Comparison between XGBoost, LightGBM and CatBoost using Bayesian Optimization for Parameter Tuning on Imbalanced Dataset

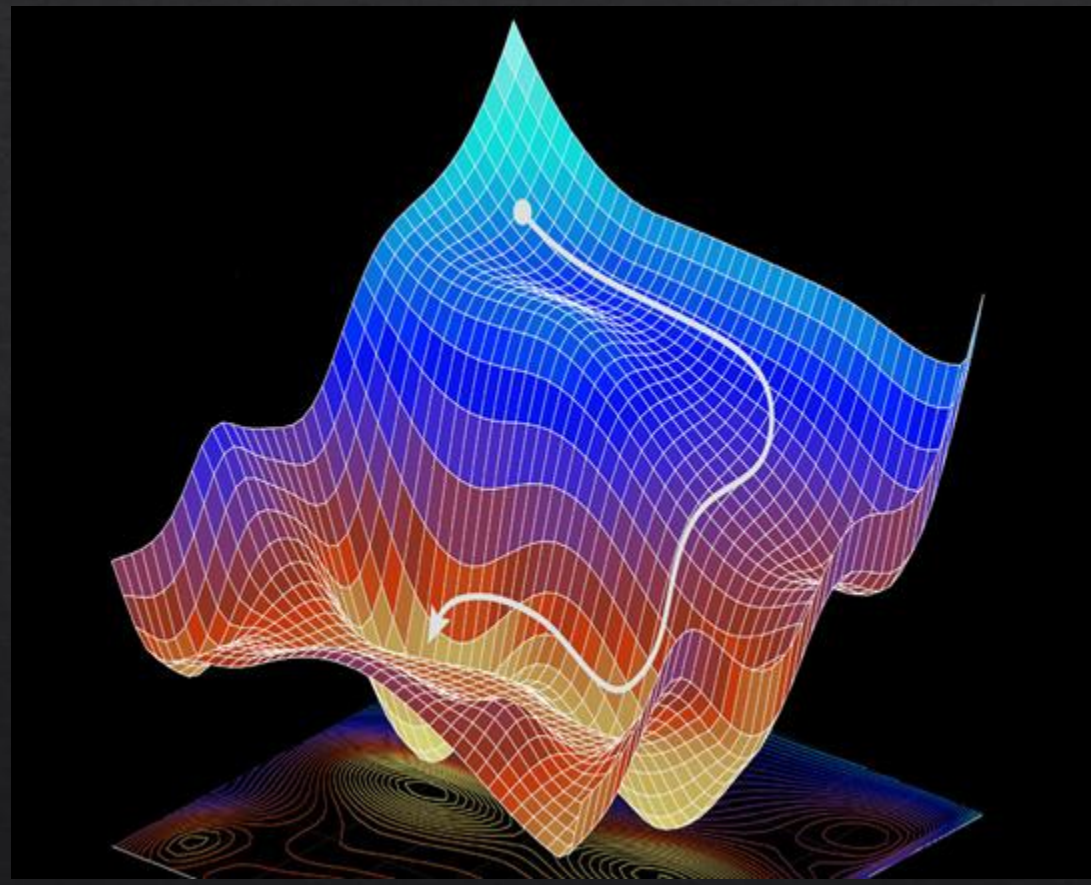
Supervised by Tanaby Zibamanzar Mofrad

Presented By: Birkamal Kaur, Kshitij Mamgain, Sasha Hajy Hassani

Contents

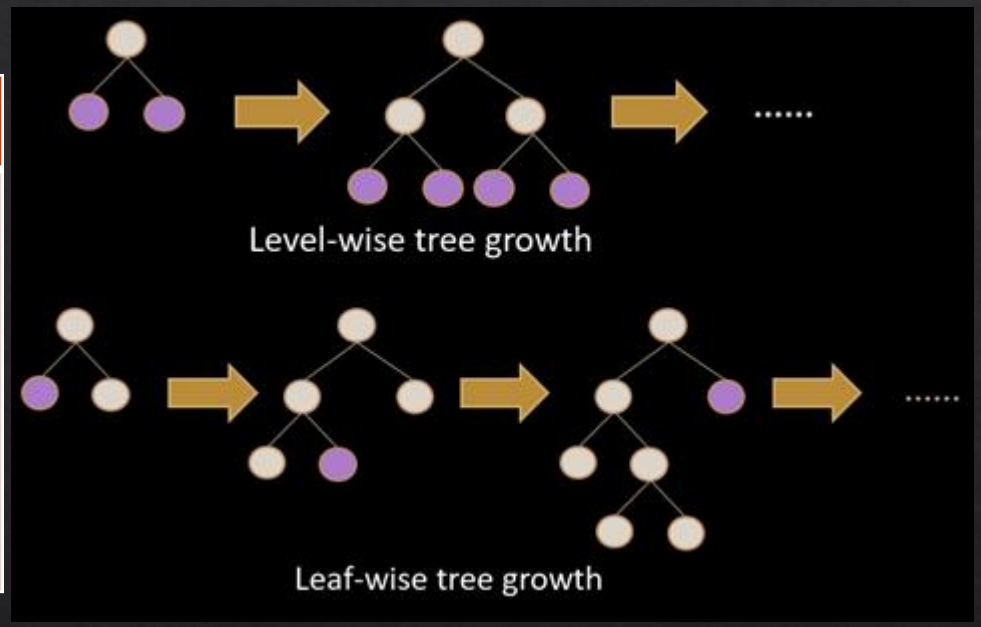
Motivation

- XGBoost and its derivative algorithms have been credited with winning numerous Kaggle competitions
- These algorithms are the driving force for several cutting-edge industry applications
- Commercial Popularity of Gradient Boosting:
  - Can deal with missing data, without fancy interpolations
  - Does not require data scaling
  - Outliers do not impact the model fitting
  - Can learn complex non-linear decisions
  - Computationally scalable for large problems (instances and features)
- Importance of Tuning
  - In real life, the ideal parameter configuration is not known
  - Helps to identify the locality with the highest likelihood of producing the strongest model



Overview of Algorithms & Optimization Techniques

XGBoost	LGBM	Catboost
• Introduced regularized objective function • Build trees by level, scans all the data points for splitting	• Histogram based algorithm. Each feature is bucketed into discrete bins • Build trees by leaves with depth first approach	• Utilizes "Ordered Boosting" to prevent residuals from shifting • Better-suited for categorical data



Random Search	Hyperopt	Optuna
• Applies random sampling in the search space • Less prone to concentrate on local minimum/ maximum	• Applies define-and-run API •Parameter search space is defined before the creation of objective function	• Applies define-by-run API • Easy to set-up and interpret • Parameter search space is defined at execution

Background

- Initially the project scope included more datasets and model comparison on the basis of observations, feature spaces, target distribution and performance on CPU and GPU
- The budget and time constrains limited the project scope
- First leg of experiment was built on Higgs data on GCP Deep Learning VM but had issues with:

i. Building and Memory space of algorithms: XGBoost and LightGBM involved discussion with the developers

ii. Allocation of GPUs for the project

- Single VM was utilized to build the experiment later 2 more VMs were spun to expedite the experiments.

Project Aim

- The aim of experiment is to compare and evaluate the optimized performance of XGBoost, LightGBM and CatBoost algorithms using Hyperopt, Optuna and Random Search for parameter tuning on a highly imbalanced dataset
- Credit Card Fraud dataset is obtained from Kaggle. The experiment analyzes the model training time and performance, important machine learning algorithm metrics and the feature importance for tree based models
- The approach taken in this experiment is to run the experiments multiple time to address the following questions:

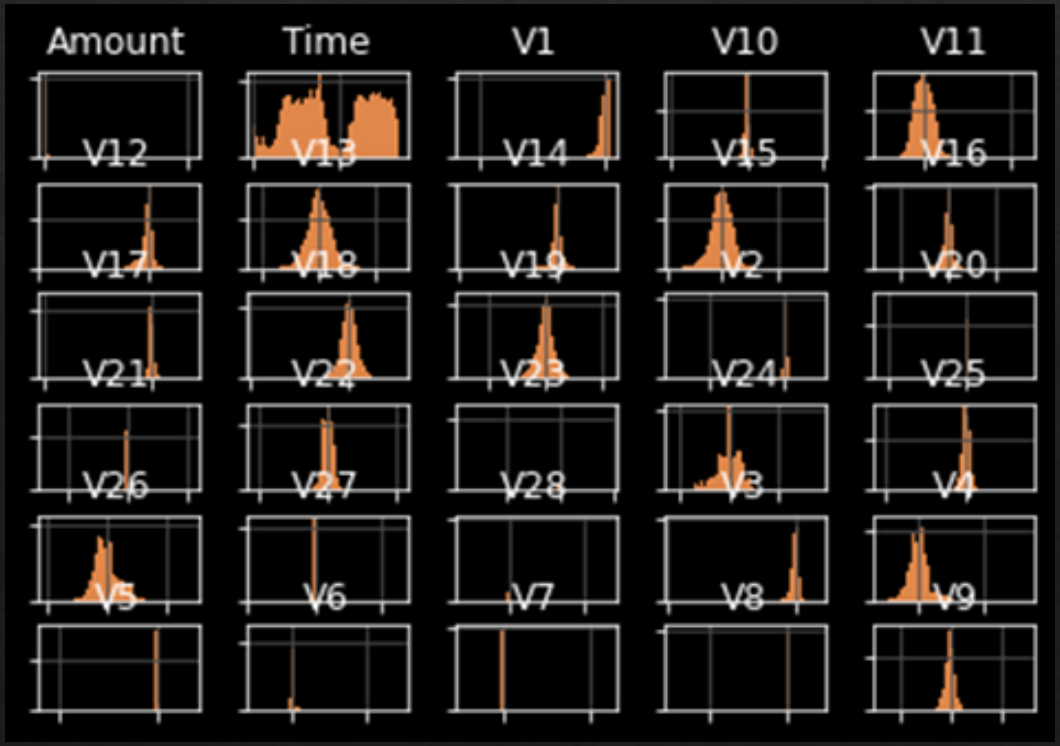
i. How do the time to hyperparameter optimization and best loss statistically compare for each algorithm and optimization?

ii. How do the performance of XGBoost, LightGBM and CatBoost compare on different evaluation metrics (F1 Score, ROC AUC curve, PR AUC and Kappa coefficient)?

iii. How robust is each model to generate feature importance?

About Dataset

- The dataset contains credit cards transactions and there are total 30 features and one binary target
- Most of the features have been transformed via a principal component analysis (PCA) transformation.
- The dataset is highly imbalanced, the positive class (frauds) account for 0.167 % of all transactions



Feature	Description
Time	Number of seconds elapsed between transactions
Amount	Transaction Amount
Class	0-Normal 1-Fraud

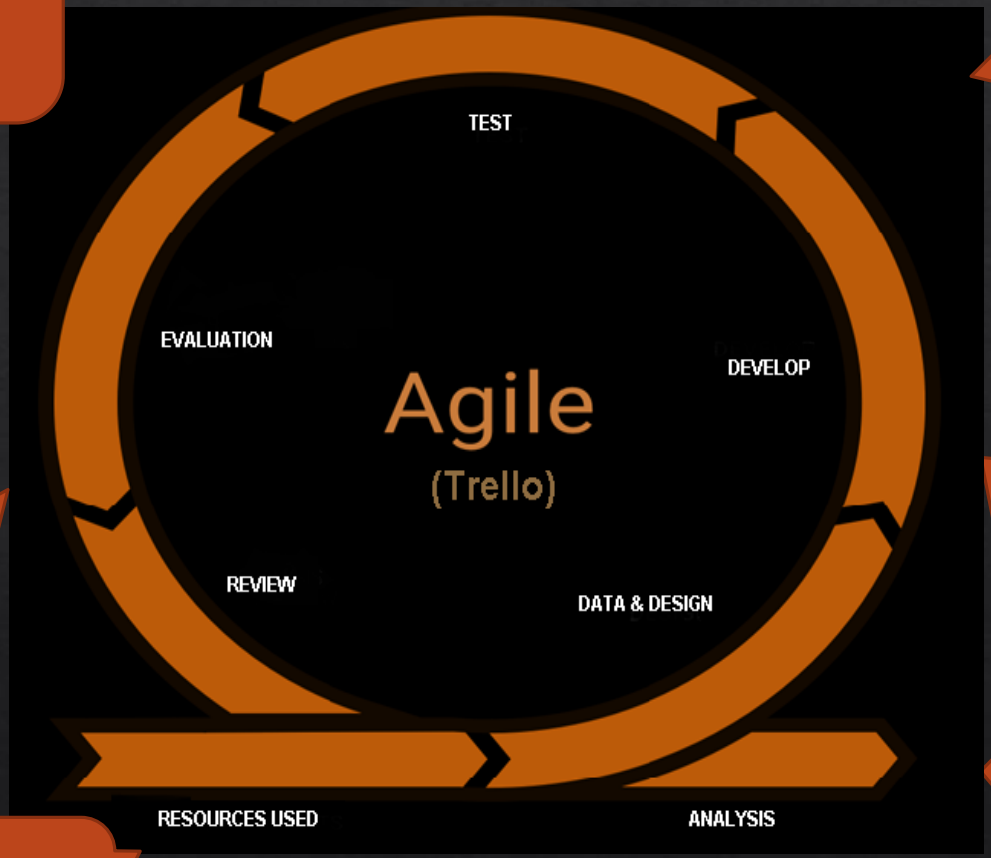
Class	Count
0	283,253 instances
1	473 instances

Project Pipeline

•30% of data was used as Test Set for all the 3 algorithms
• F1-score was used as evaluation metric

•Used imbalanced data on Credit Card Fraud from Kaggle
• Hyperopt, Optuna & Random Search were used for hyper-parameter optimization
• Gradient Boosting methods were used for training
• In total, 105 experiments were run

•Google Cloud Repository (GCP)
• Three 8 v CPU Deep Learning VM
•Cloud Source, Google Colab



Unit & Integration testing to check the smooth running of experiments

3 models were developed to record:
• Optimization time
• Minimum loss
• Best parameters
• Confusion matrix
• Classification report

Statistical tests were performed to check the performance of the algorithms

Parameter Tables

Common Parameters

- Optimizing - learning rate: 0.3, objective: logloss, eval: F1, nthread: 8, early stopping: 100
- Testing - learning rate: 0.05
- The hyper parameter ranges for all algorithms were fixed considering the guideline from its developers (link) or based on general practice on Kaggle and other popular forums.

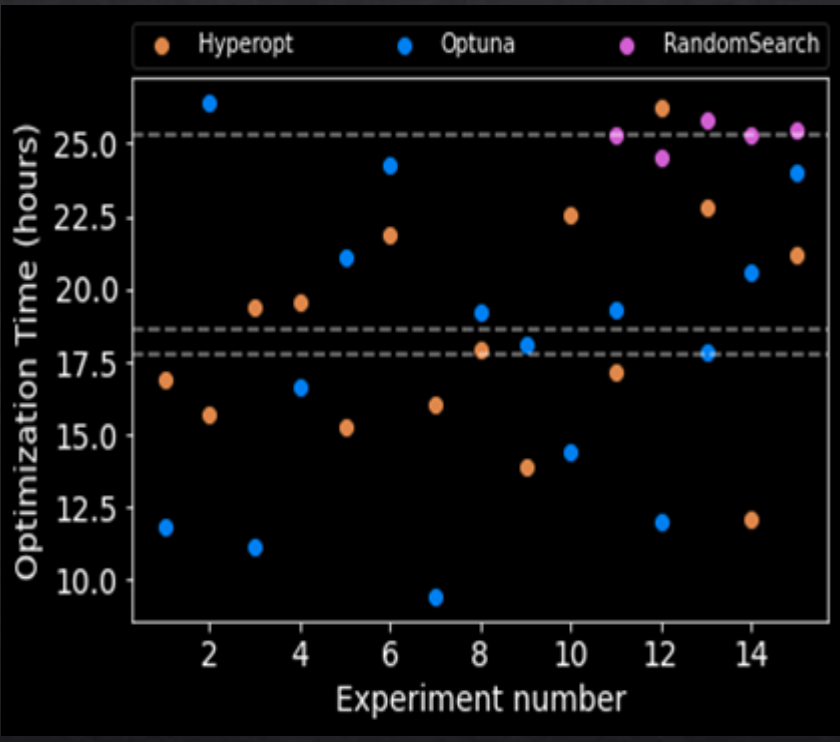
XGBoost Hyperparameter Range	
gamma	0.001-100
max_depth	3-7
subsample	0.7-1
colsample_bytree	0.7-1
colsample_bylevel	0.7-1
reg_lambda	0.001-100
reg_alpha	0.001-100
tree_method	approx, hist
grow_policy	depthwise, lossguide

LightGBM Hyperparameter Range	
boosting_type	gbdt, goss
num_leaves	31-1023
lambda_l1	0.0-0.5
lambda_l2	0.0-0.5
min_gain_to_split	0.0-5.0
min_data_in_leaf	20-500
min_child_weight	0.1-10
feature_fraction	0.4-1.0

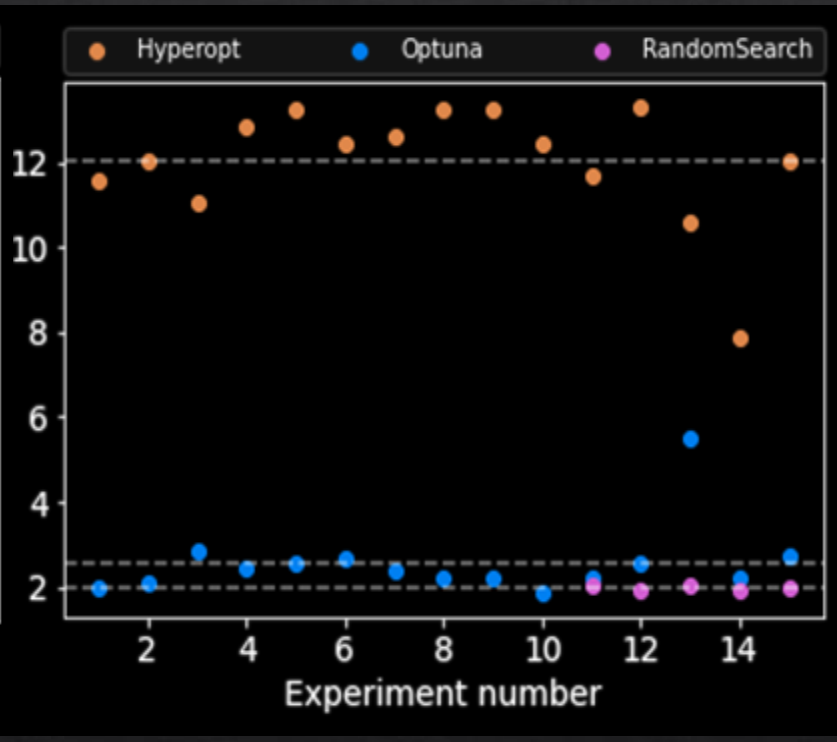
CatBoost Hyperparameter Range	
l2_leaf_reg	log1-10
border_count	32-128
bootstrap_type	Bayesian,Bernoulli
grow_policy	SymmetricTree, Depthwise, Lossguide
depth	1-16
min_data_in_leaf	1-50
od_type	IncToDec, Iter
leaf_estimation_backtracking	No, AnyImprovement
rsm	0.1-1
random_strength	1-20

Result Findings: Optimization Time

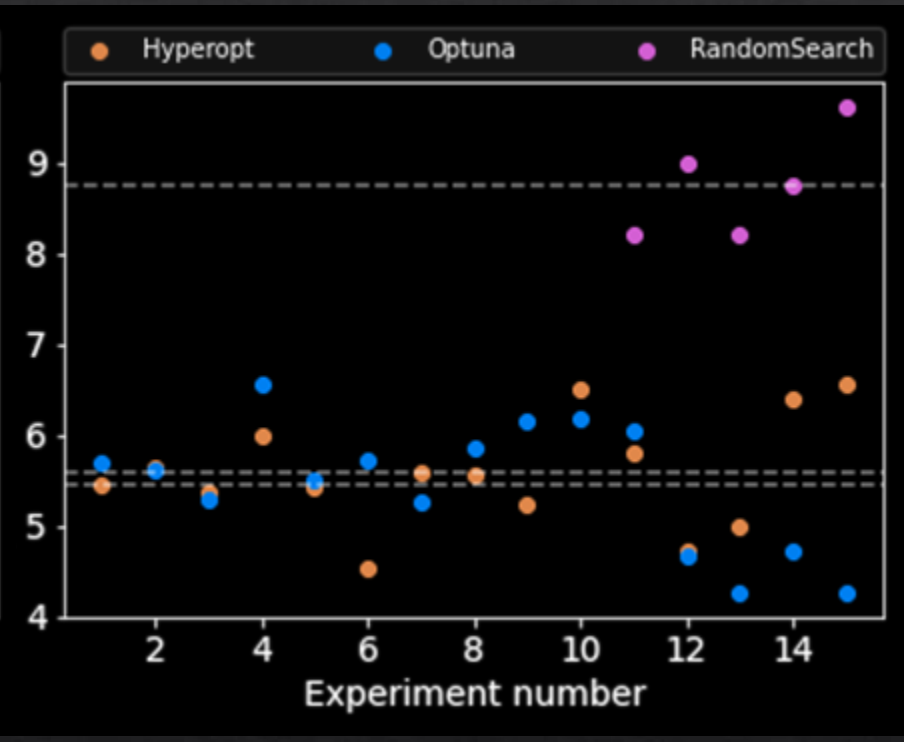
XGBoost



LightGBM



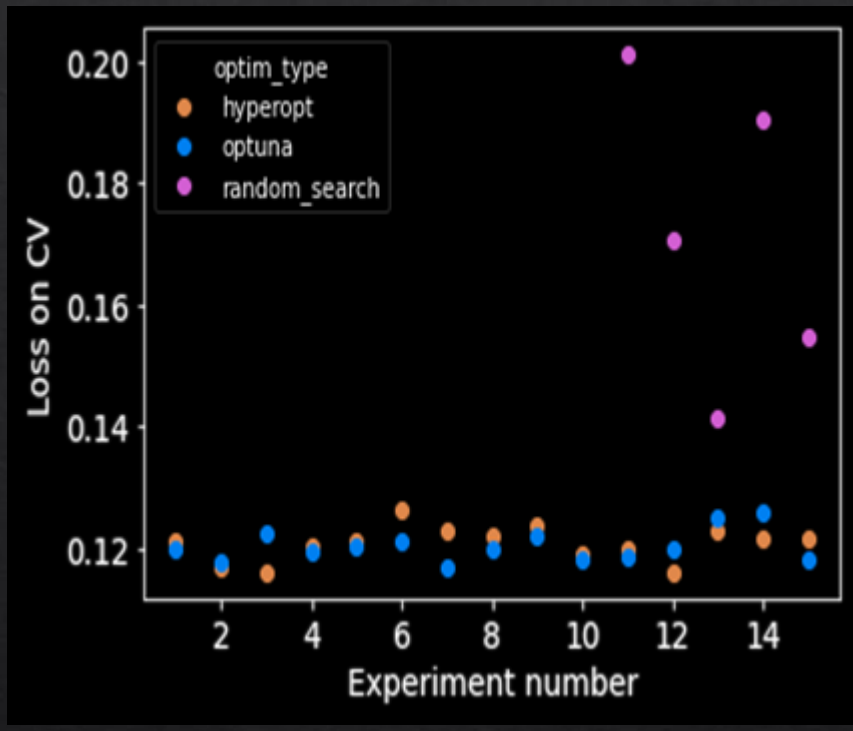
CatBoost



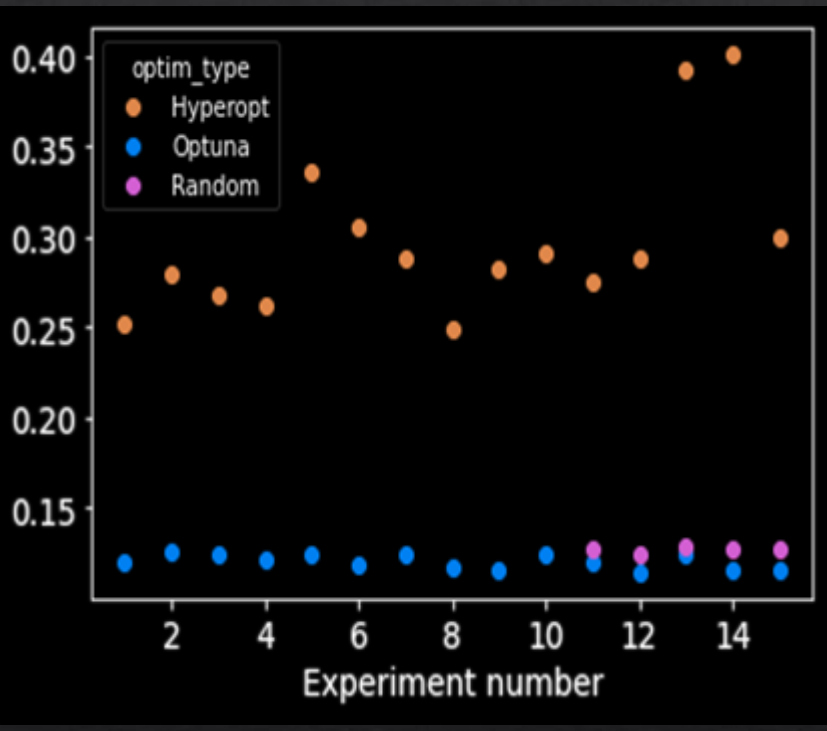
	XGBoost	LightGBM	CatBoost
Hyperopt	18.56	12	5.58
Optuna	17.73	2.56	5.45
Random Search	25.27	1.98	8.75

Result Findings: Minimum Loss

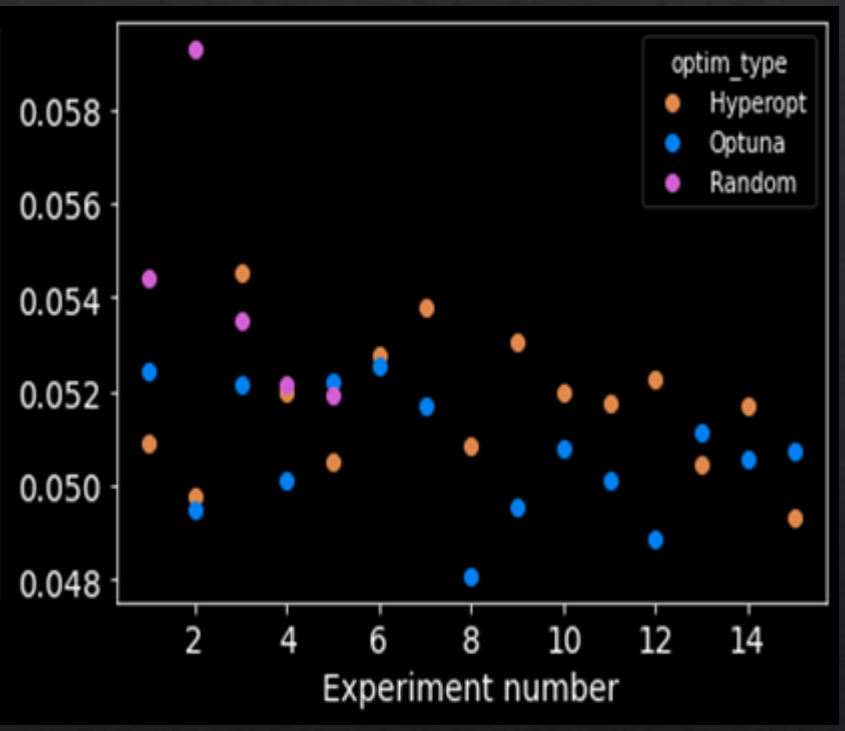
XGBoost



LightGBM



CatBoost

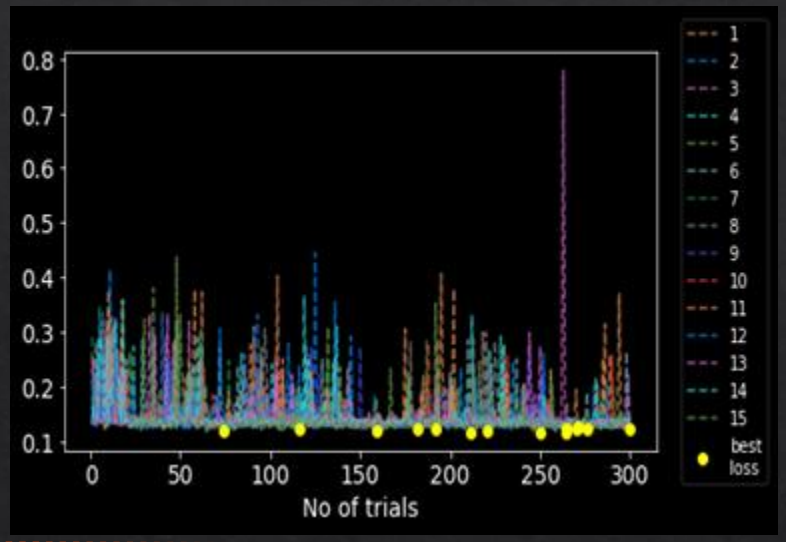


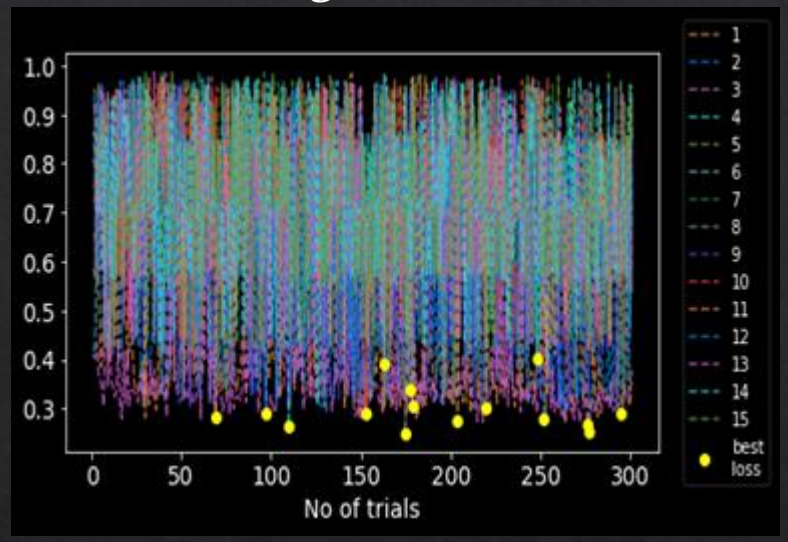
	XGBoost	LightGBM	CatBoost
Hyperopt	0.12	0.30	0.05
Optuna	0.12	0.12	0.05
Random Search	0.17	0.13	0.05

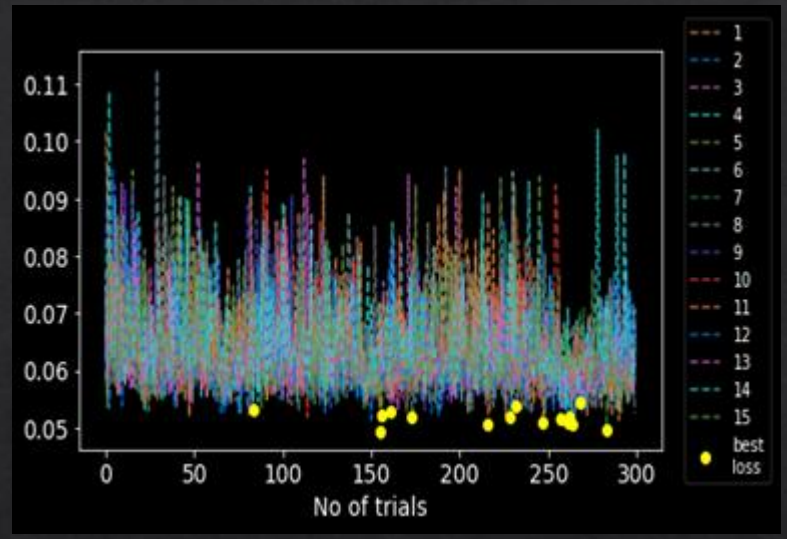
Result Findings: Convergence Comparison

XGBoost | **LightGBM** | **CatBoost**

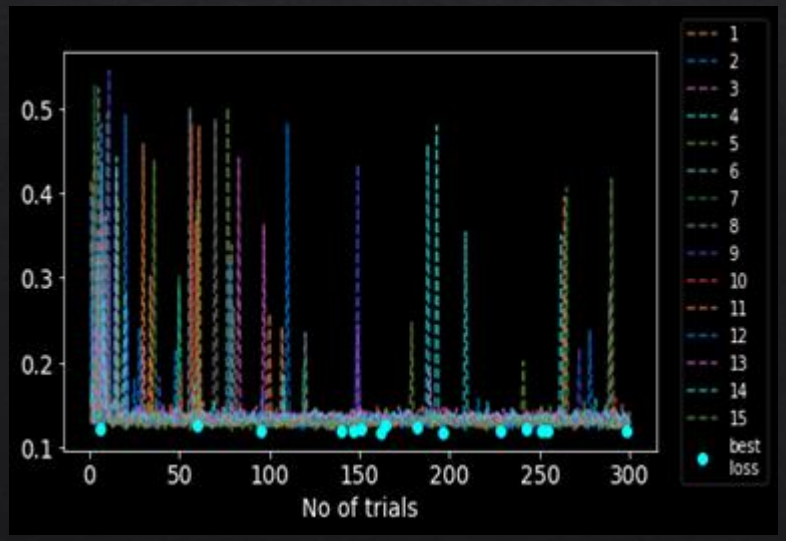
HYPEROPT

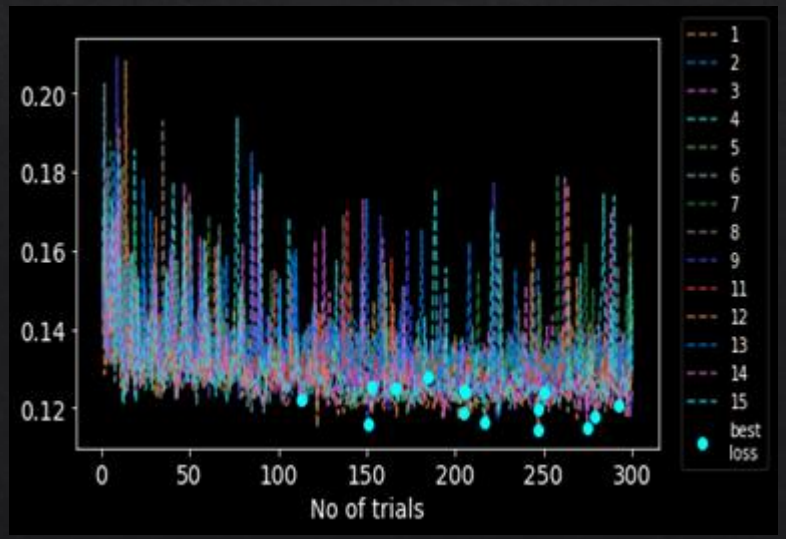


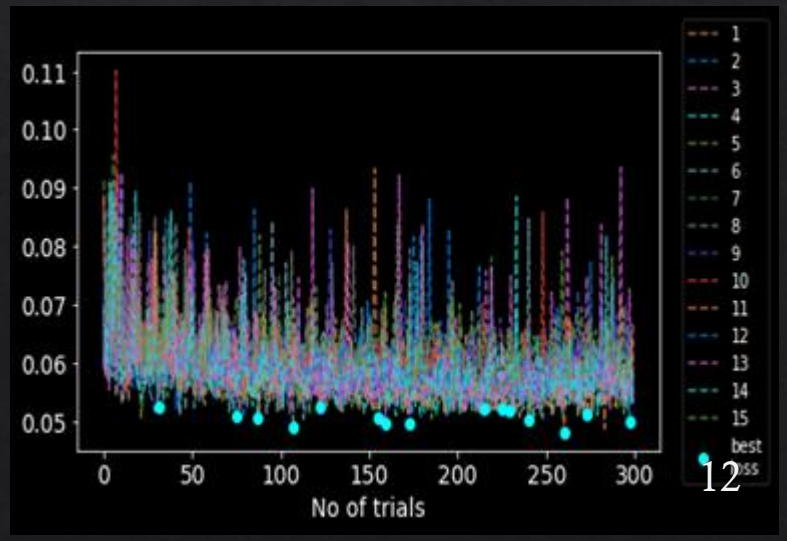




OPTUNA

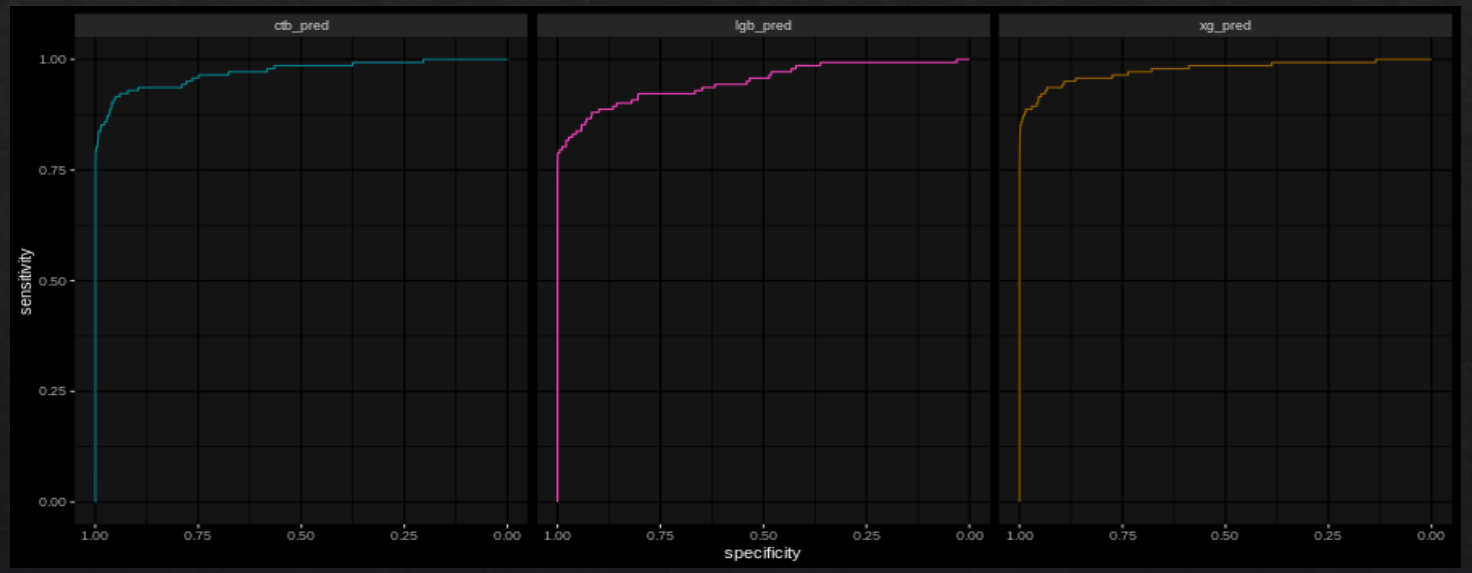






Statistical Comparison

- To compare algorithm performance statistical tests were run. The models were compared on: ROC AUC, PR AUC, F1 Score and Kappa Coefficients.
- Wilcoxon Rank Sum: A popular non-parametric version of the two-sample t-Test to compare distribution of results
- Delong's ROC: tool to compare AUC of ROC of 2 curves and it relies on the empirical values of AUCs
- Kappa Coefficient: Good measure to handle imbalanced class problems



Result Findings: Statistical Tests

- Wilcoxon Rank Sum: XGB is consistent and better than LGB and CTB in all performance metrics in Hyperopt
- LGB performs equally well using Optuna yet 6 times faster
- Min, Median, Max summary for ROC AUC and Kappa coefficient

Wilcox Rank Sum (P-values multiplied by e-6)

Metrics	XGB>LGB	XGB>CTB	LGB>CTB
Hyperopt			
ROC AUC	1.063*	458.2	1000000 *
PR AUC	72.36	2.44*	1662.0
F1-score	34.55	1.529*	1.662*
F1-Score	130.9	1.692*	8326.0
Optuna			
ROC AUC	992700.0	2472.0	10.94
PR AUC	999800.0	6.991*	1.372*
F1-score0.5	1000000*	0.006447	1.629*
F1-score best	1000000*	2.52*	1.683*

Delong Test For ROC AUC (P-values)

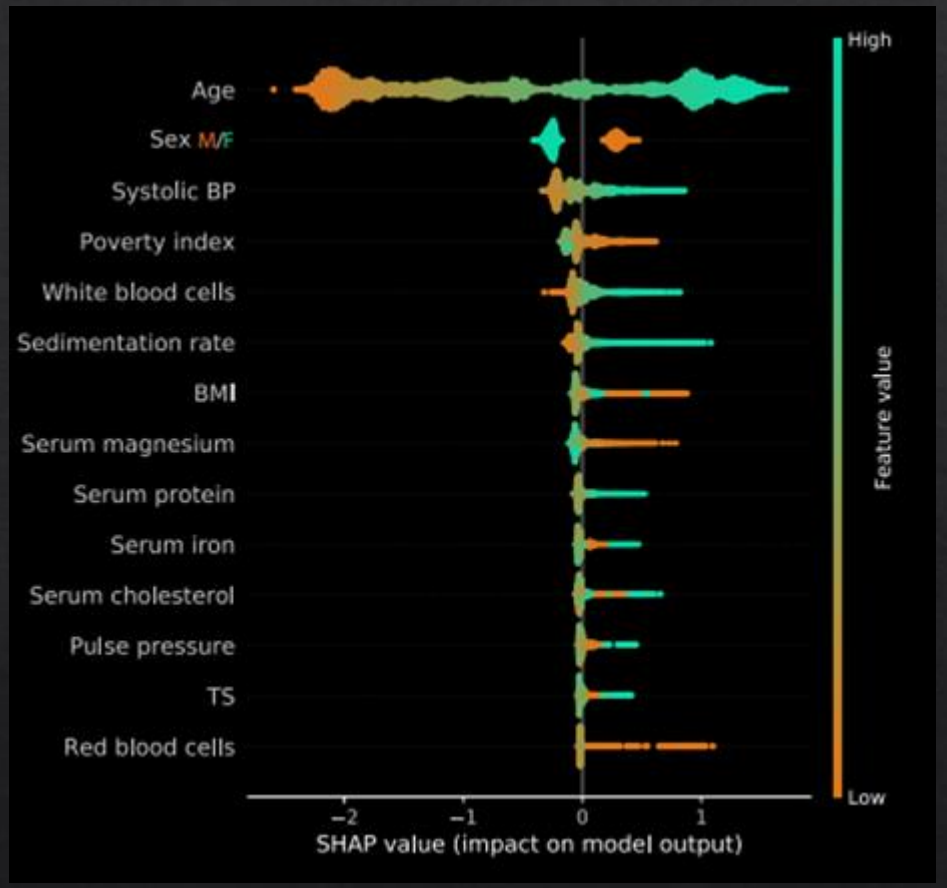
Tuning Type	XGB>LGB	XGB>CTB	LGB>CTB
Hyperopt	0.2470	0.0008**	0.9774*
	0.0824	0.0310	0.9262
	0.0140*	0.0273	0.4720
Optuna	0.2600	0.4056	0.2051
	0.1023	0.4055	0.0327
	0.0467	0.1018	0.0045
Random Search	0.3998	0.1110	0.1847
	0.0069*	0.0597	0.1220
	0.0015**	0.0597	0.0110*

Kappa Coefficients

Tuning Type	XGB	LGB	CTB
Hyperopt	0.85	0.83	0.73
	0.84	0.79	0.67
	0.84	0.63	0.64
Optuna	0.84	0.84	0.79
	0.81	0.84	0.77
	0.79	0.83	0.75
Random Search	0.83	0.85	0.77
	-	-	-
	0.80	0.82	0.55

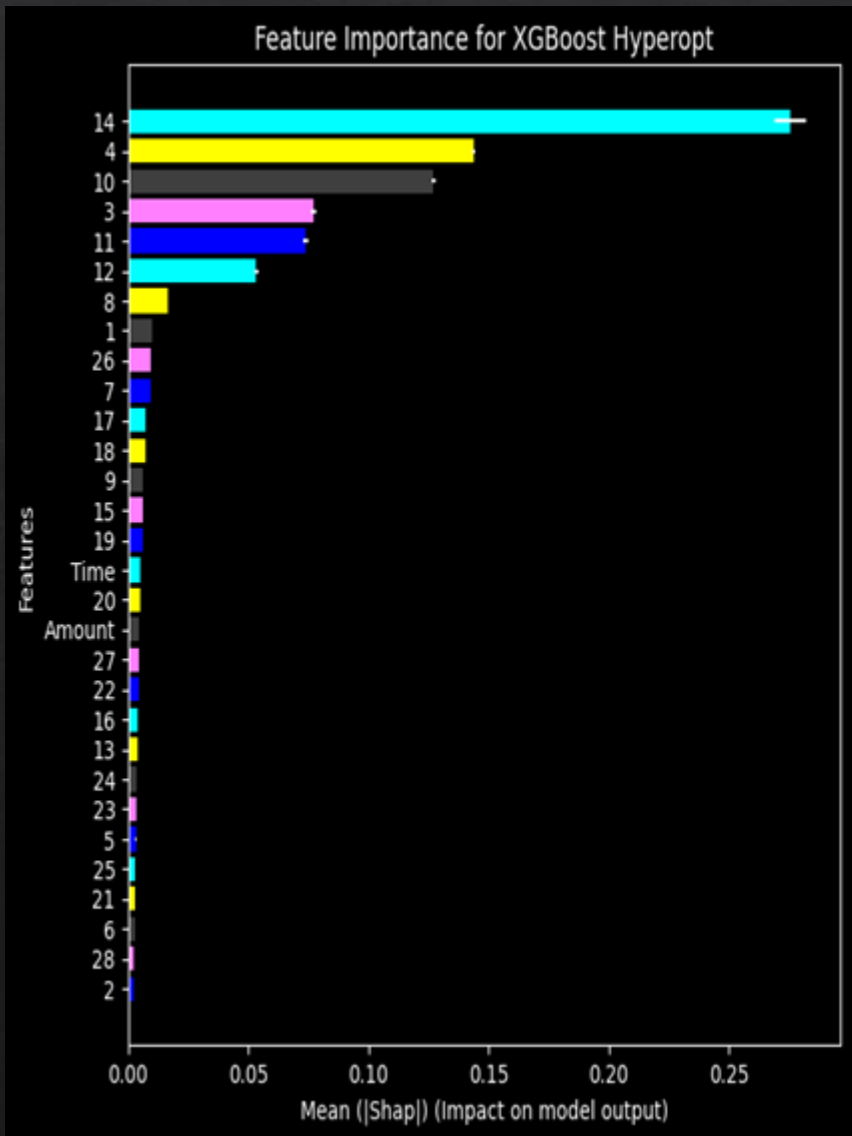
Result Findings: Based on SHAP Values

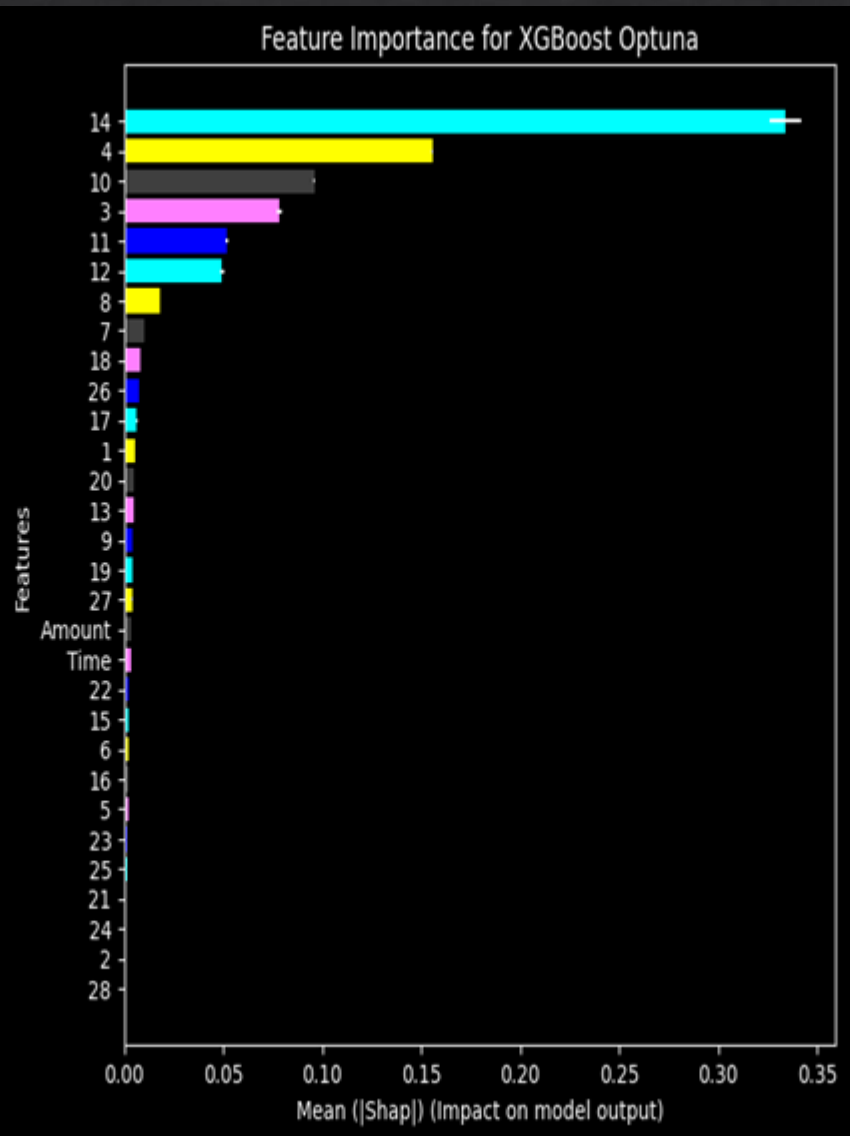
- SHAP (SHapley Additive exPlanation) values are the unique consistent and locally accurate attribution values for a model based on game theory.
- Impact on model output: Every observation in the dataset is run through the model and a dot is created for each feature attribution value. Dots are colored by the feature's value for that observation and pile up vertically to show density.
- The mean absolute value of the SHAP values for each feature to get a standard bar plot

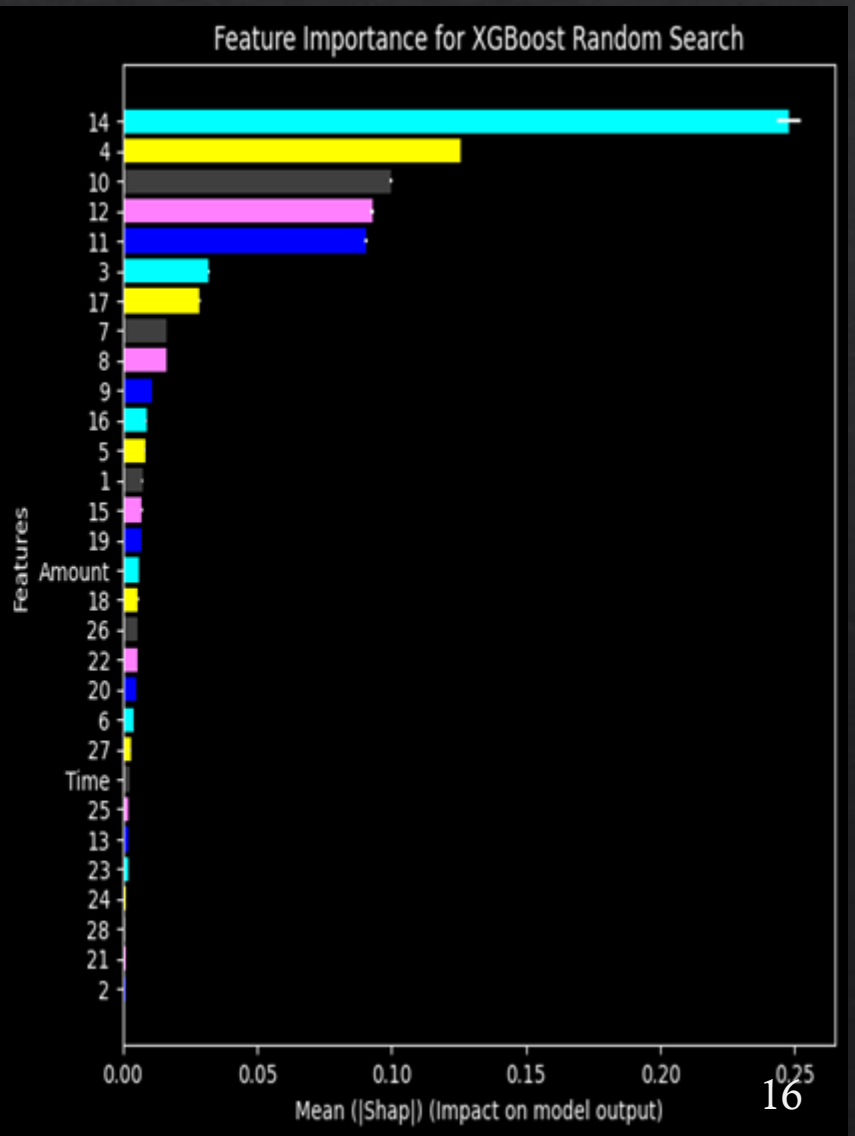


* Consistent Individualized Feature Attribution for Tree Ensembles

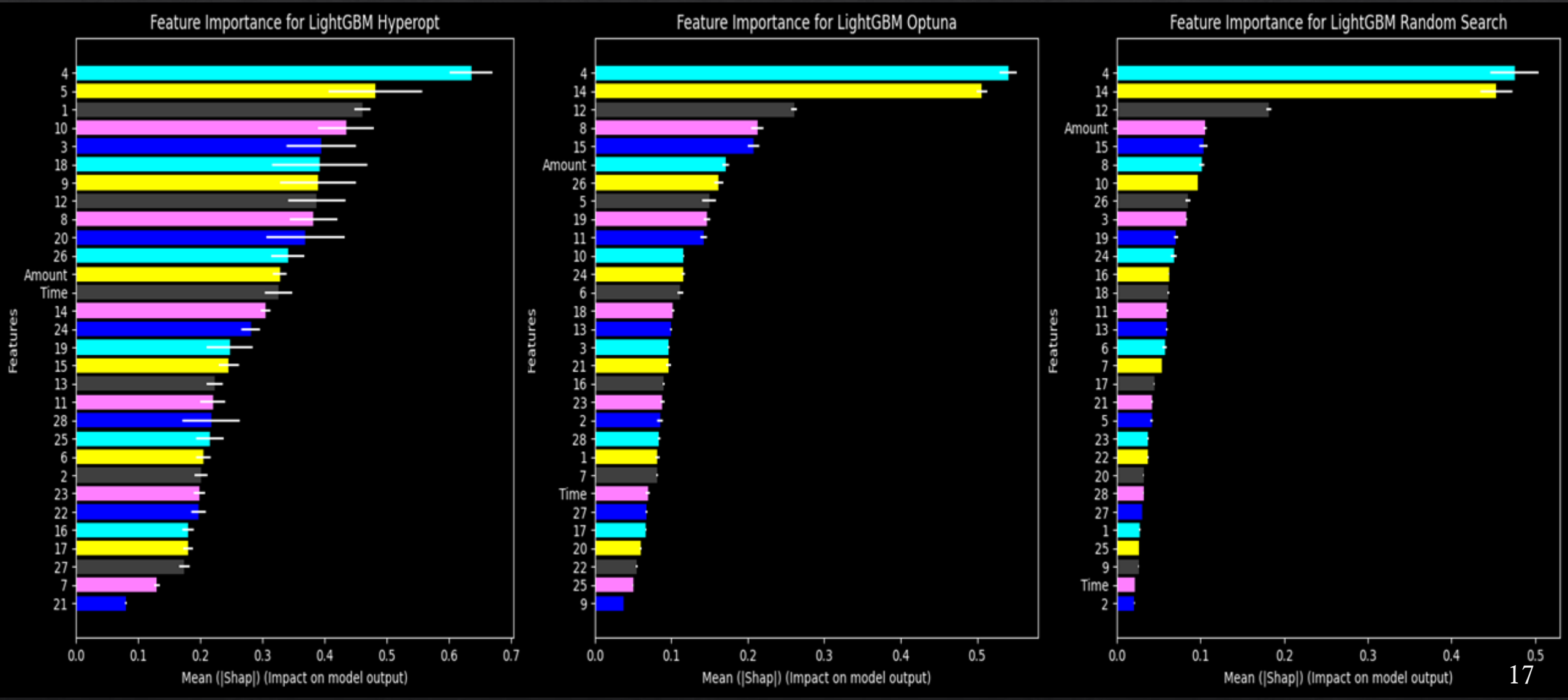
Result Findings: SHAP Values XGB



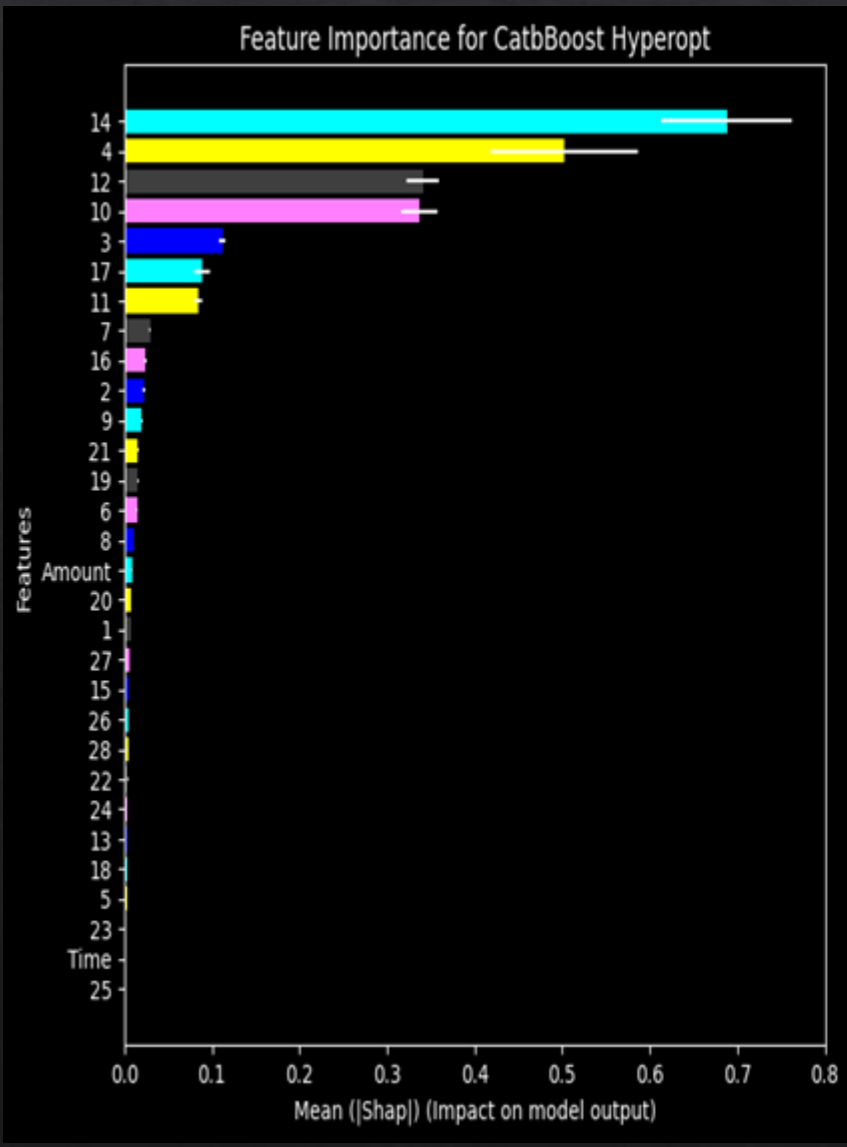


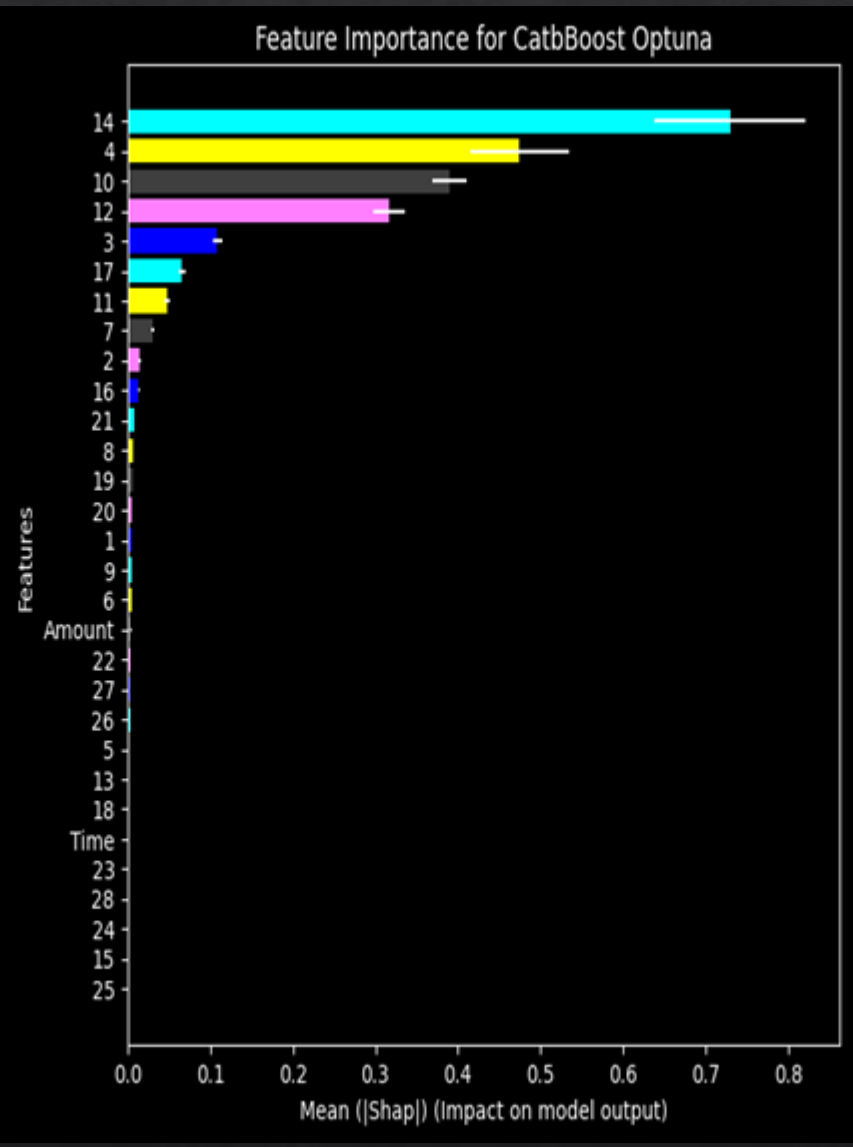


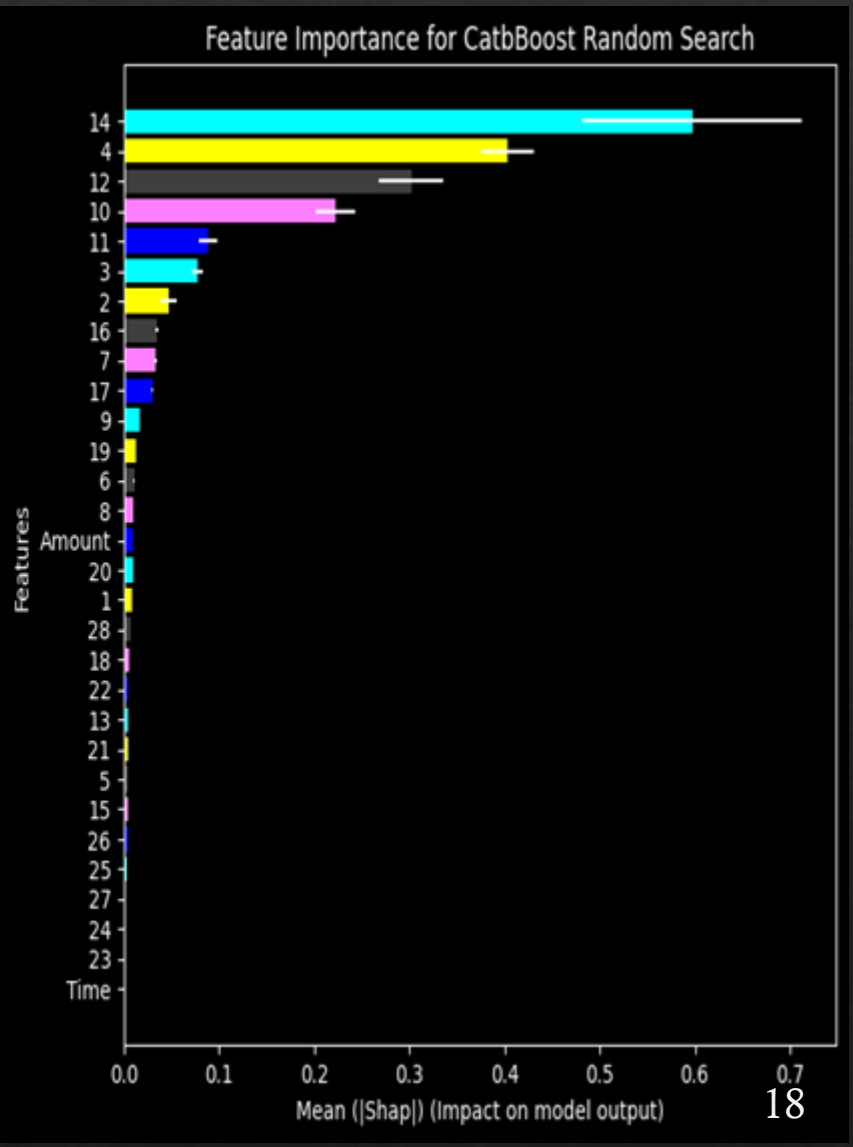
Result Findings: SHAP Values LGB



Result Findings: SHAP Values CTB







Experiment Conclusion

- Optimization time and loss
  - In both optimization time and loss Optuna is cost effective than Hyperopt. Optuna pruning also helps in finding the best loss in lesser trials and corroborates the findings of its developers.
  - Algorithm wise XGBoost had more consistent results all 3 optimization techniques but is highly cost intensive on the other hand LightGBM integrated with Optuna has similar performance yet 6 times faster.
- Performance on Metrics: AUC, PR AUC, F1 Score and Kappa
  - CatBoost looses in overall comparison since the dataset does not have categorical features
  - XGBoost has similar performance in both Hyperopt and Optuna
  - The cross comparison of algorithms for optimizations also proves that Kappa Score is better using Optuna for both LightGBM and CatBoost
- Feature Importance
  - Based on feature importance using mean SHAP absolute values and the variations shows thatXGBoost and CatBoost work identically for feature selection.
  - LightGBM utilized all the features with lesser features around zero. This could be due to histogram based approach. Feature engineering becomes easier with XGBoost and CatBoost

Future Work

- Extend on the project to include more diverse datasets
- Include the model performance on GPUs and impact of infrastructure on the model performance
- Explore the reasons for high losses for LightGBM with using Hyperopt technique.

Thank You