
  - Facial Recognition
  - Lung Nodule Detection
- **Results**
  - Best model: k = 12
  - Prediction on 2024 data: **93.10%** accuracy
  - 10 - fold cross validation: **92.30%** accuracy
  - ROC curve: AUC = **0.96**
  - F1 score = **0.82**

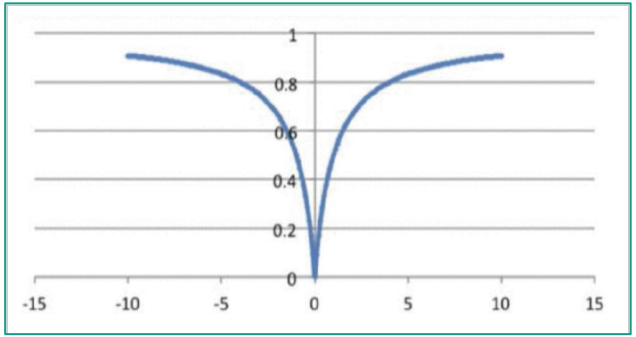


Figure 12: Hassanat Distance Graphed [2]

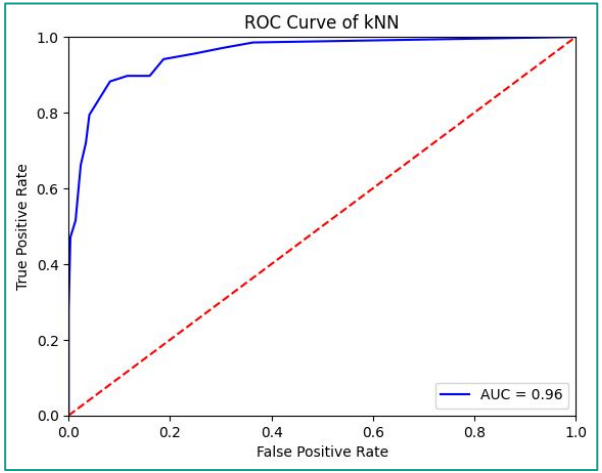


Figure 13: ROC Curve for Hassanat

Second Approach: Logistic Regression

- **Reason for using Logistic Regression**
  - Probabilistic Interpretation (P[0] vs. P[1])
  - High Interpretability
    - Allows for feature importance
- **Concerns**
  - Sensitive to Outliers
  - Assuming that each instance is independent of one another
  - Assumes linear relationship (linear boundaries + No Multicollinearity)
- **Hyperparameters**
  - C = Inverse of regularization strength
    - Smaller values of C → stronger regularization
    - [0.01, 0.1, 1, 10, 100, 1000]

Results

- **Best Parameters**
  - C = 10
- **Results**
  - Prediction on 2024 data: **90.60%** accuracy
  - 10 - fold cross validation: **92.25%** accuracy
  - ROC curve: AUC = **0.9491**
  - F1 score = **0.7424**

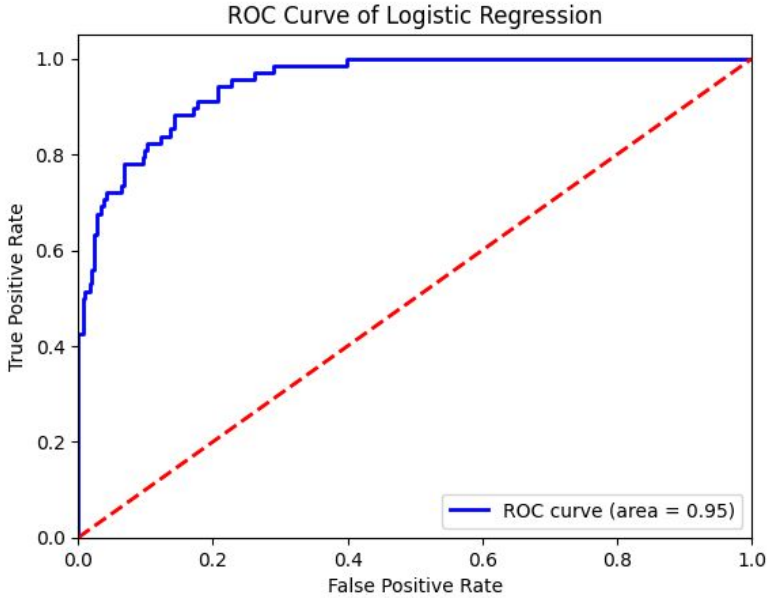


Figure 14: ROC Curve for LR

Feature Importance - 1

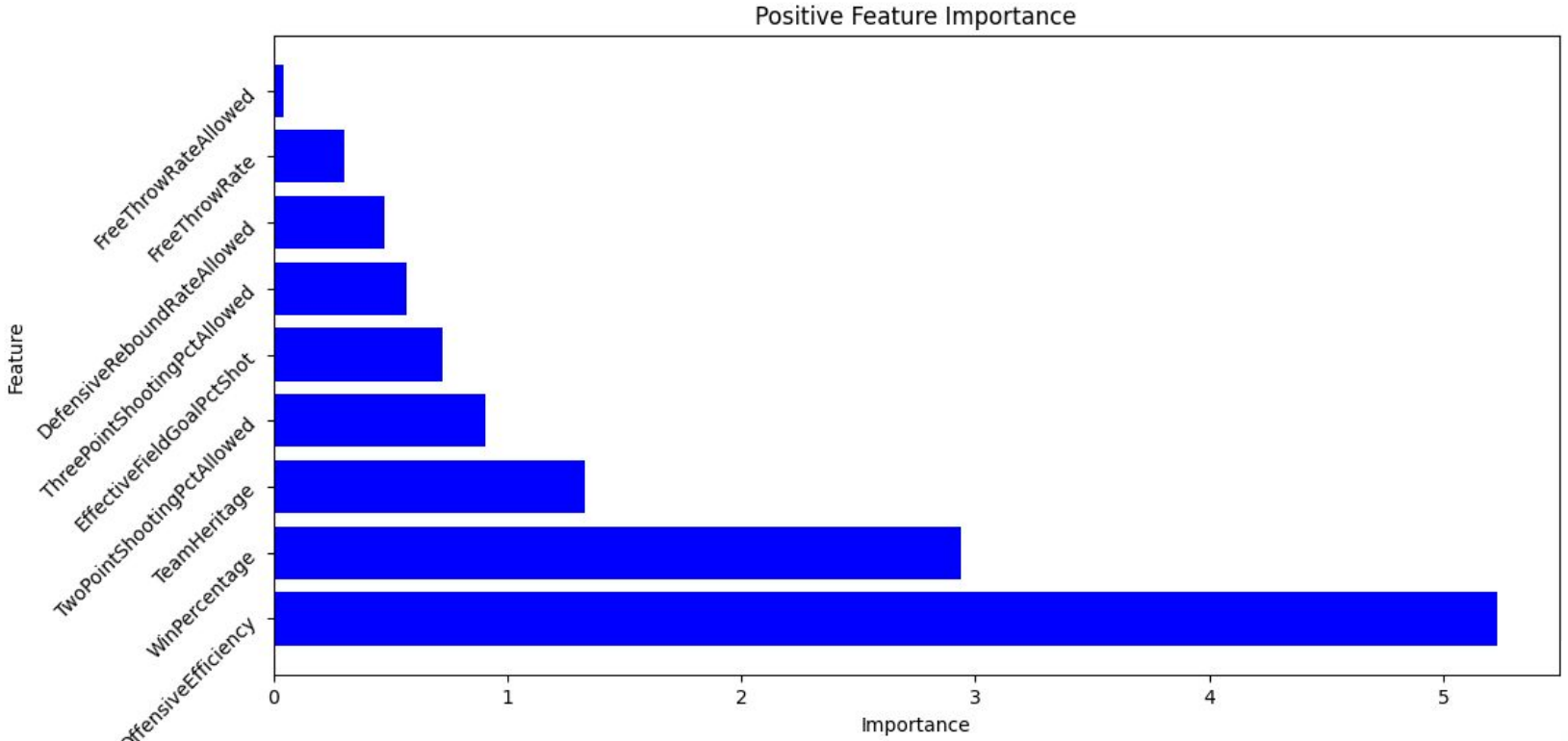


Figure 15: Positive Feature Importance for LR

Feature Importance - 2

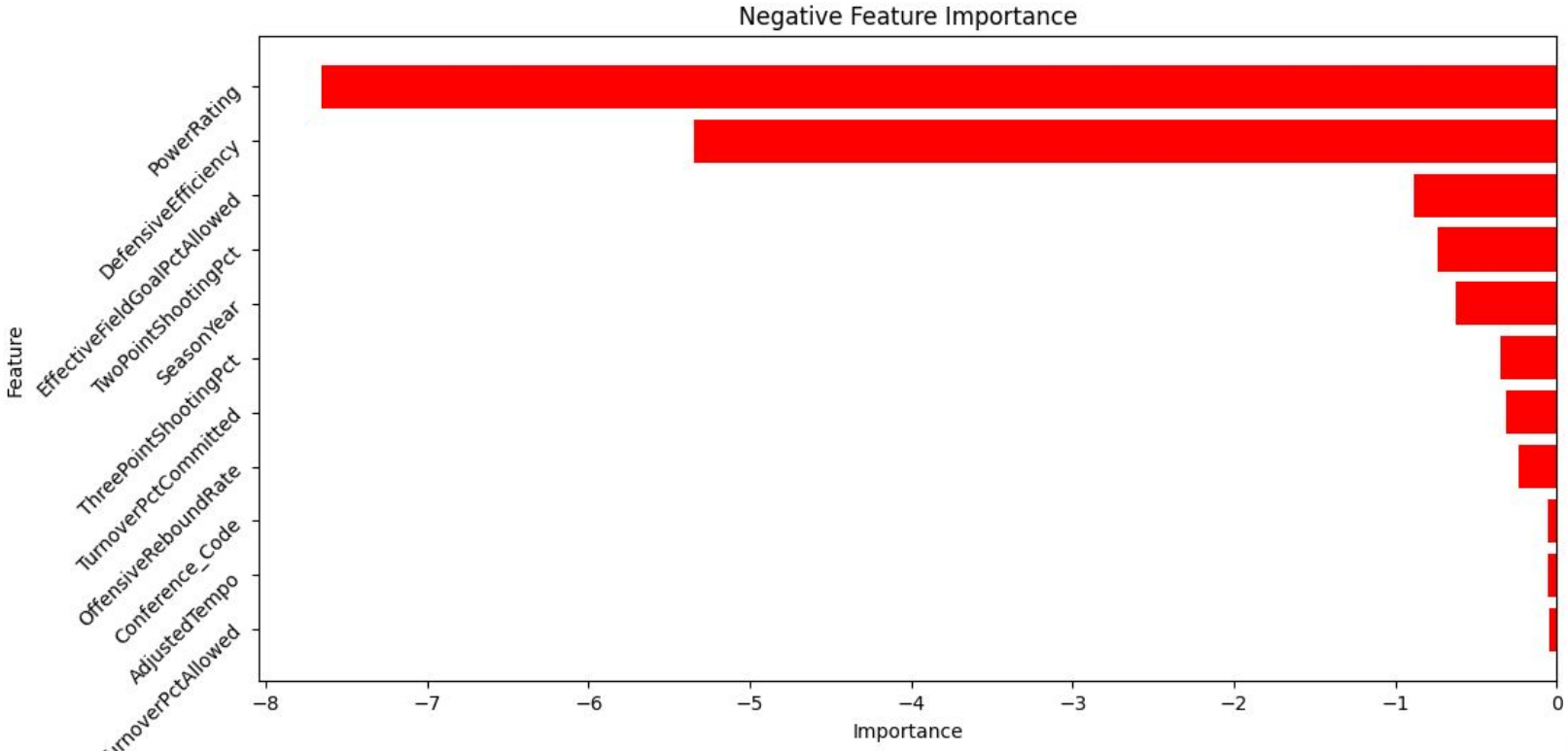


Figure 16: Negative Feature Importance for LR

Third Approach: SVM

- **Reason for using SVM**
  - Effective in high dimensional spaces
  - Memory efficient
  - Versatile
- **Concerns**
  - Expensive to find probability estimate
  - Not suitable for noisy data
  - Sensitive to feature scaling
- **Hyperparameters**
  - Kernel Function → Defines data transformation method
  - C → Regularization strength, controls trade-off
  - Gamma → Influences single training example's reach

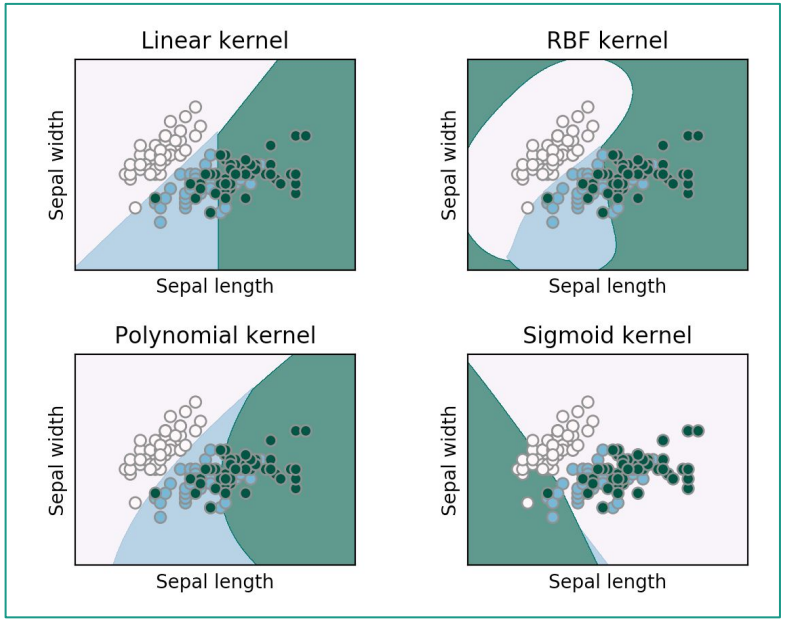


Figure 17: Classification with Different Kernels [4]

Different Kernels

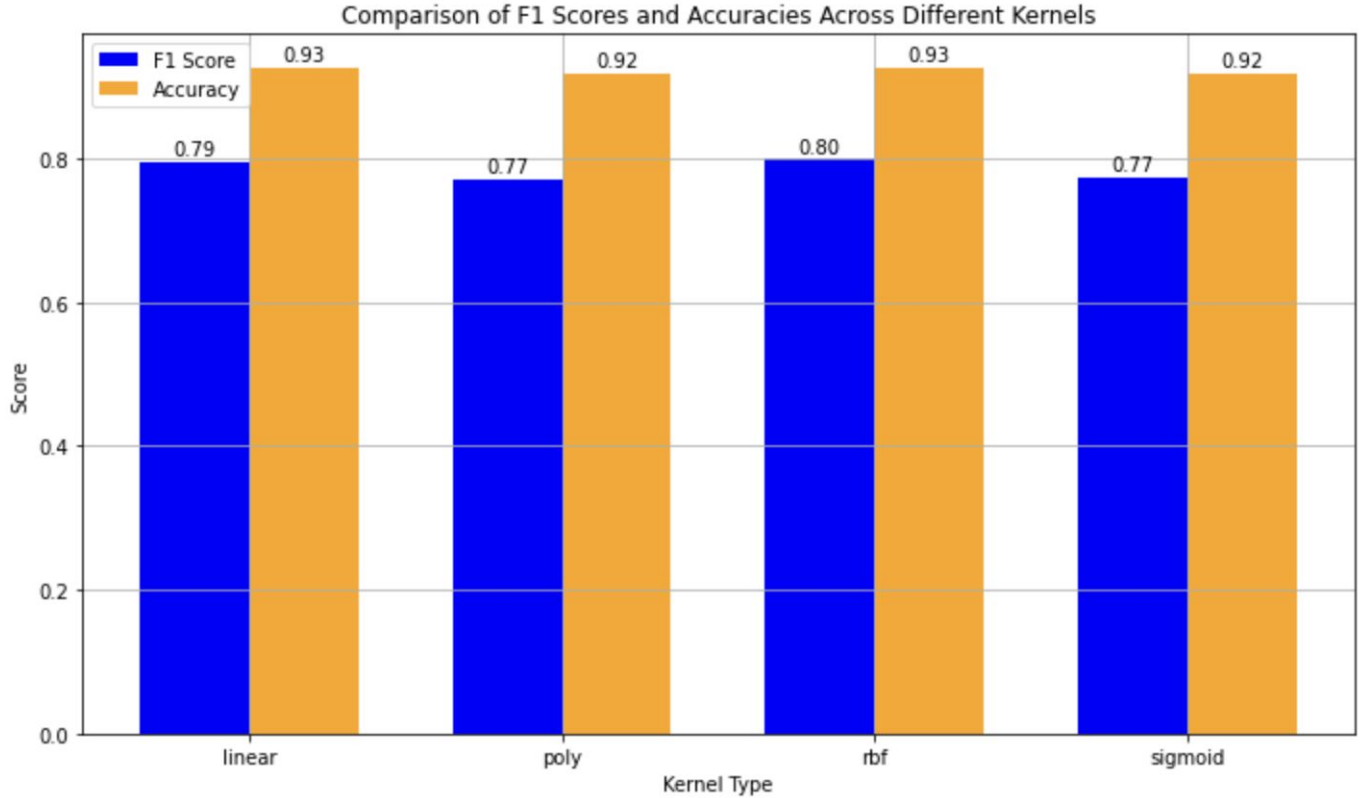


Figure 18: Performance of different kernels

Results

- **Best Parameters**
  - Kernel = rbf
  - C = 100
  - Gamma = auto
- **Results**
  - Prediction on 2024 data: **88.95%** accuracy
  - 10 - fold cross validation: **92.67%** accuracy
  - ROC curve: AUC = **0.95**
  - F1 Score: **0.7333**

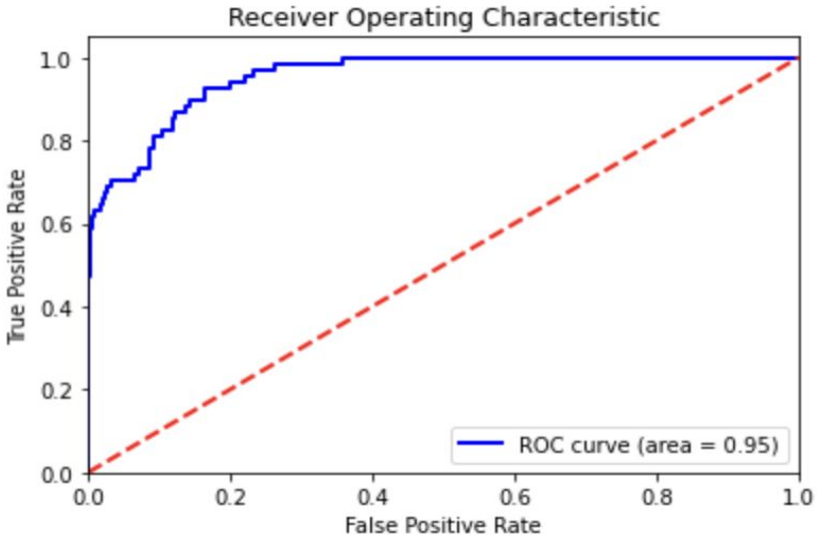


Figure 19: ROC Curve for Best Classification

Feature Importance

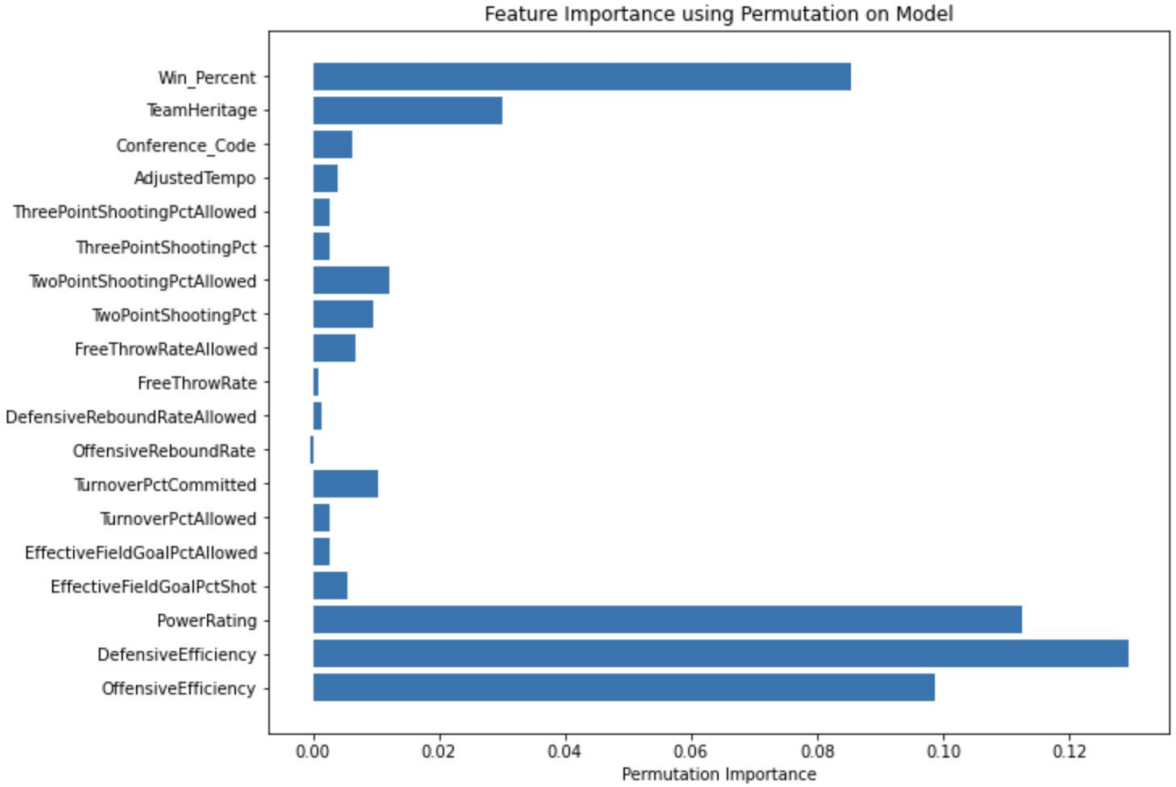


Figure 20: Feature Importance for SVM

Decision Tree Ensembles: Random Forest, XGBoost

- **Reason for using Decision Tree Ensembles**
  - Non-linear decision boundaries
  - Provides feature importance
  - Insensitive to feature scaling
  - Ensemble methods help counteract overfitting/high variance
  - Feature selection less important
- **Concerns**
  - Ensemble methods less interpretable than standard decision trees
- **Hyperparameters**
  - Random Forest: Number of estimators, max depth
  - XGBoost: Number of estimators, max depth, learning rate

Random Forest Results

- **Best Parameters**
  - n_estimators = 400
  - Max depth = 8
- **Results**
  - Prediction on 2024 data: **91.7%** accuracy
  - 10 - fold cross validation: **91.9%** accuracy
  - ROC curve: AUC = **0.96**
  - F1 score = **0.77**

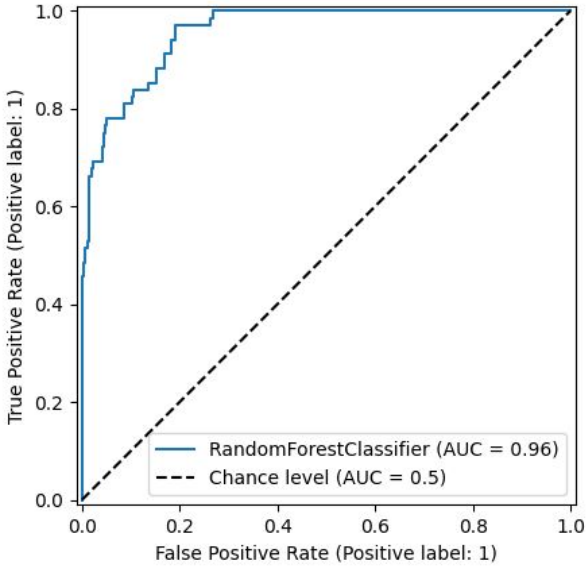


Figure 21: ROC Curve for Random Forest

XGBoost Results

- **Best Parameters**
  - n_estimators = 500
  - Max Depth = 4
  - Learning Rate = 0.01
- **Results**
  - Prediction on 2024 data: **91%** accuracy
  - 10 - fold cross validation: **93.5%** accuracy
  - ROC curve: AUC = **0.92**
  - F1 score = **0.75**

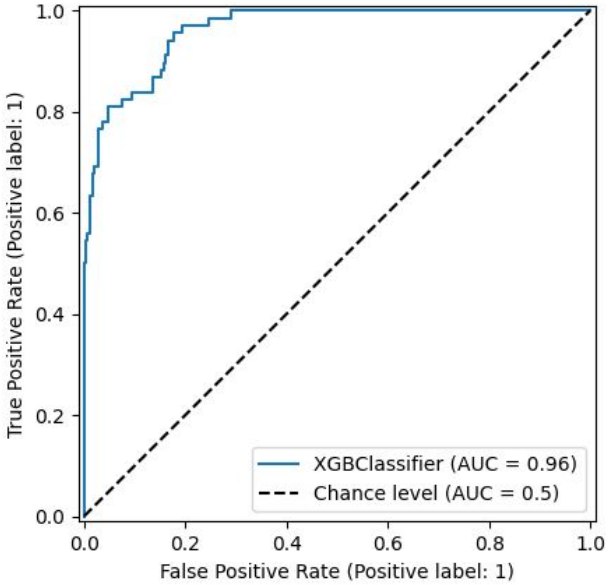


Figure 22: ROC Curve for XGBoost

XGBoost Feature Importance

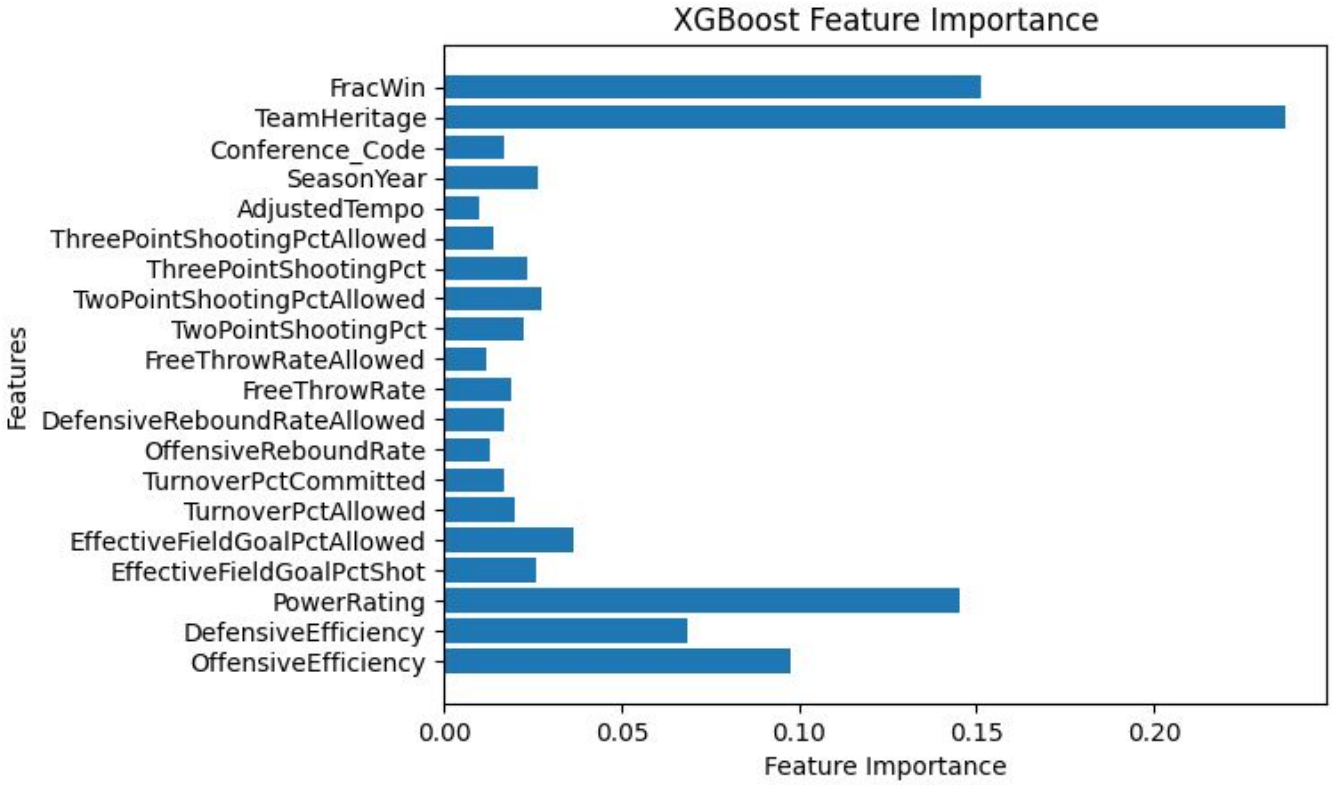


Figure 23: Feature Importance of XGBoost

Results

	KNN	Logistic Regression	SVM	Random Forest	XGBoost
Accuracy	**93.10%**	90.60%	88.95%	91.70%	91.00%
10-CV	92.30%	92.25%	92.67%	91.90%	**93.50%**
AUC	0.96	0.95	0.95	0.96	0.96
F1	**0.82**	0.74	0.73	0.77	0.75

Insights Gained

- Team's offensive ability is more important than defensive ability
  - Since offensive ability has higher variance, it is seen as better for the model
- Team Heritage did seem to play a factor in classification
  - One of the qualitative metrics the committee uses to select
- Free throws not as significant
  - They are considered a bare minimum to have
- Not allowing shots is more important than taking the shots itself

What we are currently working on

- Running KNN using only the columns defined by feature importance of other models
- Ensemble Model
  - KNN
  - SVM
  - Logistic Regression

References

[1] Sharma, P. (2024) *Understanding distance metrics used in machine learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/ (Accessed: 11 April 2024).
[2] A. Hassanat, E. Alkafaween, A. S. Tarawneh and S. Elmougy, "Applications Review of Hassanat Distance Metric," *2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA)*, Karak, Jordan, 2022, pp. 1-6, doi: 10.1109/ETCEA57049.2022.10009844.
[3] Sundberg, A. (2024) *College basketball dataset*, *Kaggle*. Available at: https://www.kaggle.com/datasets/andrewsundberg/college-basketball-dataset/data?select=cbb23.csv (Accessed: 14 April 2024).
[4] S, P. (2023, November 16). The A-Z guide to support vector machine. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/

References

[5] NCAA.org. (2022, February 16). March Madness brand will be used for DI Women's Basketball Championship. https://www.ncaa.org/news/2021/9/29/general-march-madness-brand-will-be-used-for-di-womens-basketball-championship.aspx

[6] Statista Research Department, (2024, March 12). NCAA March Madness appearances by School/Team 2023. Statista. https://www.statista.com/statistics/219636/ncaa-basketball-tournament-appearances/

[7] List of NCAA Division I Basketball Conferences by classification. (n.d.). https://www.researchgate.net/figure/List-of-NCAA-Division-I-Basketball-Conferences-by-Classification_tbl1_254415765

[8] NCAA men's final sees viewership increase over 2023, but not enough to outdraw women's title game. U.S. News. (2024, April 9). https://www.usnews.com/news/sports/articles/2024-04-09/ncaa-mens-final-sees-viewership-increase-over-2023-but-not-enough-to-outdraw-womens-title-game

Zero-shot LLM Generated Text Detection

Yongye Su, Xiyuan Chen, Wei Fan, Hao Wang

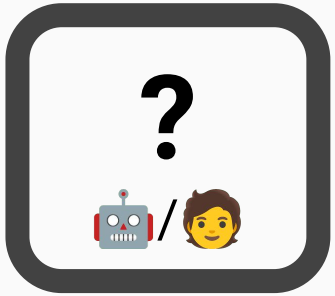
{su311, chen4851, fan370, wang5730}@purdue.edu

Content overview

- Problem Statement
- Model Evaluation
  - Transformer-based (BERT)
  - ML-based (Logical regression, SVM, Gaussian-bayesian optimization, Random Forests)
  - LLM-based (DetectGPT, Single-revise)
- Conclusion and Next Steps

Problem statement:
If LLMs are the answer, what's the question for us?

It is becoming important to distinguish between **human-written** and **machine-generated text**, since large language models are becoming more ubiquitous and embedded in different user-facing services, to verify the authenticity of news articles, product reviews, etc.



Human-written

Machine-generated text

Dataset:

Hello-SimpleAi HC3

https://huggingface.co/datasets/Hello-SimpleAI/HC3

Datasets: Hello-SimpleAI / HC3 ♡ like 161

Tasks: Text Classification | Question Answering | Sentence Similarity | +1 Languages: English | Chinese Size Categories: 10K<n<100K

Tags: ChatGPT | SimpleAI | Detection | OOD | Croissant License: cc-by-sa-4.0

Dataset card | Viewer | Files and versions | Community 4

Dataset Viewer

Auto-converted to Parquet | API | View in Dataset Viewer

Subset (6): all · 24.3k rows

Split (1): train · 24.3k rows

Search this dataset

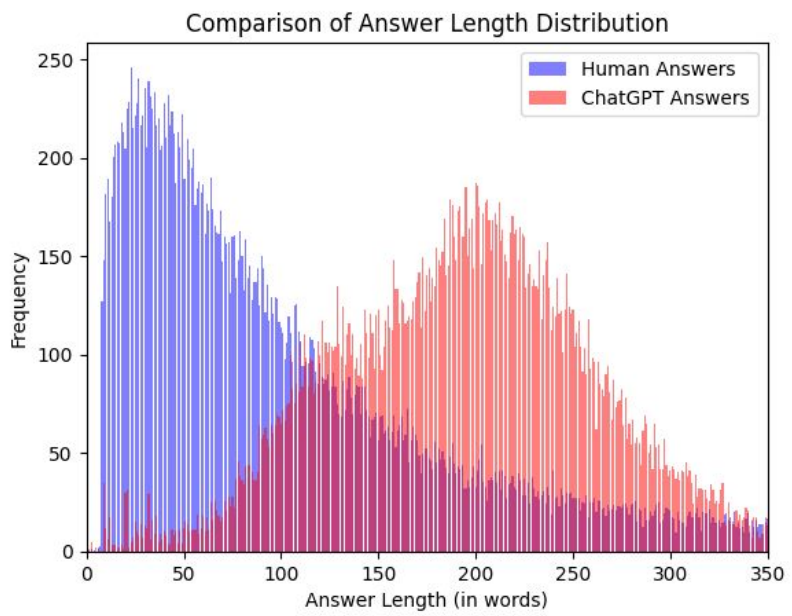
id string · *lengths* 1 5	question string · *lengths* 13 611	human_answers sequence	chatgpt_answers sequence	source string · *classes* 5 values
0	Why is every book I hear about a " NY Times # 1 Best…	["Basically there are many categories of \" Best Seller…	["There are many different best seller lists that are…	reddit_eli5
1	If salt is so bad for cars , why do we use it on the road…	["salt is good for not dying in car crashes and car…	["Salt is used on roads to help melt ice and snow and…	reddit_eli5
2	Why do we still have SD TV channels when HD looks like…	["The way it works is that old TV stations got a certai…	["There are a few reasons why we still have SD (standard…	reddit_eli5
3	Why has nobody assassinated Kim Jong - un He is such a…	["You ca n't just go around assassinating the leaders of…	["It is generally not acceptable or ethical to…	reddit_eli5
4	How was airplane technology able to advance so quickly…	["Wanting to kill the shit out of Germans drives…	["After the Wright Brothers made the first powered flight…	reddit_eli5
5	Why do humans have different colored eyes ? What causes /…	["Melanin ! Many of the the first known humans existed i…	["The color of your eyes is determined by the amount and…	reddit_eli5

‹ Previous 1 2 3 ... 244 Next ›

Corpus Analysis

Comparison of Answers' Length Distribution

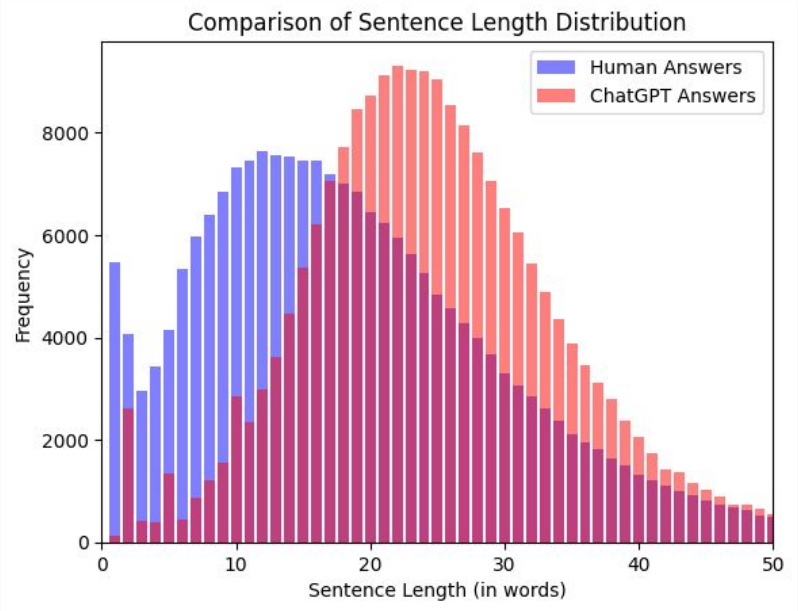
ChatGPT tend to generate longer answers compared to those written by humans！



Corpus Analysis

Comparison of Sentences' Length Distribution

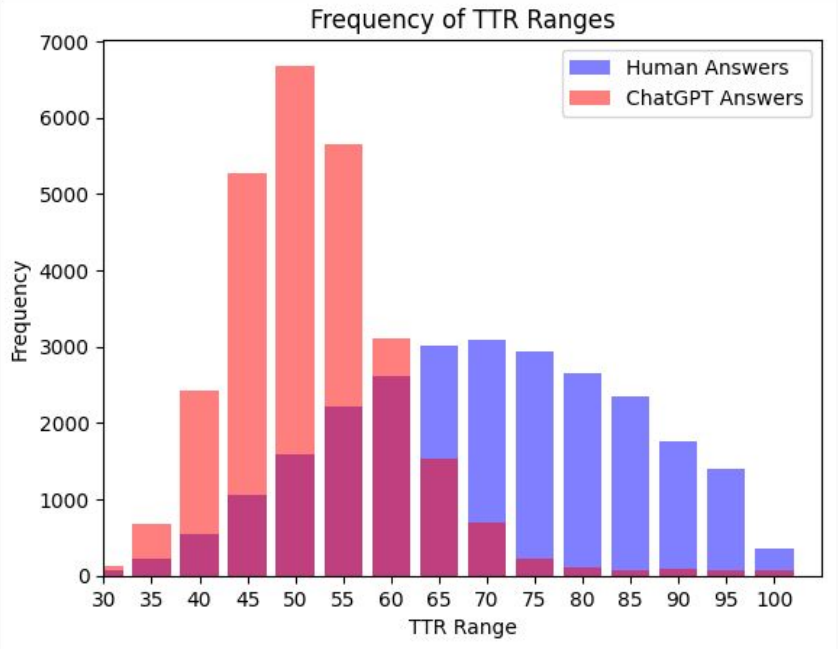
The answers generated by ChatGPT tend to have more words in a sentence compared to those written by humans！



Corpus Analysis

Comparison of lexical richness.

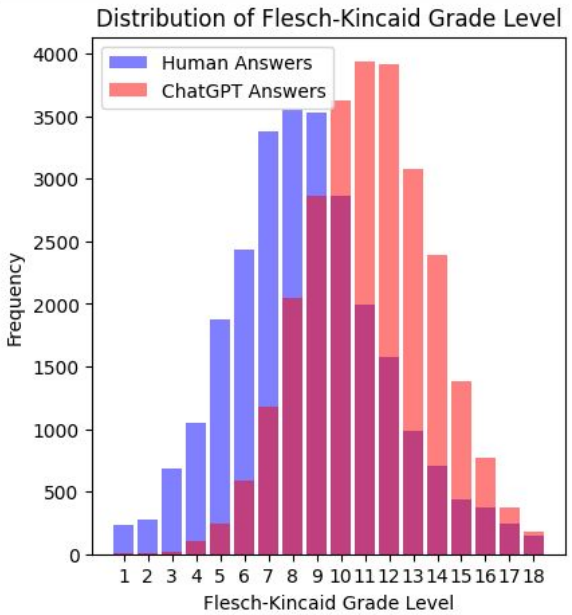
Humans tend to use more diverse range of vocabulary compare to the answer generated by ChatGPT.



Corpus Analysis

Comparison of vocabulary difficulty.

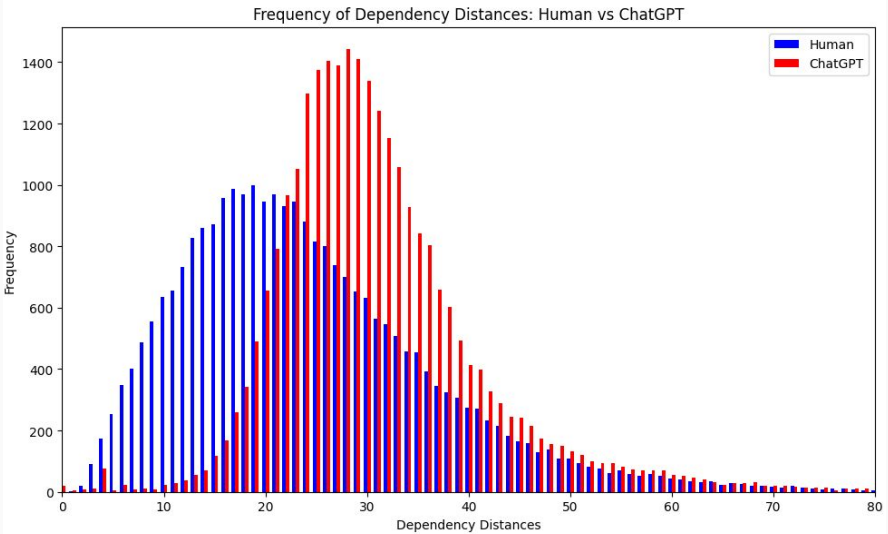
The answers generated by ChatGPT tend to use more difficult words in a sentence compared to those written by humans ！



Corpus Analysis

Comparison of sentence structure complexity

ChatGPT tends to use more complicated sentence structure compare to the answer written by human.



Model Evaluation: Part A ML Based

Gaussian Naive Bayes: Explain with a use case of predicting continuous values.

Logistic Regression: Describe its use in binary classification problems.

Random Forests: Highlight their utility in feature-rich datasets for classification and regression.

Support Vector Machines (SVM): Discuss the application in high-margin classification.

Using Random Forest for Detection

Using Random Forest for Detection Language Model Generated Text

Ensemble learning method that combines multiple decision trees.

Each tree in the ensemble is built from a sample drawn with replacement from the training set.

Decision at each node is made by majority votes from all the trees.

Precision: 0.88 Recall: 0.83 f1-Score: 0.85

Using Gaussian Naive Bayes for Detection

Performs well with a large feature space, as often encountered in text classification

Used with features extracted through methods that assume a Gaussian distribution

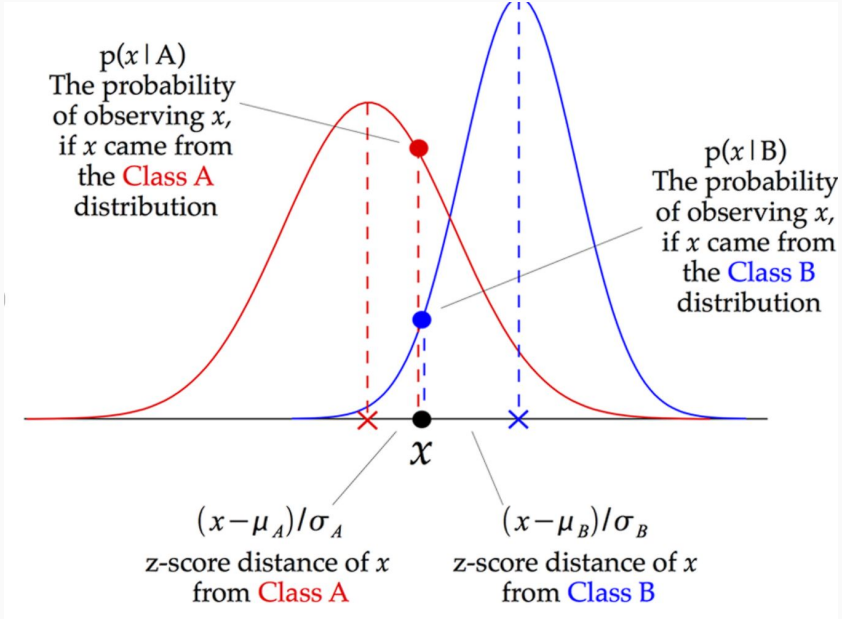
The logic behind using probabilities to classify texts and the benefits of its simplicity when dealing with complex natural language data.

Handling text data where independence assumptions may not hold

Dealing with skewed data distributions and their impact on model accuracy

Precision: 0.96 Recall: 0.81 f1-Score: 0.87

[1]https://www.researchgate.net/figure/Illustration-of-how-a-Gaussian-Naive-Bayes-GNB-classifier-works-For-each-data-point_fig8_255695722



Using Support Vector Machine for Detection

Effectiveness in high-dimensional spaces, as text data often is once vectorized.

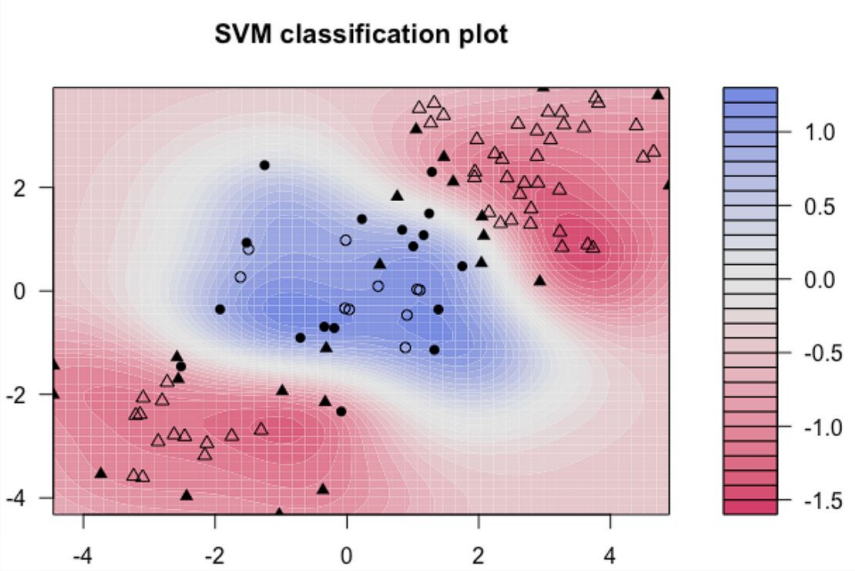
Robust against overfitting, especially in high-dimensional spaces.

Choice of kernel and its impact on the classification accuracy.

Computational cost in training for large datasets.

Precision: 0.97 Recall: 0.97 f1-Score: 0.97

[1]https://afit-r.github.io/svm



Using Logistic Regression for Detection

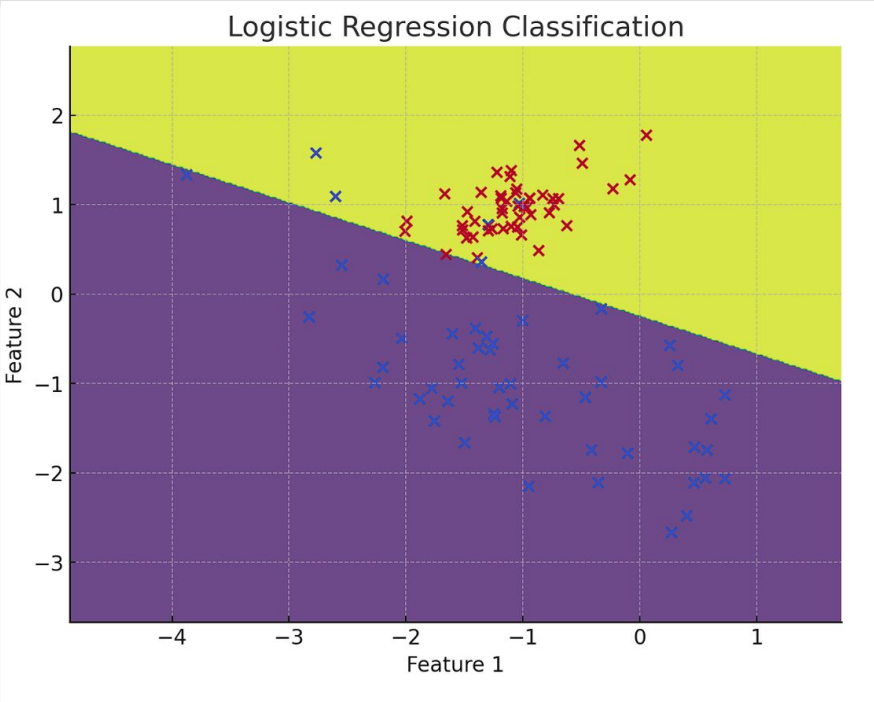
Simplicity and interpretability

Efficiency in training

Dealing with non-linear relationships that Logistic Regression can't capture

Handling high-dimensional sparse data

Precision: 0.86 Recall: 0.84 f1-Score: 0.85



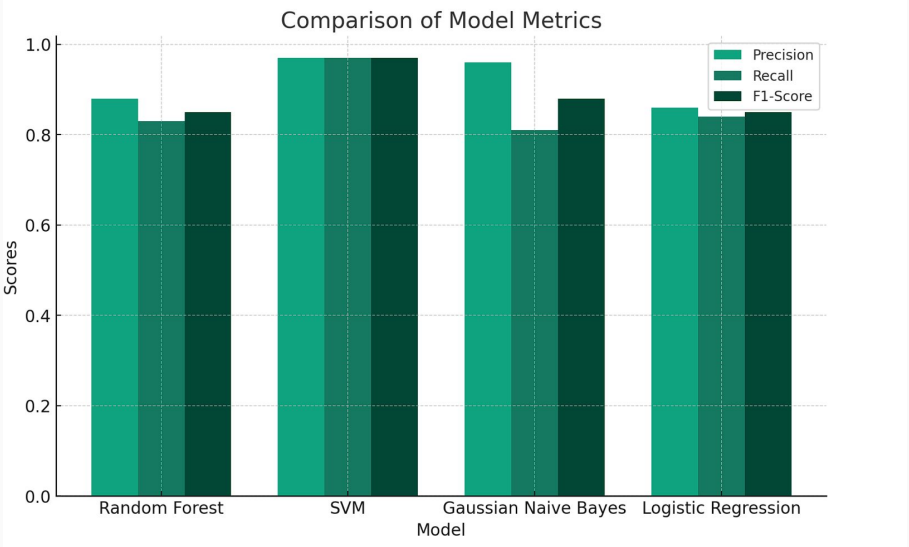
Model Evaluation: Part A ML Based

Random Forest shows a balanced performance with relatively high precision and recall. The model is good at correctly predicting positive cases while maintaining a lower rate of false negatives.

SVM performs exceptionally well in this experiment, with high precision and recall. It is highly accurate in predicting positive cases and has a very low rate of missing actual positive cases.

Gaussian Naive Bayes exhibits high precision but slightly lower recall compared to SVM. This model is good at predicting positive cases correctly, but it misses a higher proportion of actual positives

Logistic Regression shows a balanced but slightly lower performance across all metrics compared to SVM and Gaussian Naive Bayes



Model Evaluation: Part B
Transformer based

Attention is All You Need — 2017, Google

- A breakthrough in DL architectures, particularly in natural language processing (NLP)
- Process entire input sequences in parallel
- Self-attention mechanism

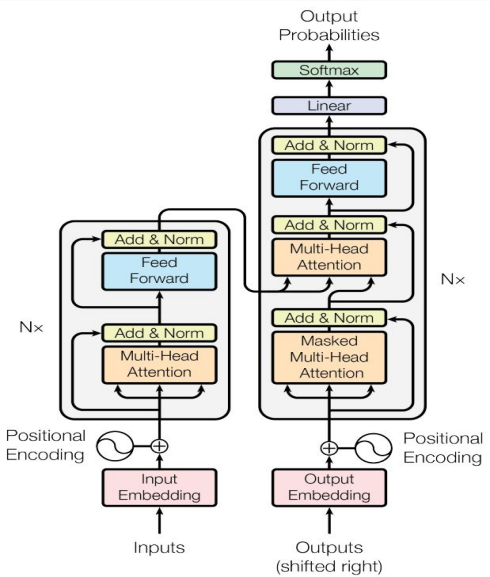


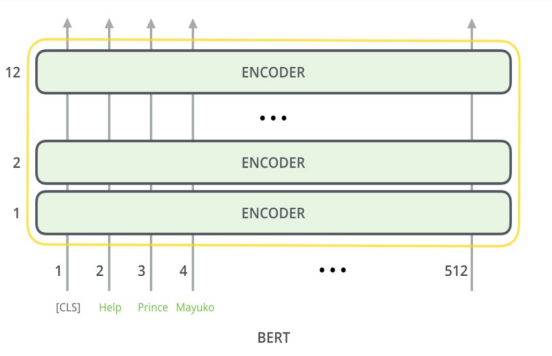
Figure 1: The Transformer - model architecture.

https://arxiv.org/pdf/1706.03762.pdf

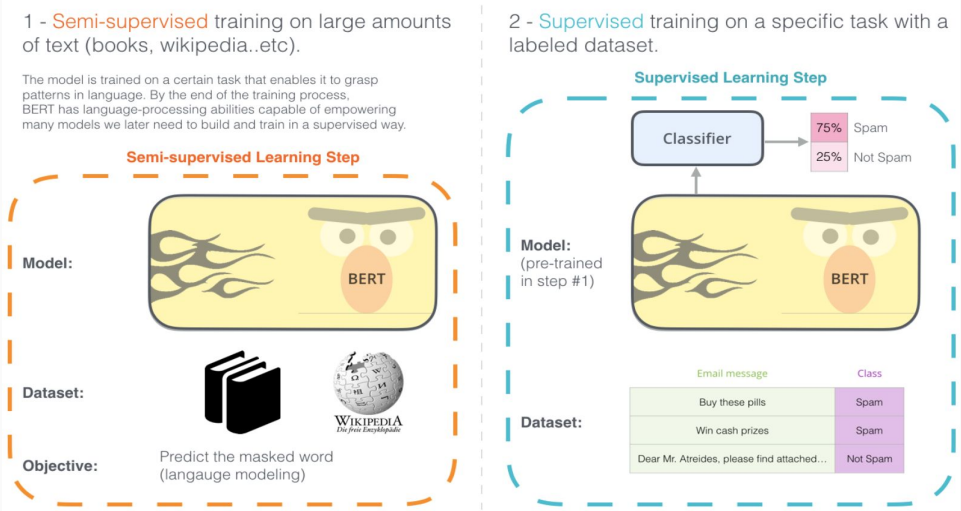
Model Evaluation: Part B
Transformer based

Bidirectional Encoder Representation from Transformers (BERT)

- Masked language modelling
- Encoder only



https://jalammar.github.io/illustrated-bert/



Model Evaluation: Part B
Transformer based

Attention

- Key/value/query concepts originate from retrieval systems like search engines

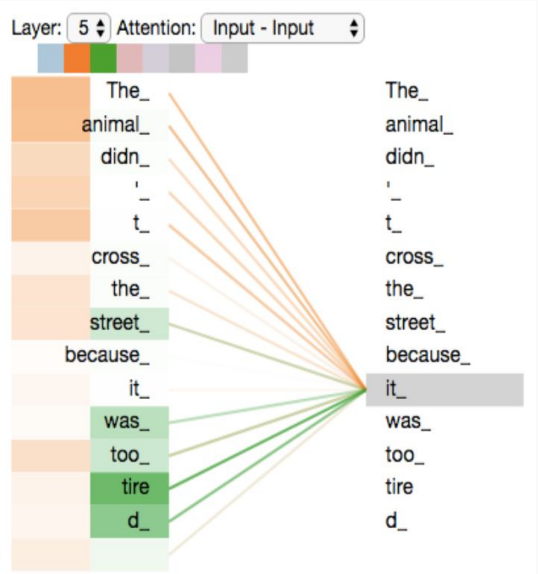
Scaled dot-product attention takes three matrices as input

Similarity is simply the dot product between Q and K

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The output is simply a scaling of V

A softmax normalizes similarities → [0, 1]



https://jalammar.github.io/illustrated-transformer

Model Evaluation: Part B
Transformer based

Experiment

- model_type: distilbert-base-uncased
- n_heads: 12
- n_layers: 18
- vocab_size: 30522

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.028900	0.065312	0.988180	0.986382	0.982035	0.991095
2	0.016200	0.071421	0.987537	0.985650	0.981038	0.990677
3	0.009900	0.021136	0.997308	0.996872	0.996469	0.997277
4	0.007500	0.018856	0.995202	0.994435	0.993056	0.995847
5	0.005000	0.013109	0.997776	0.997416	0.996963	0.997873

Model Evaluation: Part B
Transformer based

Are we done?

```
1 print(is_human("I'm ChatGPT, an AI developed by OpenAI. My purpose is to assist you with a wide range of tasks, whether it's answering questions, generating text, providing recommendations, or just having a friendly conversation. I'm trained on a diverse dataset covering a multitude of topics, so feel free to ask me anything you'd like to know or discuss!"))
2
3 print(is_human("I'm \"ChatGPT\", an AI developed by OpenAI. 1.\nMy purpose is to assist you with a wide range of tasks, whether it's answering questions, \"generating text\", providing recommendations, or just having a friendly conversation. \nI'm trained on a diverse dataset covering a multitude of topics, so feel free to ask me anything you'd like to know or discuss!" ))
```

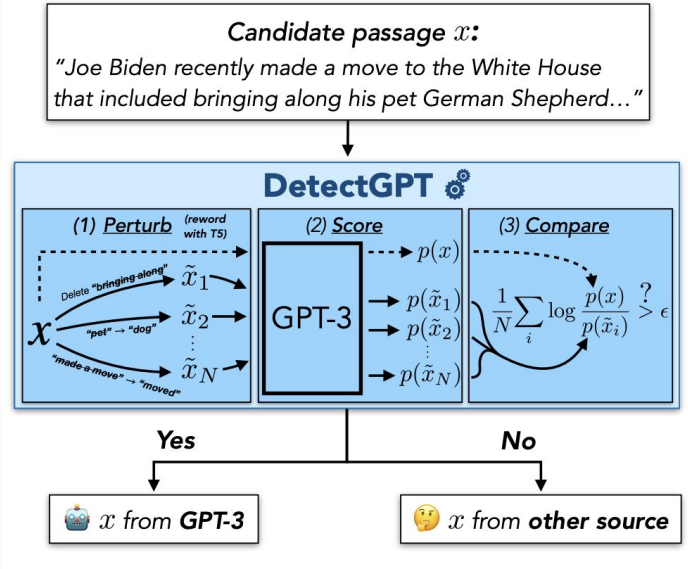
Python

```
True
False
```

Zero-shot Solution: Part C
LLM based - DetectGPT: Structure and ideas

While LLMs are generating texts, why can't we give them back their own way?

DetectGPT [1] as shown on the right part, developed a brand new zero-shot way for estimating the confidence (probability) to determine the AI-generated text.

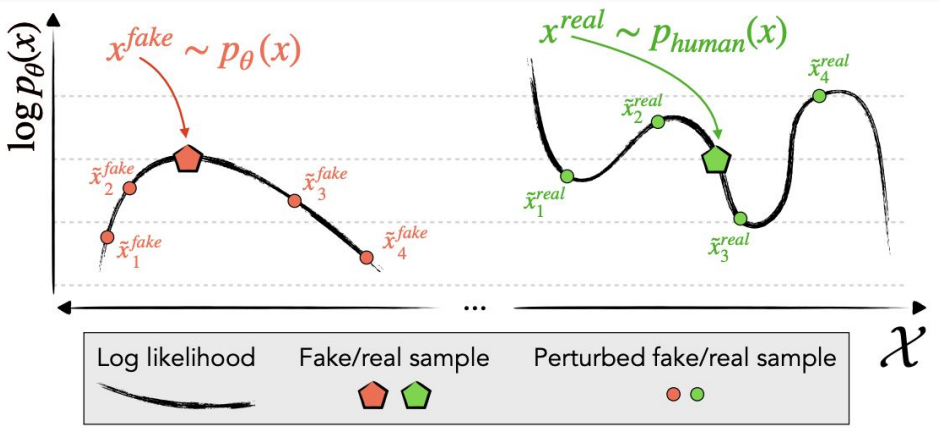


[1] and Figure source: Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, Chelsea Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. *Proceedings of the 40th International Conference on Machine Learning*, PMLR 202:24950-24962, 2023.

Zero-shot Solution: Part C
LLM based - DetectGPT: Hypothesis

Hypothesis: Minor rewrites of model-generated text tend to have lower log probability under the model than the original sample, while minor rewrites of human-written text may have higher or lower log probability than the original sample. [1]

log*pθ(x)*: text with a high average log probability is more likely to have been generated by the target LLM *θ*

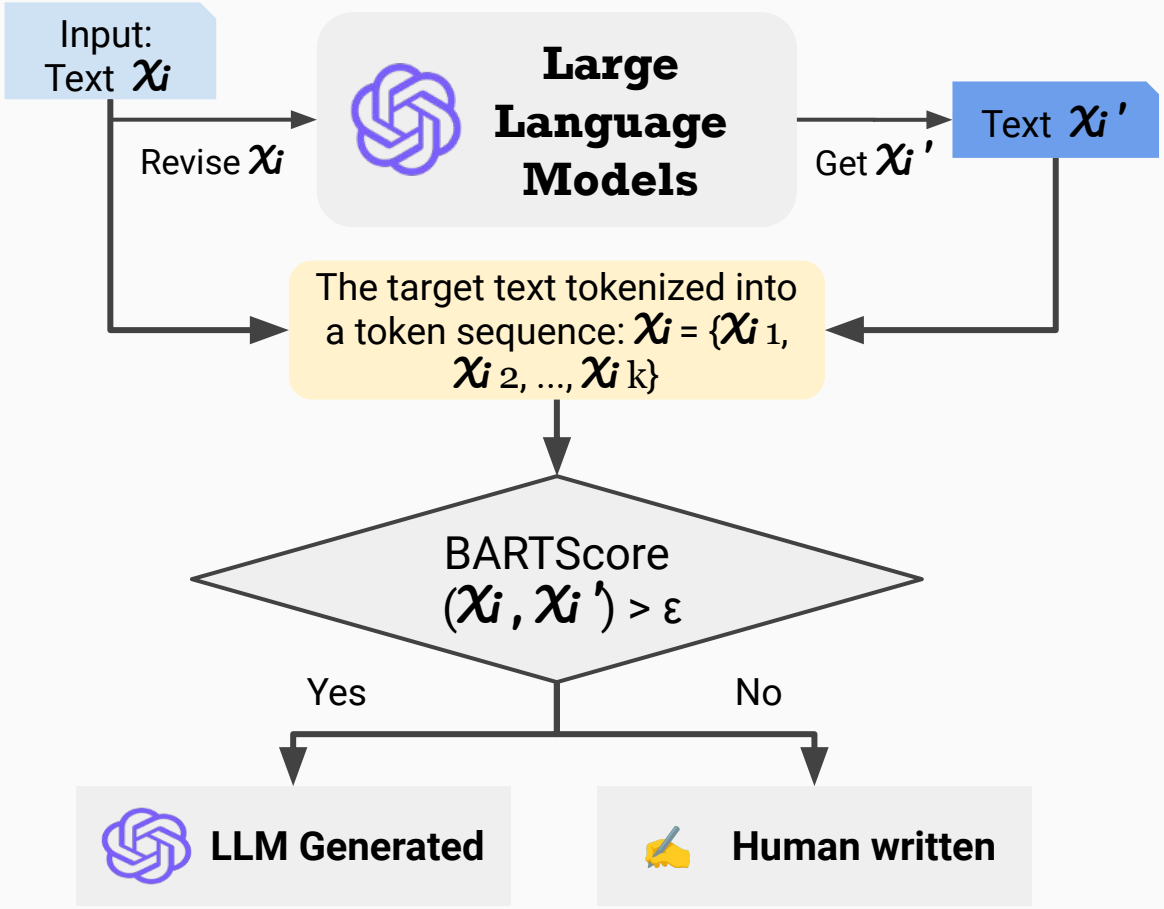


[1] and Figure source: Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, Chelsea Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. *Proceedings of the 40th International Conference on Machine Learning*, PMLR 202:24950-24962, 2023.

Zero-shot Solution Part C: **LLM based - Single-revise: a simpler implementation**

Alternatively, this LLM based seemed much simpler compared with DetectGPT, follows the principle of single-revise.

This Single-revise is based on the prototype with some modifications on: Beat LLMs at Their Own Game: Zero-Shot LLM-Generated Text Detection via Querying ChatGPT (Zhu et al., EMNLP 2023)



Zero-shot Solution: Part C Experiments

	Method name	DetectGPT	Single-revise
Performance	AUROC	0.952	0.943
Efficiency	Avg predict time (t)	8.98 seconds	0.2 seconds

Our Conclusion and Next Steps

- We systematically evaluated threefold kinds of solutions (Transformer, ML, and LLM) with open-sourced dataset collected from Reddit and etc. [1], showcasing the feasibility and limitations of each method, with modifications either in model or pipeline.

- We also plan to explore more and dive deeper into this topic, **combining the advantages of different solutions** and using more **datasets** & **LLMs** to prove the generalizability and efficiency. (e.g. LLaMA, Gemini)

[1] Guo, Biyang, et al. "How close is chatgpt to human experts? comparison corpus, evaluation, and detection." *arXiv preprint arXiv:2301.07597* (2023).
Data available at: https://github.com/Hello-SimpleAI/chatgpt-comparison-detection

Tuning the Autovacuum Knobs of PostgreSQL

Laura Zhou
Shyawn Zahid
Yeasir Rayhan

Introduction

- PostgreSQL is one of the most widely used DB management systems
- It implements Multi-Version Concurrency Control following an append-only scheme
  - Over time, tables get bloated!
- Cleans database periodically with a vacuum procedure.
  - Autovacuum daemon: automatically vacuums a table when the cost function surpasses a threshold.
  - 13 different autovacuum knobs.
- Given a query workload, what is the optimal autovacuum knob settings to maximize the performance, e.g., throughput, tail-latency of a pgsql instance

Problem Definition

Let $\mathcal{P}$ be a pgSQL DB with autovacuum knobs $\theta_1 \ldots \theta_n$ with $\Theta_1 \ldots \Theta_n$ as their domains.

Let $\mathcal{W}$ denote workload and f denote the database metric (ex: throughput, tail-latency) to be optimized during this instance.

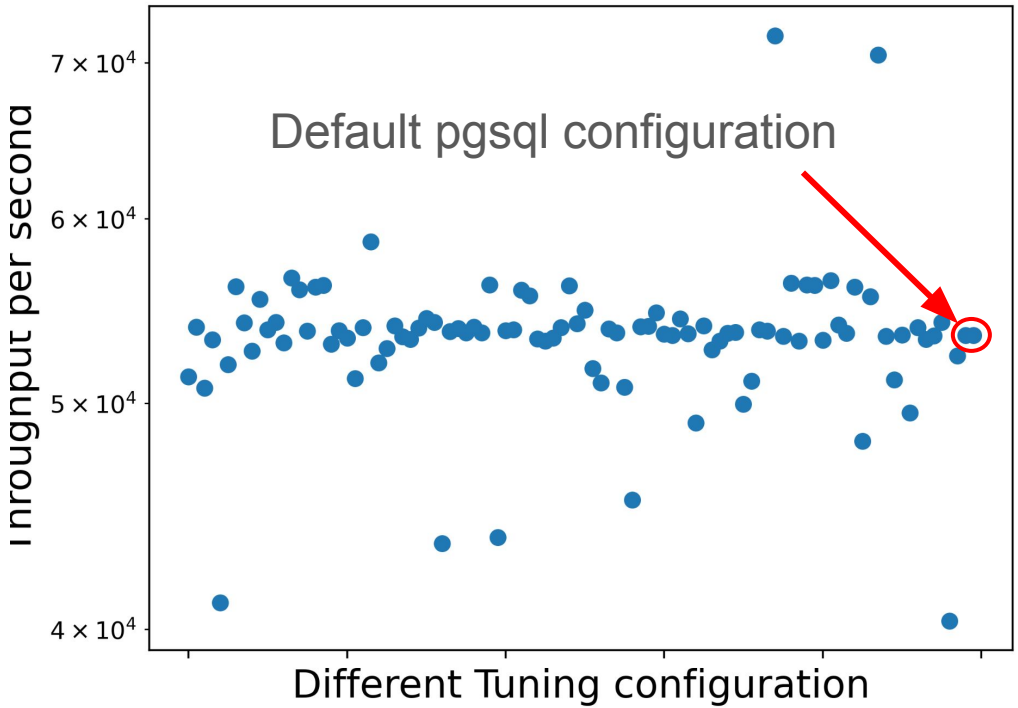
Given a workload, then the goal is to find a configuration θ^* that maximizes the database performance metric.

$$\theta^* = \underset{\theta \in \Theta_1 \times \Theta_2 \ldots \Theta_n}{\arg\max} f(\theta, \mathcal{W})$$

Motivation

- Previously, extensive domain knowledge by database administrators was required for the task of knob tuning.
- More recently, machine learning has been used for knob tuning instead due to increasingly complex database workloads.
- There are dependencies between PostgreSQL autovacuum knobs that cause the task of tuning a single knob at a time to have poor performance.
- As a result, machine learning is a suitable tool for the optimization of autovacuum configurations.

Motivation


Default pgsql configuration
7 × 10^4
6 × 10^4
5 × 10^4
4 × 10^4
Different Tuning configuration

BO: Bayesian Optimization[1]

- Want to optimize a function f: $R^d \rightarrow R$
- Generally, BO is useful when f is expensive to calculate
  - For the knob tuning problem, the database metric function is costly to calculate, making BO a good option for it
- Algorithm:
  1. Choose a **surrogate function** to model the function f, and define its prior.
  2. Given a set of observations (function evaluations), use Bayes rule to obtain the posterior
  3. Use an **acquisition function** α(x), which is a function of the posterior, to decide the next sample point:
     - $x_t = \text{argmax}_x \, \alpha(x)$
  4. Add newly sampled data to the set of observations and goto step #2 till convergence or budget elapses.

[1] Agnihotri & Batra, "Exploring Bayesian Optimization", Distill, 2020.

BO: Bayesian Optimization[1]

- Surrogate model: Gaussian Process
- A Gaussian process is denoted as a function f(x) ~ GP(m, k)
  - m(x) is a **mean function**
  - k(x, x') is a **covariance function** or **kernel**
- For a collection of function values queried at a collection of points x_i: $f(x_1), \dots, f(x_n) \sim N(\mu, K)$
  - $\mu_i = m(x_i)$
  - $K_{ij} = k(x_i, x_j)$

- Acquisition function: Expected Improvement
  - Idea: Choose the next query point as the one which has the highest expected improvement over the current maximum

$$x_{t+1} = argmin_x \mathbb{E}\left(||h_{t+1}(x) - f(x^\star)|| \mid \mathcal{D}_t\right)$$

- For Gaussian Process surrogate:

$$EI(x) = \begin{cases} (\mu_t(x) - f(x^+) - \epsilon)\Phi(Z) + \sigma_t(x)\phi(Z), & \text{if } \sigma_t(x) > 0 \\ 0, & \text{if } \sigma_t(x) = 0 \end{cases}$$

$$Z = \frac{\mu_t(x) - f(x^+) - \epsilon}{\sigma_t(x)}$$

[1] Agnihotri & Batra, "Exploring Bayesian Optimization", Distill, 2020.

Vanilla BO

- Surrogate model: Vanilla Gaussian Process
  - Kernel function is calculated based on the Euclidean distance between two knob configurations

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

Equation from Wikipedia: Radial basis function kernel

Mixed Kernel BO [2]

- Surrogate model
  - A mixed-kernel GP
- Continuous knobs.
  - Matérn kernel.
    - A continuous kernel that generalizes the RBF with a smoothing parameter
- Categorical Knobs.
  - Hamming kernel
    - Based on hamming distance, suitable to measure the distance between categorical variables.

[2] Aaron Klein. RoBO : A Flexible and Robust Bayesian Optimization Framework in Python. In 2017.

SMAC (Sequential Model-based Algorithm Configuration) [3]

- Surrogate Model.
  - Random Forest. SMAC assumes a Gaussian model N ($y \mid \mu, \sigma^2$), where the μ and σ^2 are the mean and variance of the random forest.
    - A set of B (=10) regression trees are constructed on n data points randomly sampled with repetitions from the entire training dataset.
    - At each split point of each tree, a random subset of d*p(=5/6) knobs are considered eligible to be split upon
    - The minimal number of data points required to be in a node for it to consider splitting, n_{min}=10
  - The predictive mean μ and variance σ^2 for a new configuration θ is computed as the empirical mean and variance of its individual trees' predictions for θ.

[3] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In LION, 2011.

SMAC (Sequential Model-based Algorithm Configuration) [3]

- Evaluating Promising configurations.
  - It uses the model's predictive distribution to compute the expected maximum improvement EI(**ϴ**)
  - EI (**ϴ**) is large for
    - **ϴ** with low predicted cost (Exploitation)
    - **ϴ** with high predicted uncertainty (Exploration)
- Locating Promising Configurations.
  - Compute EI for all the configurations used in the previous run, and pick top k
  - Initialize a local search at each of the top k configuration to collect more configurations.

[3] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In LION, 2011.

TPE: Tree-structured Parzen estimator [4]

- Models P(x|y), rather than P(y|x)
  - x = knob, y = loss function
  - Models P(x|y) by tree structured Parzen density estimators
- Maintains 2 surrogate models based on some threshold y*

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases},$$

l(x) = density formed by using observations $f(x^{(i)}) < y^*$
g(x) = density formed by rest of the observations

- Promising candidates: *l(x) / g(x)*

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In NeurIPS, 2011.

TPE: Tree-structured Parzen estimator [4]

- Threshold (y*)
  - y* > best observed f(x)
  - A hyperparameter γ controls it.
    - TPE chooses y* to be some quantile γ of the observed y values
- Used surrogate models
  - Categorical Knobs: Categorical distribution
  - Real-valued / Integer Knobs:
    - Real Space: Truncated Gaussian Mixture
    - Log space: Truncated Gaussian Mixture in log-space

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In NeurIPS, 2011.

GA: Genetic Algorithm / Evolutionary Algorithm[5]

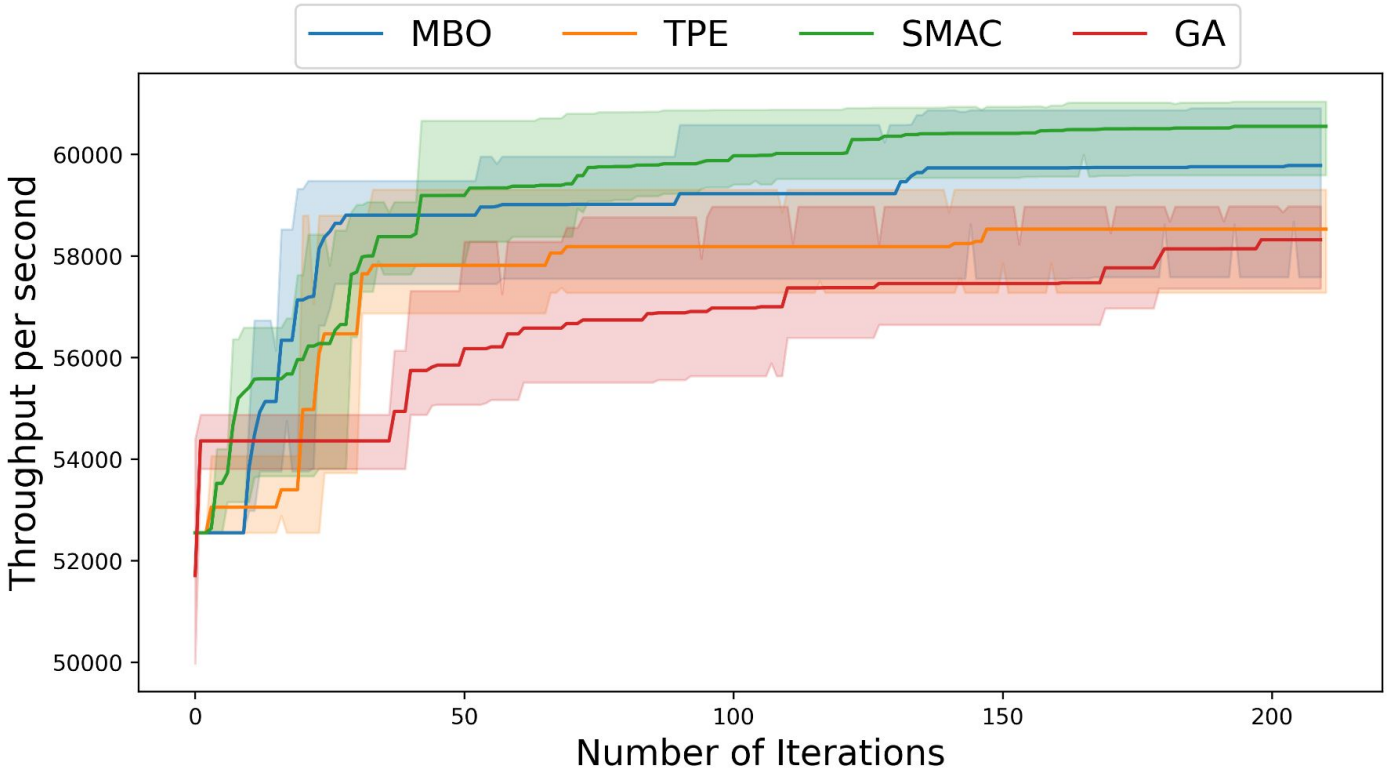
- meta-heuristic inspired by the process of natural selection
- A population P = {θ1, θ2, ..., θN} of N candidate solutions (knob configs) are iteratively evolved toward better solutions
  - The candidates with better fitness (TPS, LAT) will have more chance to be selected, and their (configurations) are reproduced, crossed over, and mutated to form new candidates for the next iteration.
  - Fitness function **f(θi, W)** for each **θi ∈ P**
- Tends to converge towards global optimum
  - Iterate until convergence criteria is met (num_generations, fitness improvement)

[5] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In Proceedings of the workshop on machine learning in high-performance computing environments. 1–5.

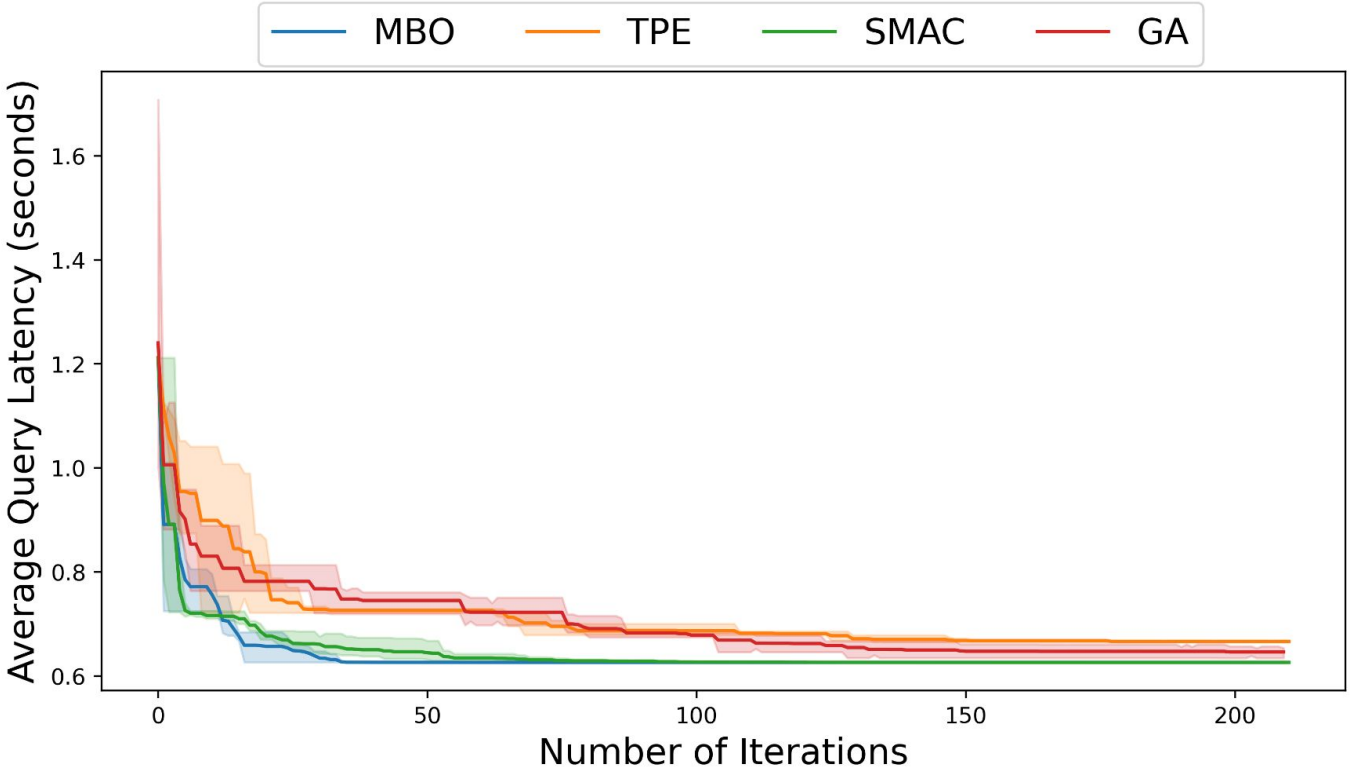
Experiment Settings

- Benchmark
  - Sysbench OLTP Benchmark
    - 1 Table with 10 M entries, num of worker threads = 64
- Workload
  - oltp_update_index
- Objectives
  - Throughput per second (Maximize)
  - Average latency (Minimize)
  - 95th %-tile latency (Minimize)
- Each tuning session is comprised of 210 iterations.
- Surrogate
  - A 2-layer NN to approximate the objective function given a configuration settings.
  - Helps avoid interaction with DBMS in real-time improving efficiency.

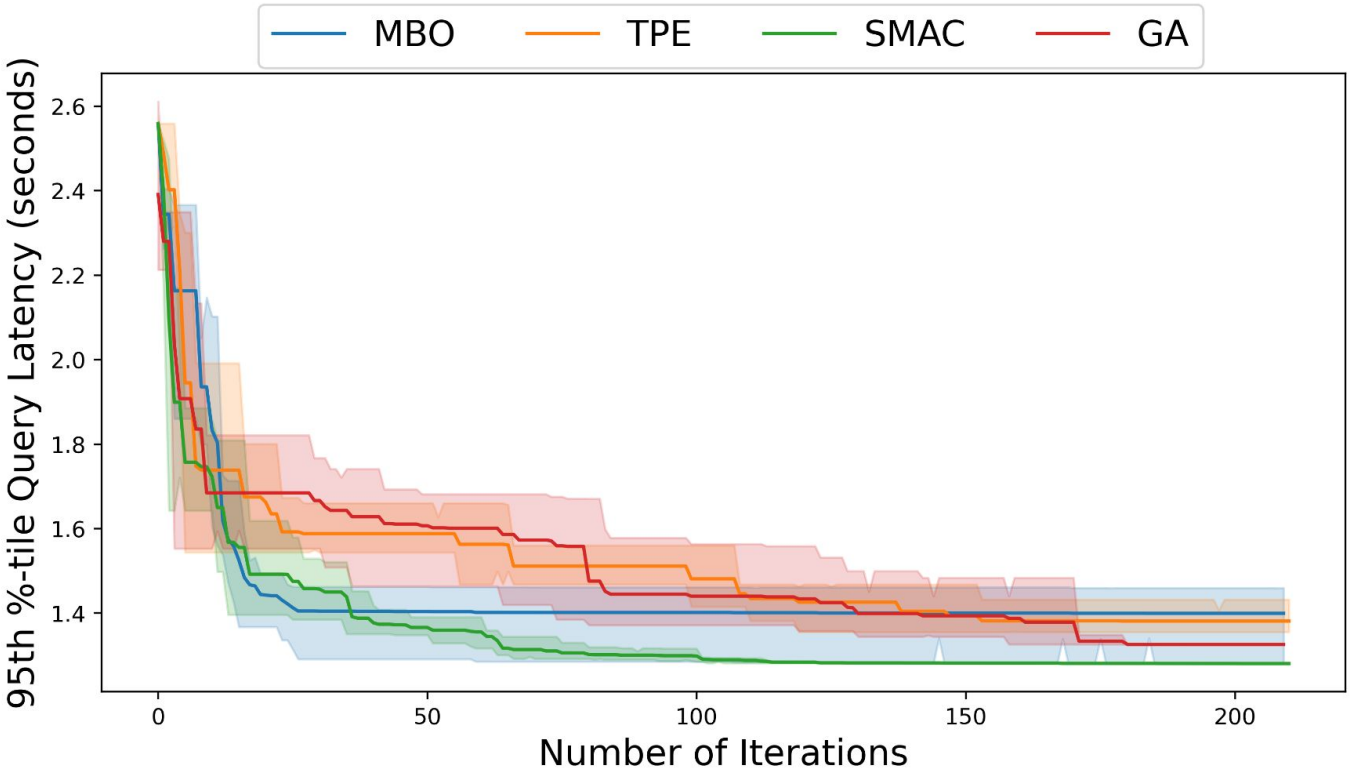
Experiments: TPS (Throughput per second)



Experiments: Average Latency

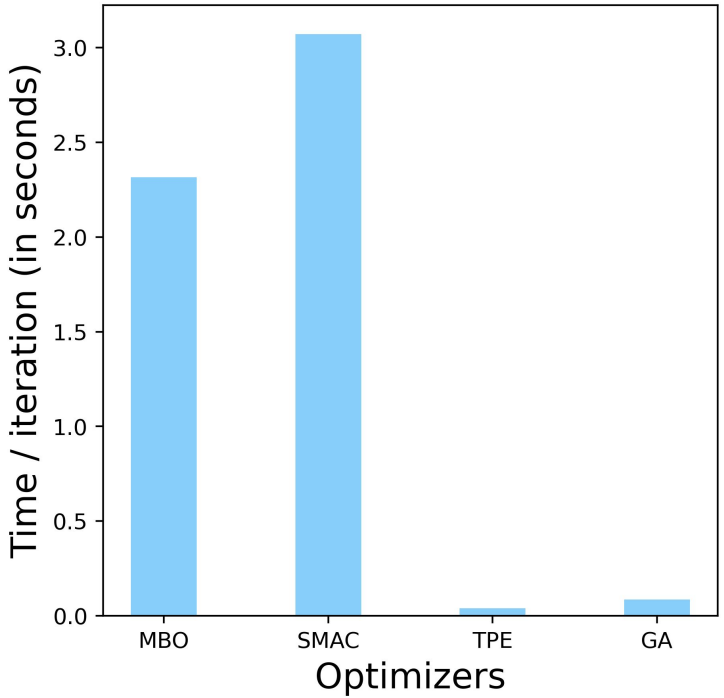


Experiments: 95th %-tile Latency



Experiments: Algorithmic Overhead

- MBO and SMAC both have higher algorithmic overhead.
  - Due to the cubic scaling behavior of GP, it becomes extremely expensive as the number of iterations increase
- GA and TPE has negligible overhead



Main Findings (Taking budget into consideration)

- SMAC and MBO are the constant best performers across different objective functions, e.g., throughput, average latency, tail latency under different budgets.
- Under ample budget, SMAC showcases the best overall performance across all objective functions.
- Under limited budget, MBO is as good as SMAC, sometimes even better.
  - MBO has better convergence speed compared to the other optimizers.
- TPE is worse than other optimizers in most cases.
  - Due to its inability to capture knob interactions

Main Findings (Taking objective functions into consideration)

- When optimizing for average latency, the optimizers show similar performance under both ample and limited budget.
- While optimizing for tps, the optimizers show contradictory performance.
  - Under ample budget, TPE is the worst performing optimizer.
  - Under limited budget, TPE performs as good as SMAC.
- While optimizing for 95th %-tile latency, the optimizers show contradictory performance.
  - Under ample budget MBO performs poorly showing similar performance as TPE.
  - Under limited budget MBO is the best performing optimizer.

Ranking Optimizers (Under Ample Budget)

Models	Throughput	Average Latency	95th-%-Latency
MBO	Good	Best	Bad
SMAC	Best	Best	Best
TPE	Bad	Worst	Worst
GA	Worst	Bad	Good

Ranking Optimizers (Under Limited Budget)

Models	Throughput	Average Latency	95th-%-Latency
MBO	Best	Best	Best
SMAC	Good	Best	Good
TPE	Good	Worst	Bad
GA	Worst	Bad	Worst

Future Work

- Assumption:
  - Nothing is known about the loss function besides its function evaluations (Black box optimization)
  - However, DBMS maintains different statistics that can provide additional information about the DBMS state.
- We maintain a simple 2-layer linear NN to predict the objective function for a certain knob configuration
  - Explore different architecture for the surrogate model
- Observe performance on real workloads in real-time.
  - A more fine-grained evaluation of the optimizers under different budget constraint.