Flow blockchain execution efforts prediction model

Introduction

The model is developed to predict the execution effort of each transaction based on the tasks performed in the Flow blockchain

Since the actual execution effort is dependent on the node hardware configuration and runtime status such as memory and CPU usage, we are only able to predict the minimum execution effort based on the data provided from specific machines. We are going to use the ms column as the proxy for execution effort, as we assume a constant relationship between effort and time.

Flow team also requests:

1. model in the linear form
2. model's coefficients should all be positive
3. linear model is better to not have the intercept constant team

Model Highlights

1. the proposed champion model achieve the better performance when compared to the current production benchmark model model in terms of lower fees and goodness of fit
2. only 4 features in the model
3. all coefficients are positive
4. calibration is not needed since the model's prediction on testnet data is not biased based on the goodness of fit plot

Data

Raw data

1. machine 1 simulated data
2. machine 2 simulated data
3. testnet data

https://github.com/onflow/flow-go/tree/janez/computation-metering-prototype-parameter-calibration/fvm/parameter_calibration/analysis

Features

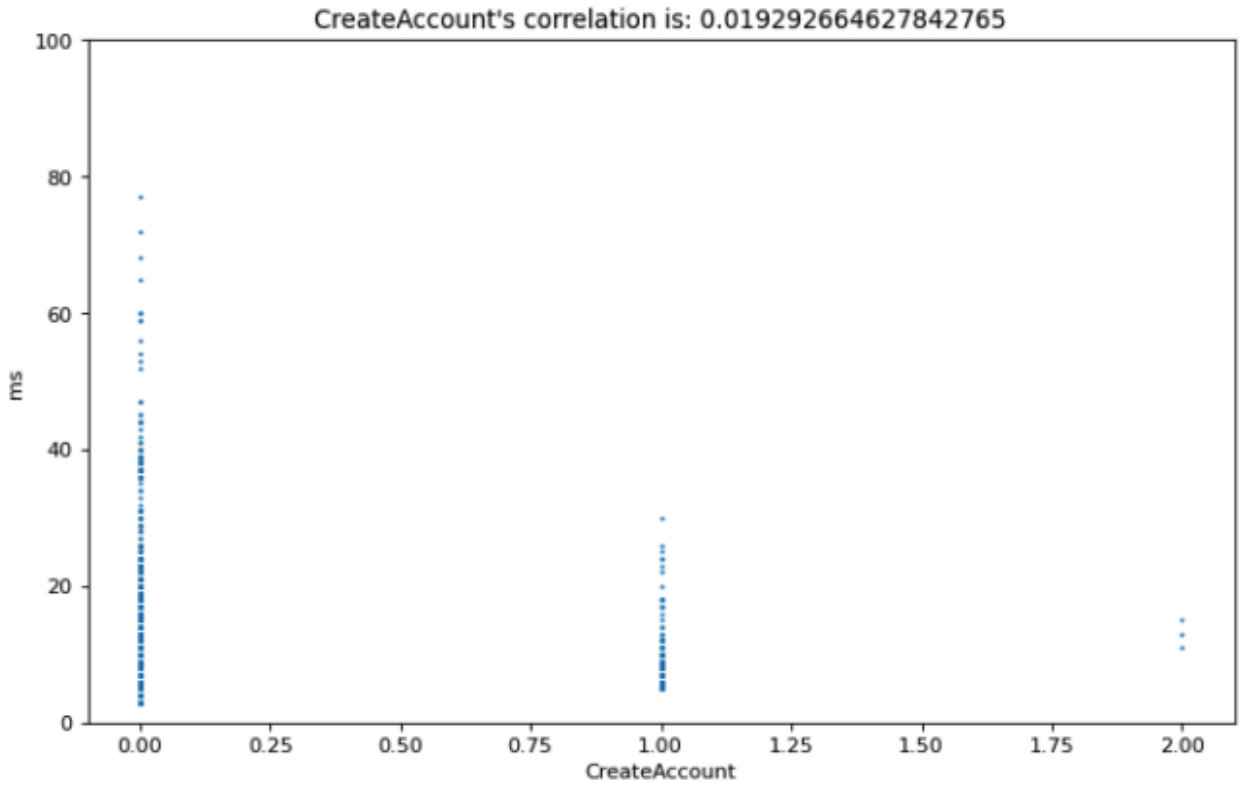
1. Dependent variable: ms
2. Top independent features
    a. function_or_loop_call
    b. SetValue
    c. GetValue
    d. CreateAccount
3. Independent features cannot be used due to mainnet data limitations of these features, data quality, or data type fit with linear model
    a. tx

b. ResolveLocation
c. SetProgram
d. ProgramParsed
e. ValueDecoded
f. ProgramChecked

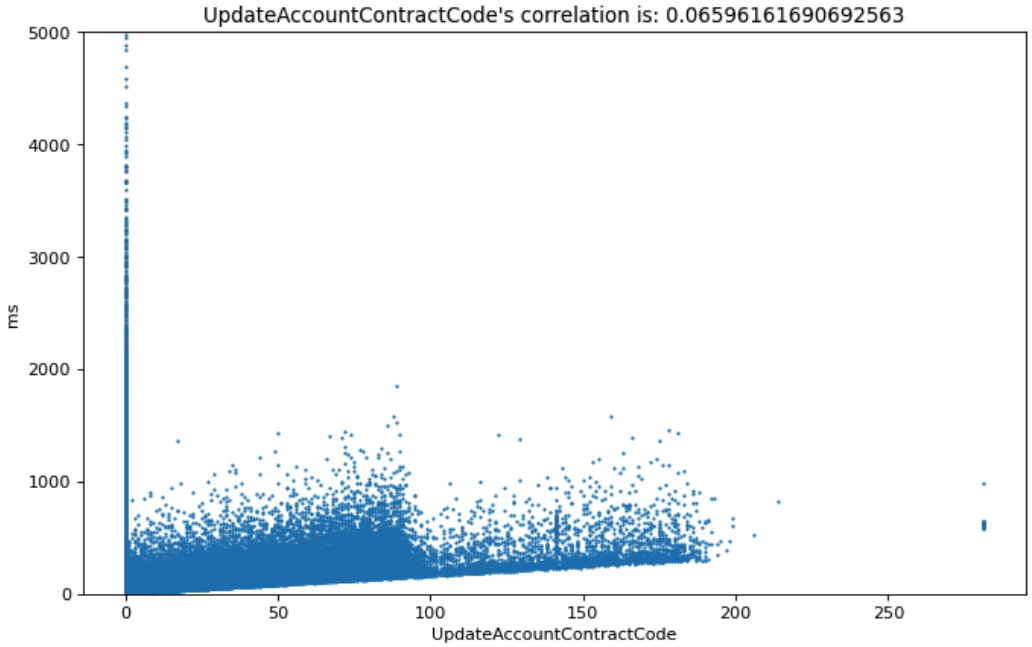
4. Independent features removed during EDA process
    a. GenerateUUID
    b. GetAccountContractCode
    c. UpdateAccountContractCode
    d. ContractFunctionInvoke
    e. GetProgram
    f. GetCode

Exploratory Data Analysis

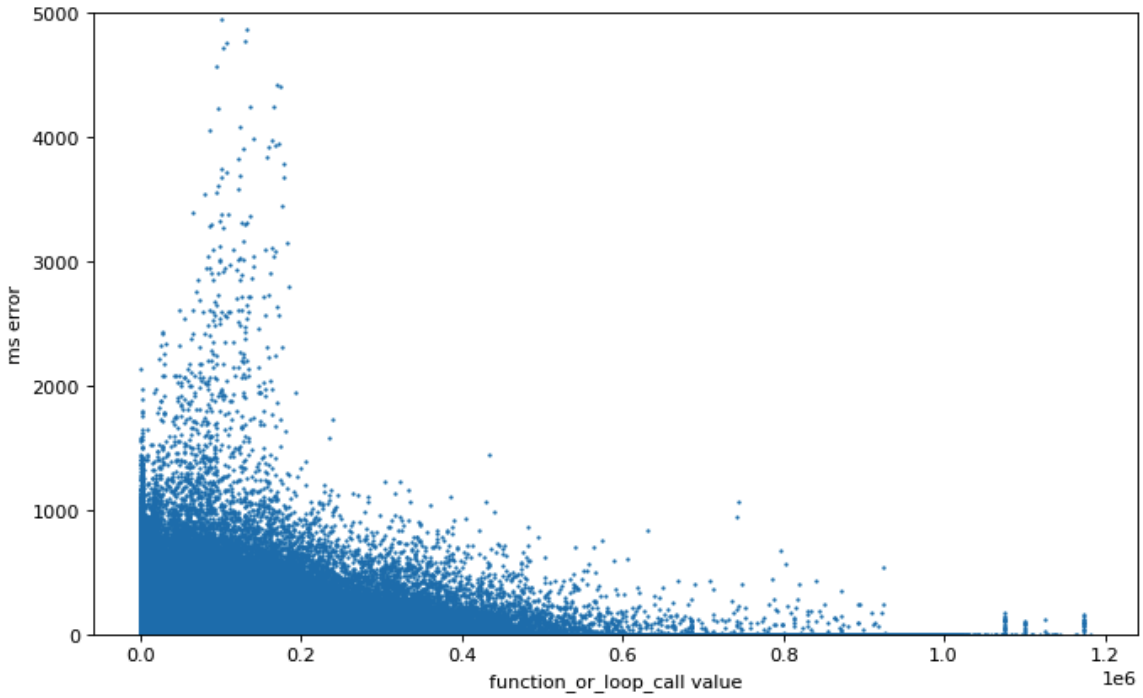
1. Highlights
    a. testnet data is not sufficient for modeling because variables exhibit discrete pattern due to small data size (less than 10k samples)
        i. e.g. CreateAccount [source: testnet]



b. many of the Flow function features were sparse with values clustered at 0 and as a result the features often had low variance
    i. e.g. UpdateAccountContractCode [source: machine 2]



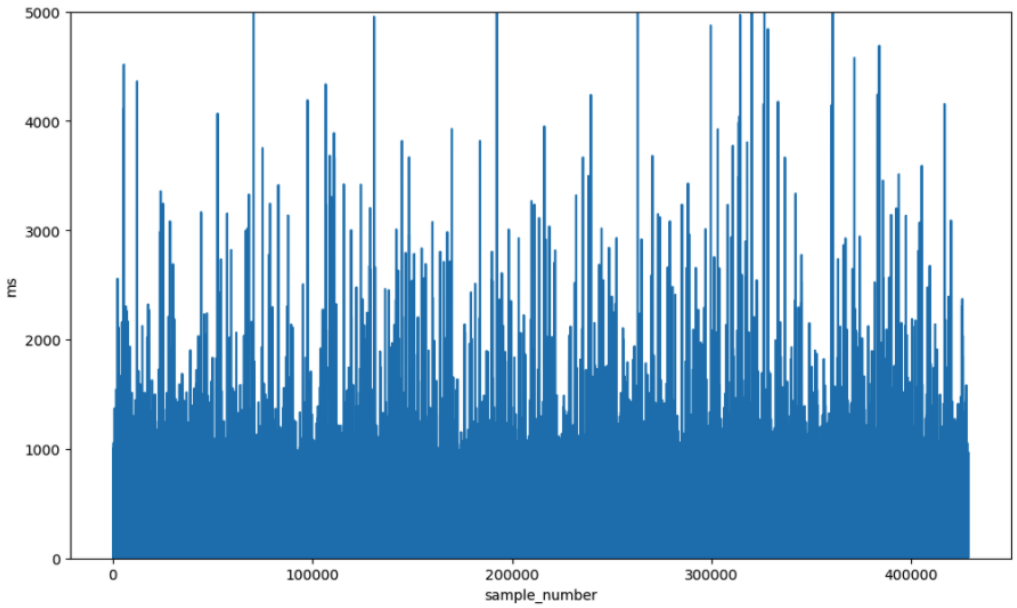
c. ms is floored for each feature's value
    i. The machine noise has an exponential shape and can be measured by the difference between the ms and minimum (floor) ms for each feature value



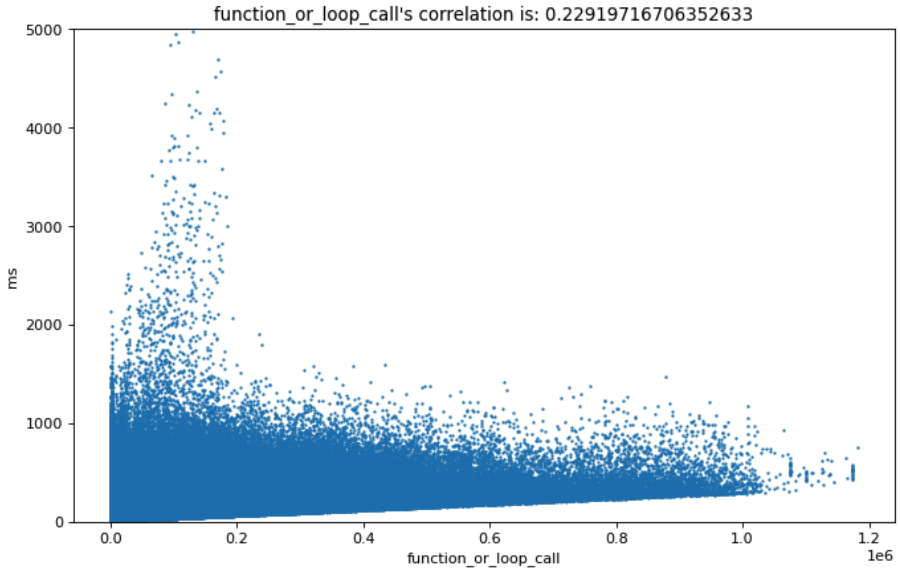
d. through the EDA process many features were removed from the training sample due to significant null values, low variance, and multicollinearity

2. Dependent Variable Univariate and Bivariate Relationships

a. The dependent variable, ms, ranged from 0 to approximately 5,000. The longer transaction times represents the error term due to noise from the garbage collector for example or other machine related factors [source: machine 2]

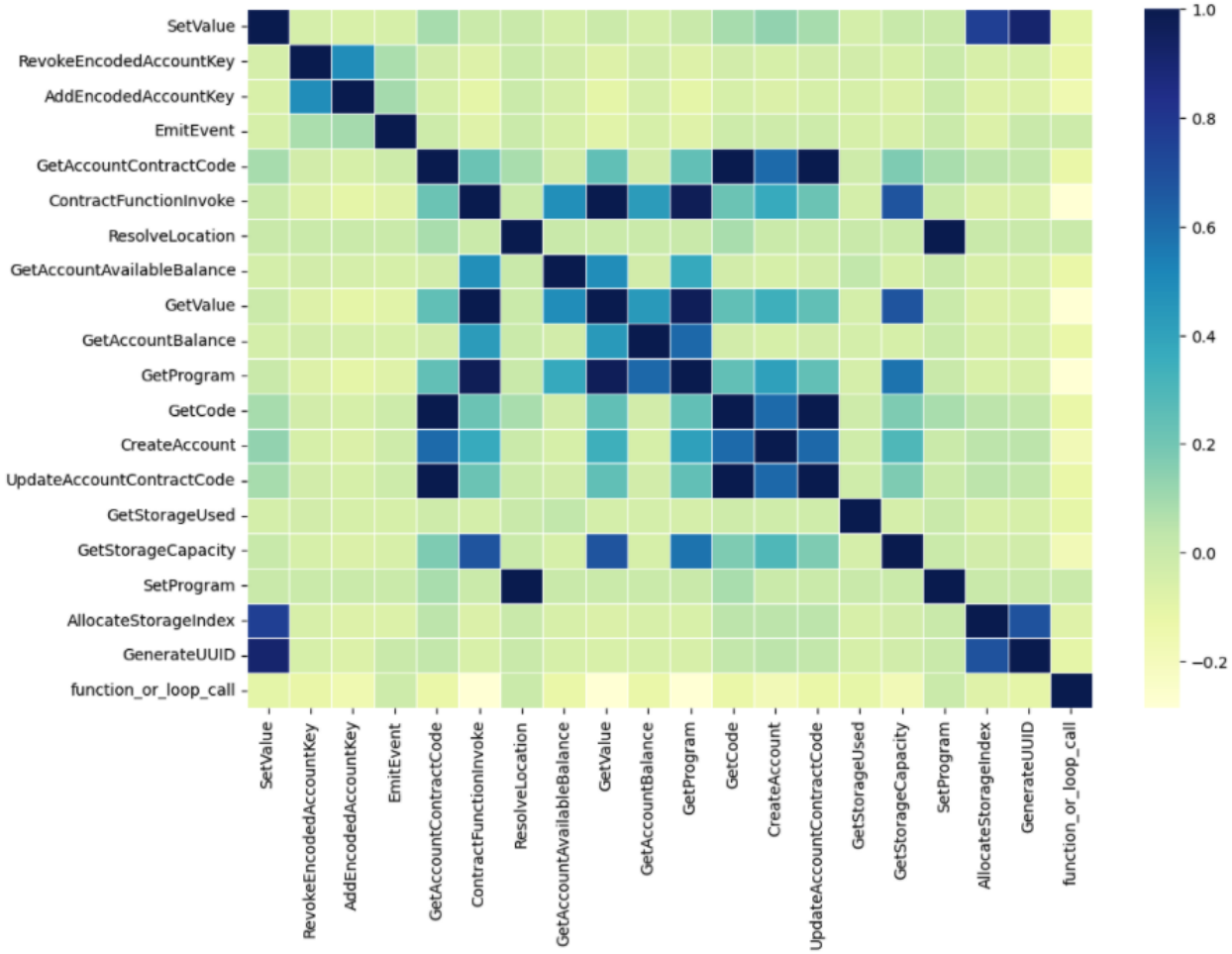


b. Most highly correlated variable with dependent variable, ms, was function_or_loop_call, which exhibited a linear relationship with target with significantly error [source: machine 2]



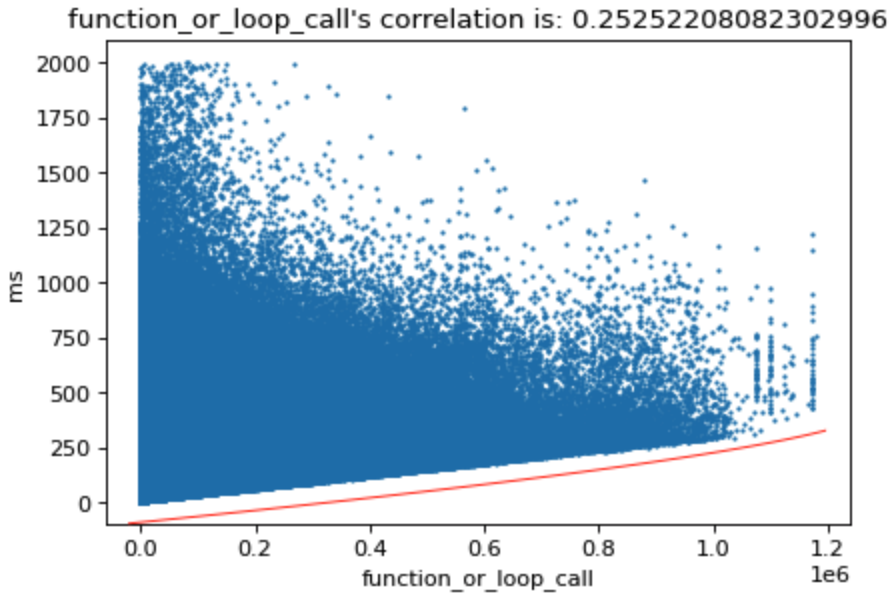
3. Multicollinearity

a. The machine datasets exhibited multicollinearity, as many of the features were highly correlated with one another. This can be seen from the below correlation matrix from machine 2 data. For example, ContractFunctionInvoke, was positively correlated with both GetProgram and GetValue as well as having a strong relationship with GetStorageCapacity. Such features were removed from the model in order to preserve the statistical significance of the coefficients estimated by the linear model. [source: machine 2]



Data Pre-Processing

1. Machine 1 and machine 2 simulated data
2. Purpose:
    a. ms is floored by the global minimum value against each variable while with a lot of noise



    b. obtain the global minimum execution time records for prediction
3. Methodology
    a. iterate through all the function_or_loop_call values
    b. for the same function_or_loop_call value, only keep the record with min ms value
    c. iterate through all the GetValue
    d. for the same GetValue value, only keep the record with min ms value
4. Output:
    a. 12000 records for model development

Data Split

1. train data: randomly selected 9600 records from data processing output
2. test1 data: randomly selected 2400 records from data processing output
3. test2 data: all testnet data

Champion Model

Execution effort = ***'function_or_loop_call'*** * 0.0004789016465827949 + ***'GetValue'*** * 0.0002466779730553598 + ***'CreateAccount'*** * 0.8660748805785956 + ***'SetValue'*** * 0.0002335080887671281

Calibration:

1. based on the goodness of fit plot on test2 data(all testnet data), the model's prediction scale is not biased so calibration is not needed

Champion Model Statistics

Highlights:

1. all the coefficients are positive
2. all the coefficients' p value is very low, which indicate the variable coefficients are statistically significant

OLS Regression Results

Dep. Variable:	ms	R-squared (uncentered):	0.791
Model:	OLS	Adj. R-squared (uncentered):	0.791
Method:	Least Squares	F-statistic:	9376.
Date:	Tue, 22 Mar 2022	Prob (F-statistic):	0.00
Time:	22:58:28	Log-Likelihood:	-57940.
No. Observations:	9920	AIC:	1.159e+05
Df Residuals:	9916	BIC:	1.159e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
function_or_loop_call	0.0005	1.04e-05	51.251	0.000	0.001	0.001
GetValue	0.0003	3.9e-06	74.166	0.000	0.000	0.000
CreateAccount	0.9672	0.034	28.113	0.000	0.900	1.035
SetValue	0.0003	6.23e-06	45.960	0.000	0.000	0.000

Omnibus:	8122.115	Durbin-Watson:	1.997
Prob(Omnibus):	0.000	Jarque-Bera (JB):	348873.749
Skew:	3.648	Prob(JB):	0.00
Kurtosis:	31.121	Cond. No.	1.24e+04

Note: the statistics above is based on the train data and is not comparable to the goodness of fit statistics mentioned in the next section

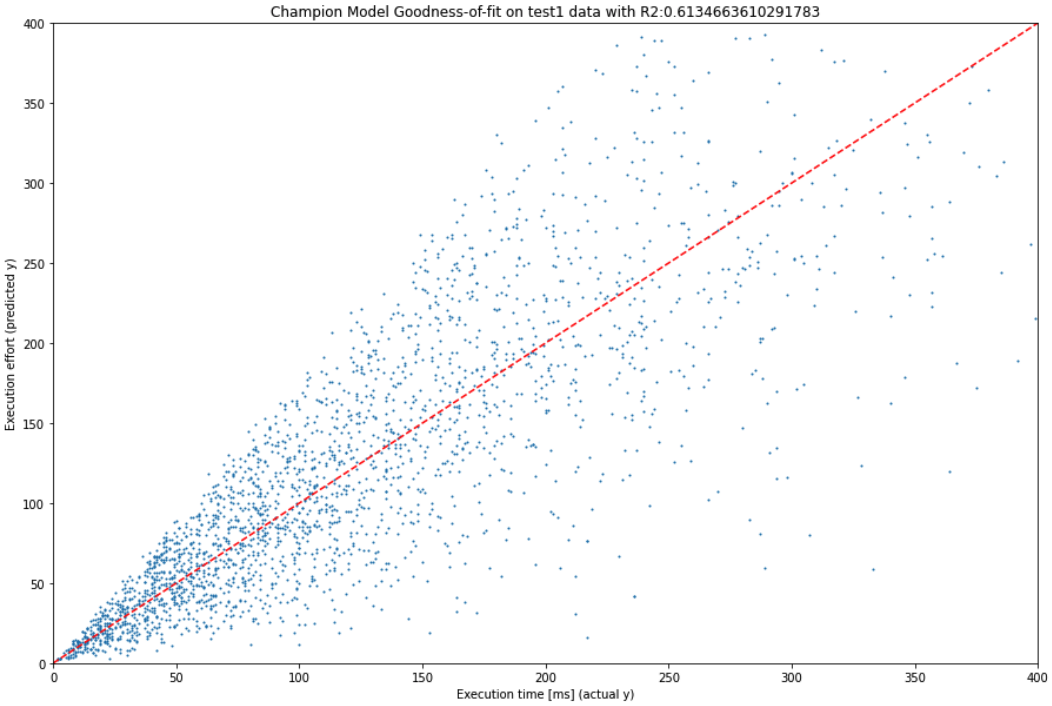
Model Goodness of Fit on test1 and test2 data

Highlights:

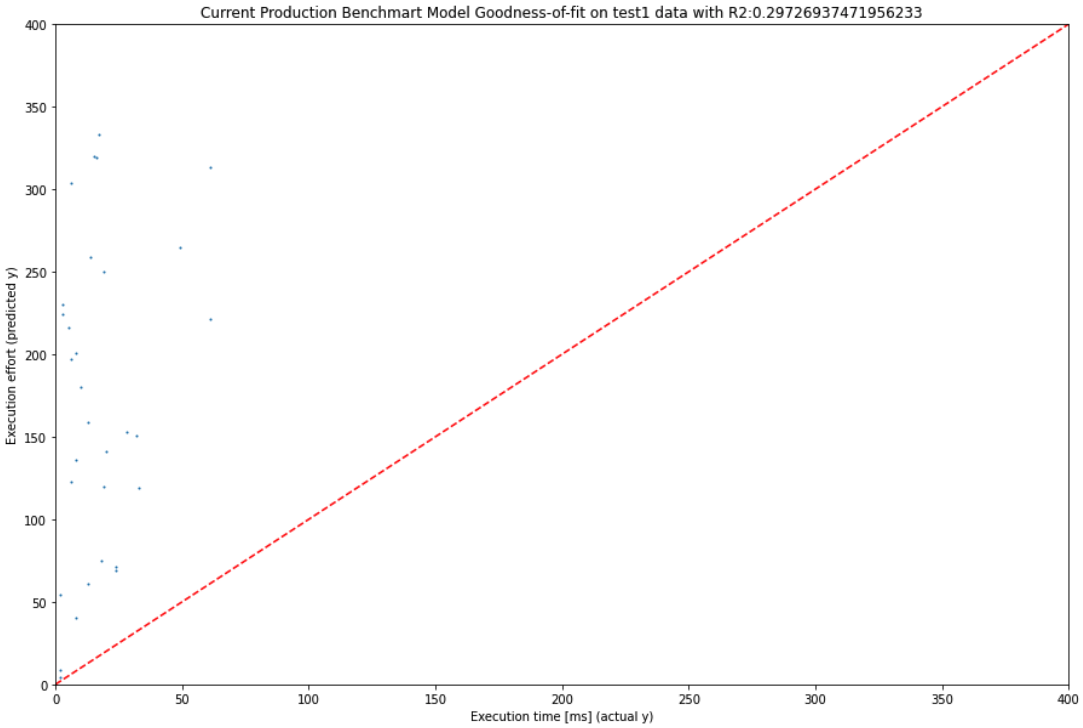
1. comparing plot 1 vs 2 and plot 2 vs 3, the champion model can when compared to current production benchmark model

a. lower the transaction fee
b. better to predict the ms based on R^2

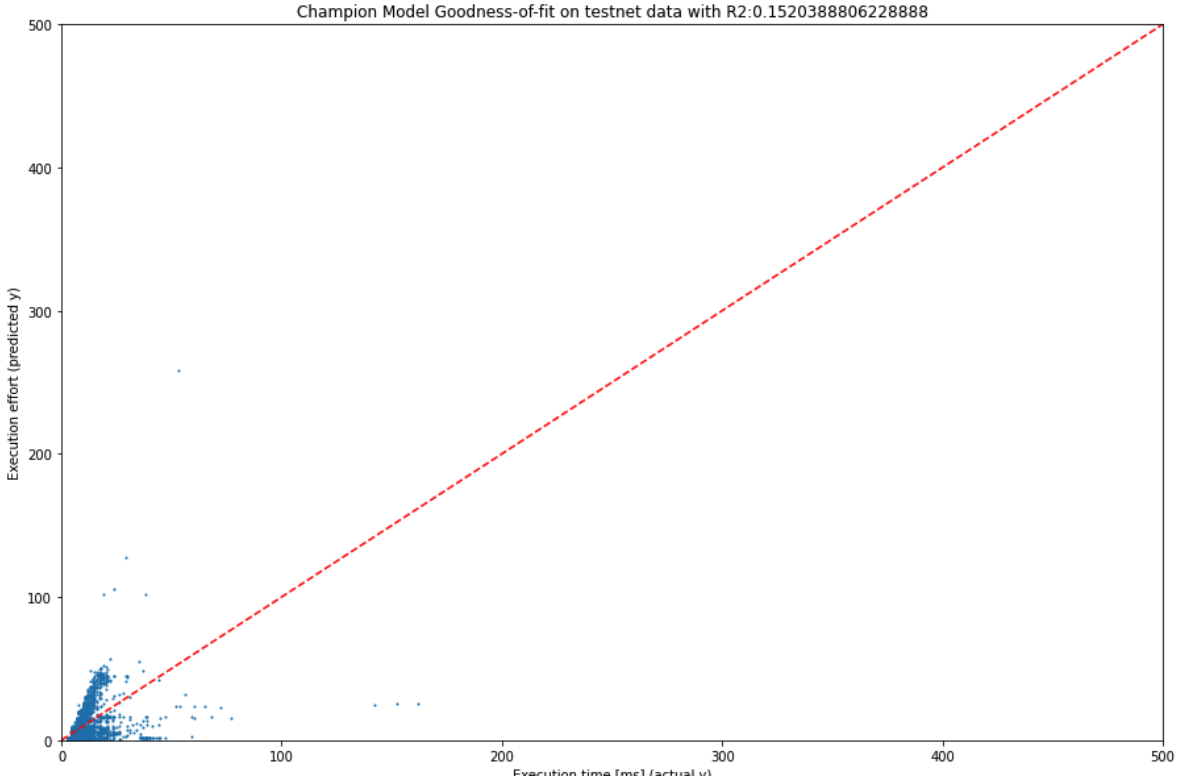
Champion model performance on test1 data



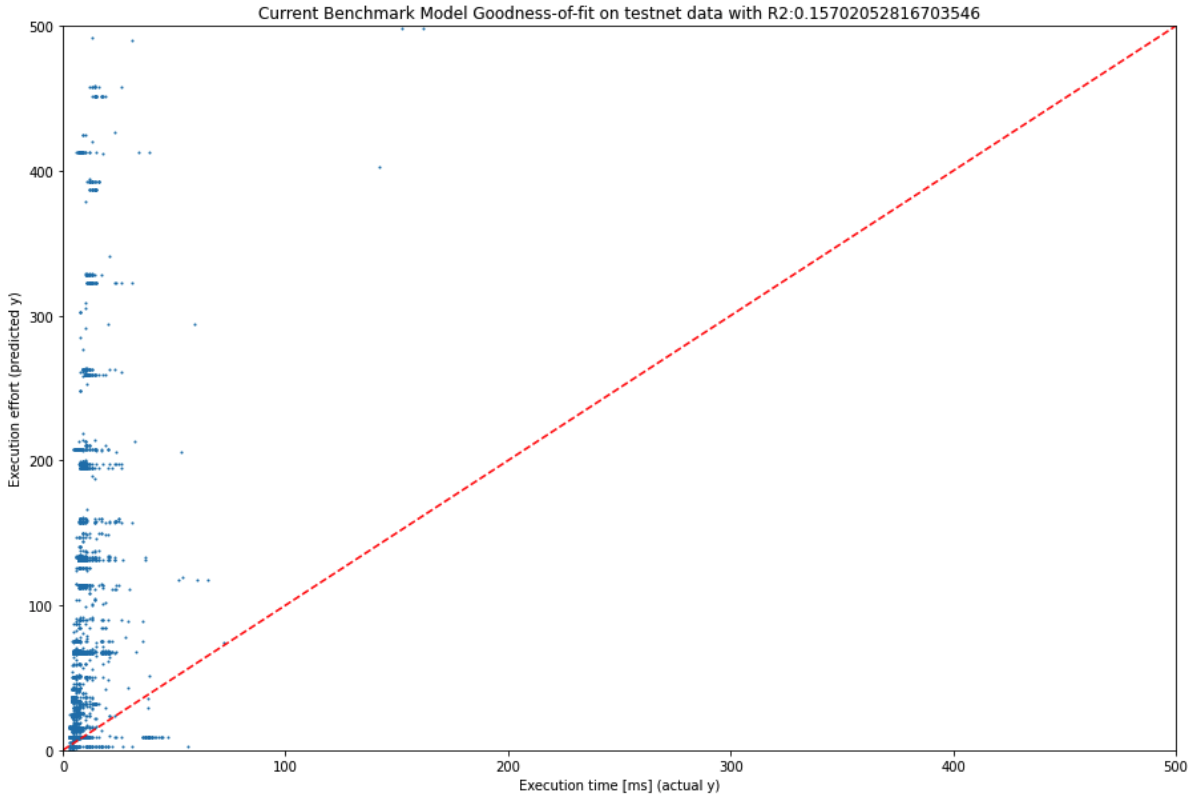
Current production benchmark model performance on test1 data



Champion model performance on test2 data

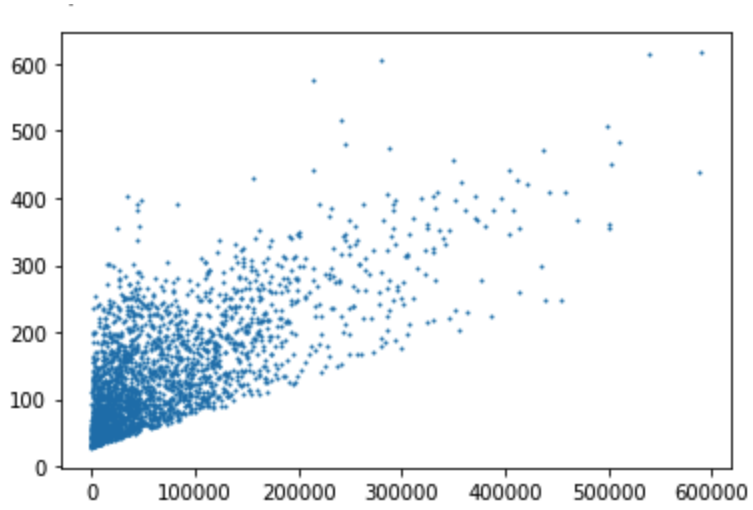


Current production benchmark model's performance on test2 data



Appendix

1. champion model predicted ms vs. function_or_loop_call on test1



2. champion model actual ms vs. function_or_loop_call on test1