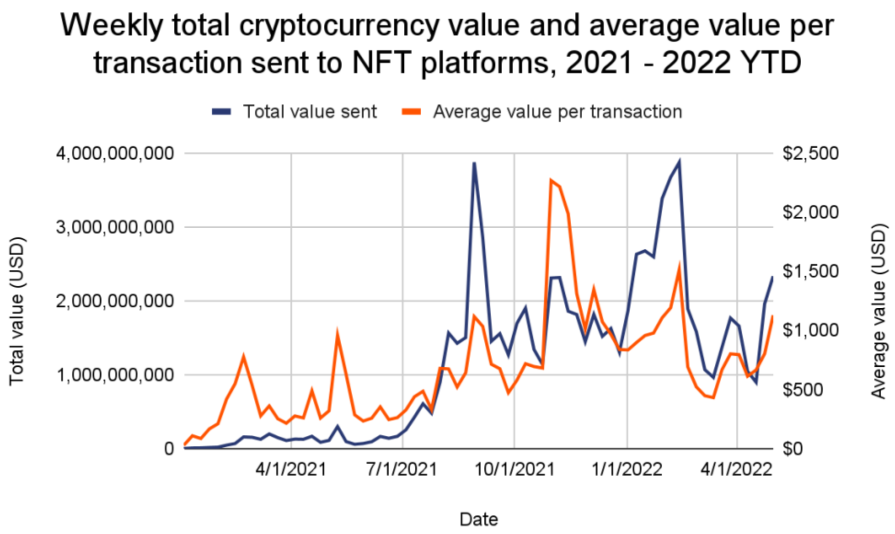
NFT Price Prediction

CS376 Machine Learning

20170435 윤지언 20170653 차정엽 20170783 이동규 20190584 정서경

Goal

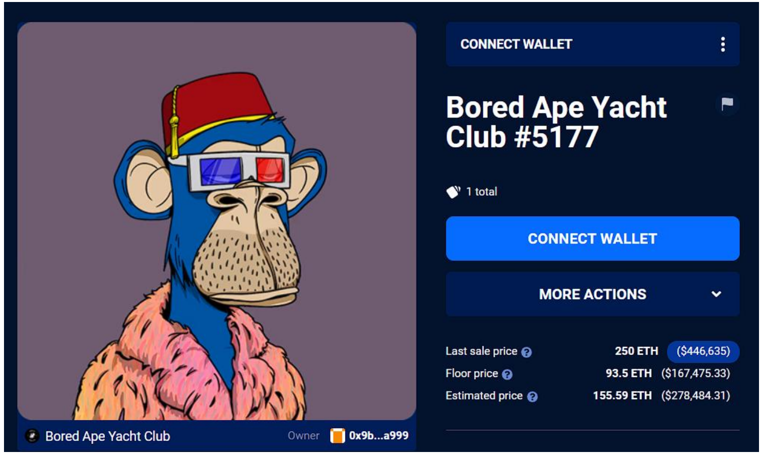


- NFT market is expanding rapidly.
- Pricing and estimation of NFT is not well developed yet because of its subjectivity.
- Our goal is provide effective method to estimate future price of different NFTs.

1) https://fortune.com/2022/05/05/nft-market-sales-not-flatlining-data-shows/

Related Works

Study	HR	RSR	VAR	ML	WL
(Kong & Lin, 2021)	*				
(Schaar & Kampakis, 2022)	*				
(Goldberg et al., 2021)	*				
(Nadini et al., 2021)	*	*		*	
(Kireyev & Lin, 2021)	*			*	
(Ante, 2021a,b)			*		
(Dowling, 2022b)					*
(Umar et al., 2022)					*



Most of current analyses are based on Regression

Using historical market data

1) Kraeussl, Roman and Tugnetti, Alessandro, Non-Fungible Tokens (NFTs): A Review of Pricing Determinants, Applications and Opportunities (May 17, 2022).
2) From https://dappradar.com/hub/assets/eth/0xbc4ca0eda7647a8ab7c2061c2e118a18a936f13d/5177

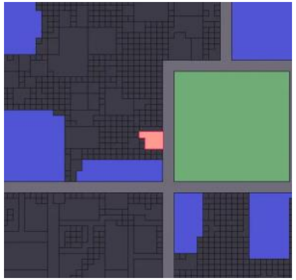
Data Overview

- Total 8 collections, 2 of each from Game, Collectible, Art, Metaverse

- From 2017.01.01 to 2022.05.15

- Internal data
    - # of active market wallets
    - # of sales
    - # of unique buyers
    - Total amount of sales in USD
    - Average USD of the collection
- External data
    - NASDAQ
    - ETH/USD
    - USD index



Linear Regression

$$\hat{y_t} = \hat{\beta}_0 + \hat{\beta}_1 x_{1,t} + \hat{\beta}_2 x_{2,t} + \dots + \hat{\beta}_k x_{k,t}$$

- not genuine forecasts of future value
- just estimate model by linear regression
- can interpret coefficient as feature importance
- feature : {‘Number of sales’, ‘Sales USD’, ‘Active market wallets’, ‘Unique buyers’}
- target variable : {‘Average USD’}

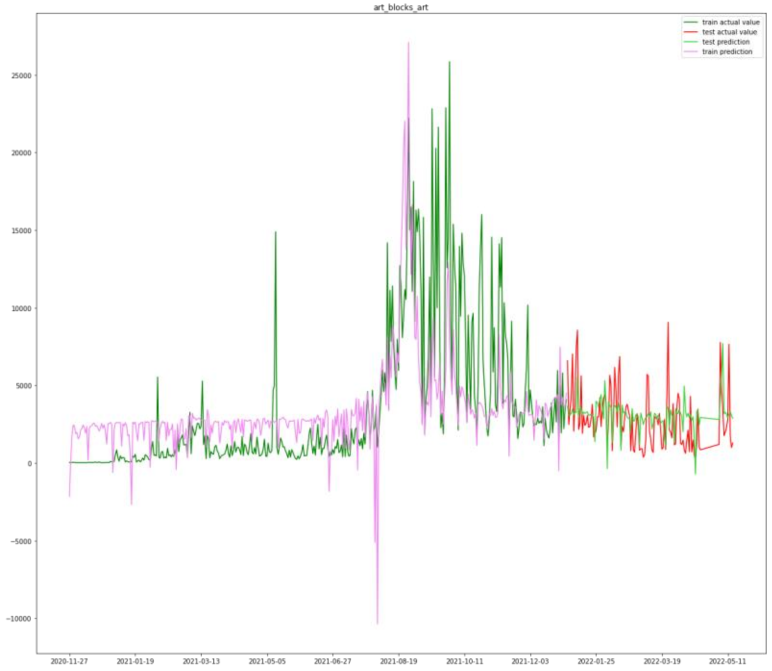
Evaluation

- $$total\,RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y_t} - y_t)^2}$$ (sklearn.metrics.mean_squared_error)
- coefficient of determination(sklearn.metrics.r2_score)

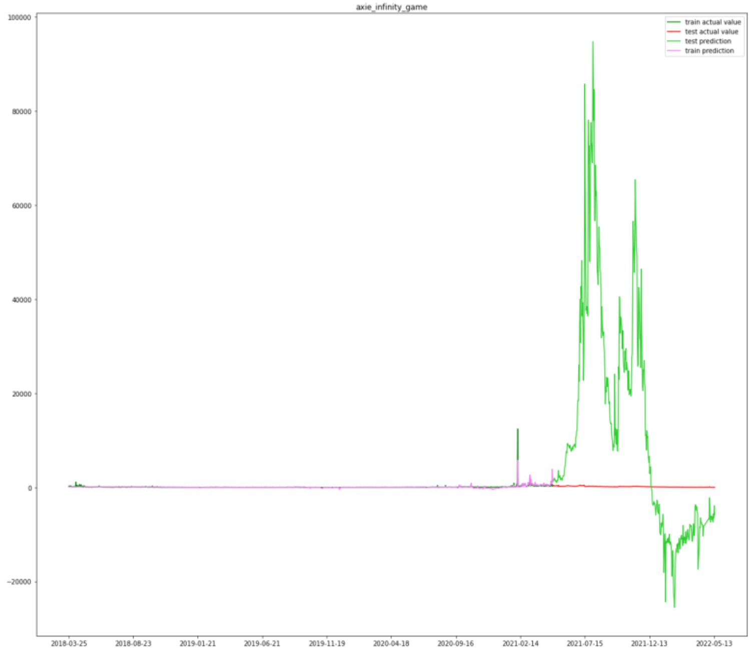
$$R^2 = \frac{\Sigma(\hat{y_t} - \bar{y})^2}{\Sigma(y_t - \bar{y})^2}$$

Linear Regression

RMSE: 1716.8893388316112
Coefficient of Determination: 0.15170165831779292



RMSE: 26837.011415394452
Coefficient of Determination: -73390.40486717146

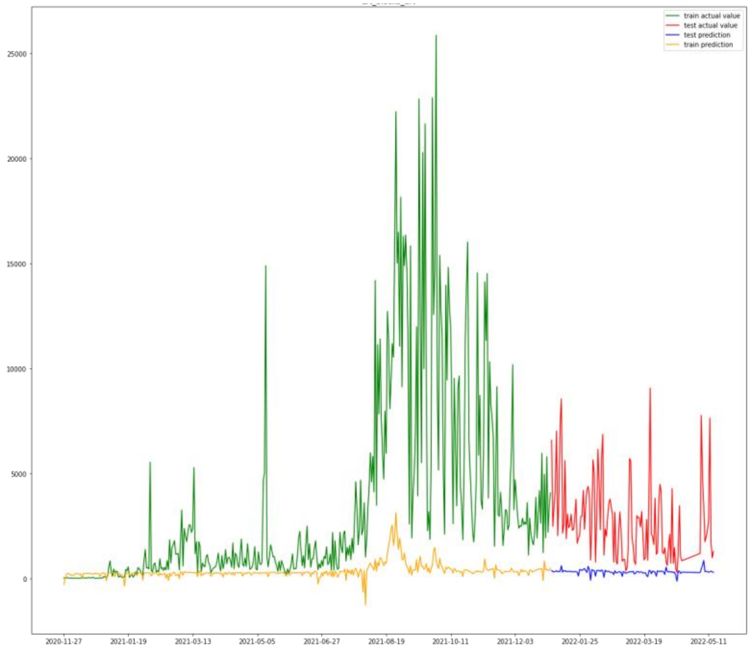


Feature scaling

- standard feature scaling (sklearn.preprocessing.StandardScaler)
- minmax feature scaling(sklearn.preprocessing.MinMaxScaler)

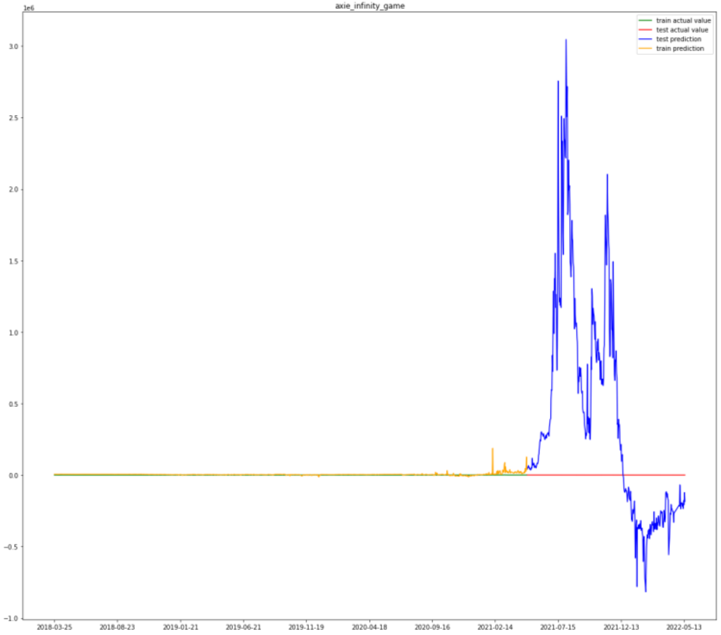
RMSE: 2920.7048660522487

Coefficient of Determination: -1.4549313883860453



RMSE: 78.55120162980465

Coefficient of Determination: -73390.40486717236



Lasso Regression (=L1 Regularization)

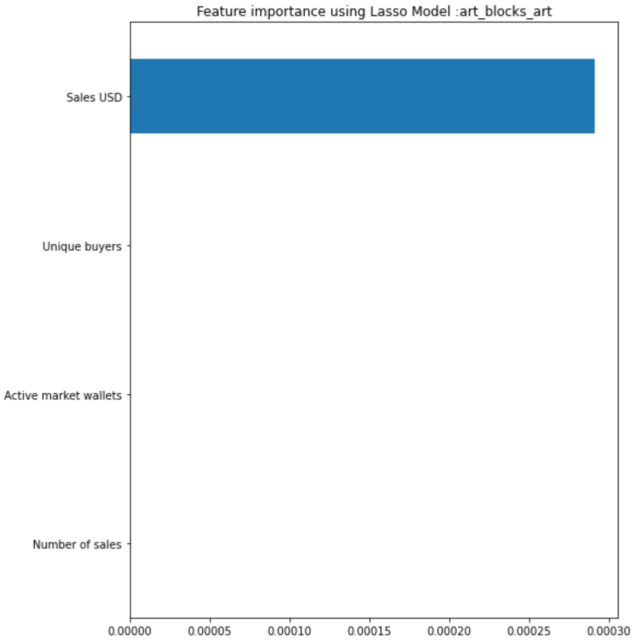
- automatic feature selection

$$L(w) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left(w^T x^{(n)} - y^{(n)}\right)^2 + \lambda\sum_{i}^{d}|w_d|$$

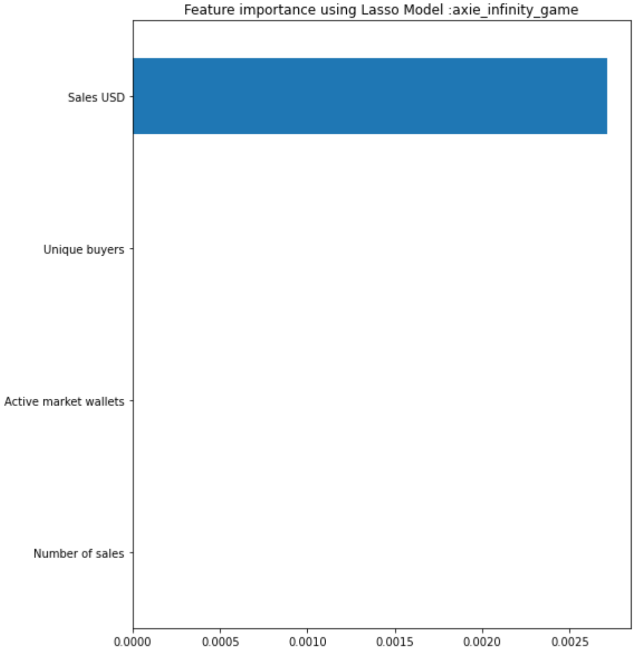
- sklearn.linear_model.LassoCV

Lasso Regression (=L1 Regularization)

```
Best alpha using built-in LassoCV: 1418431045.719116
Best score using built-in LassoCV: 0.139841
1728.8503341547337
Lasso picked 1 variables and eliminated the other 3 variables
```



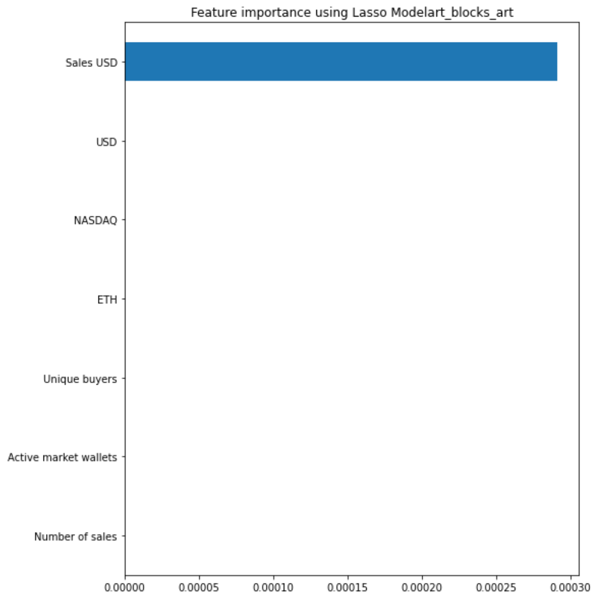

```
Best alpha using built-in LassoCV: 48899.083275
Best score using built-in LassoCV: -137176.442410
36690.43840677007
Lasso picked 1 variables and eliminated the other 3 variables
```



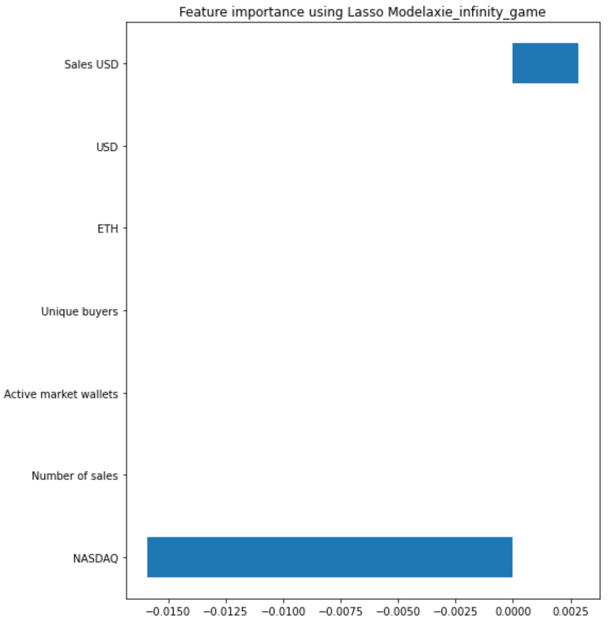
additional feature +Lasso Regression

- add new feature : {‘ETH’, ‘NASDAQ’, ‘USD’}

```
Best alpha using built-in LassoCV: 1420545063.481934
Best score using built-in LassoCV: 0.139417
1729.275958047487
0.13941728794359343
Lasso picked 1 variables and eliminated the other 6 variables
```

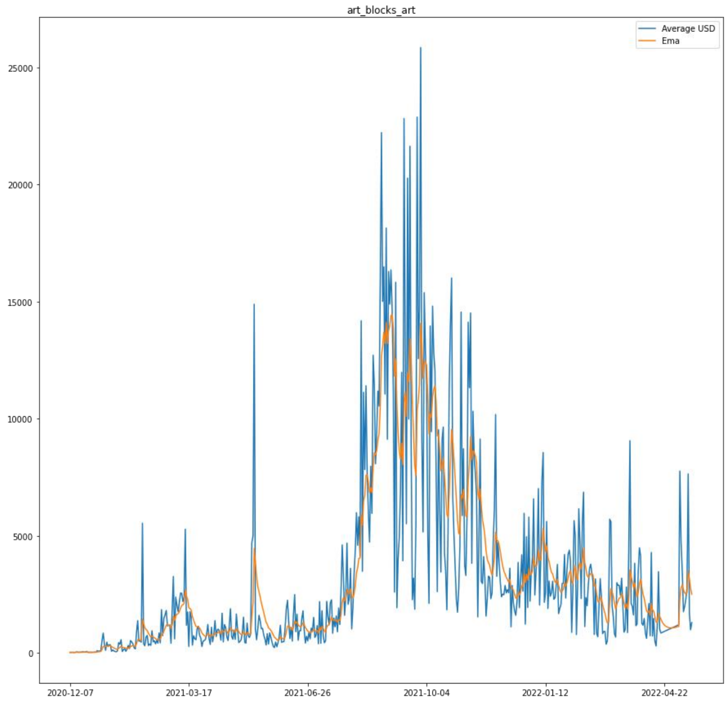


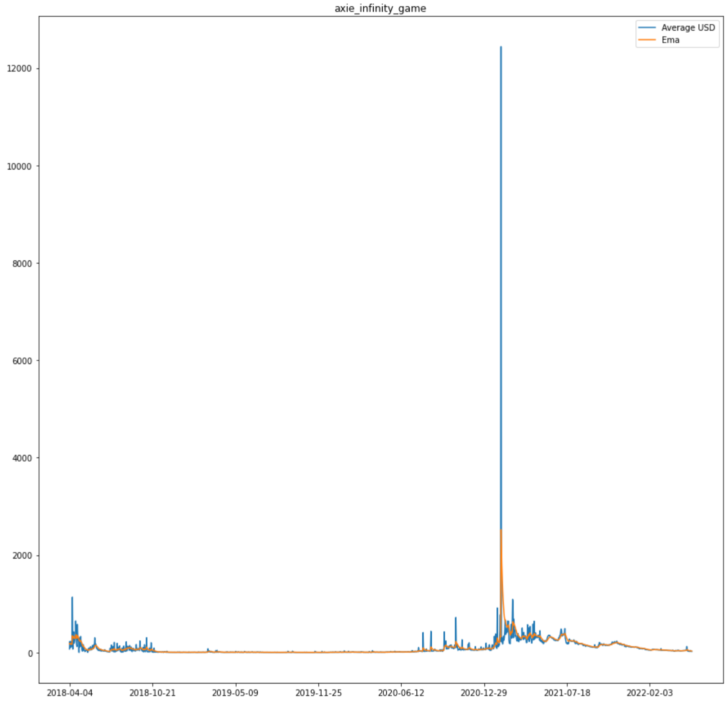

```
Best alpha using built-in LassoCV: 49035.535694
Best score using built-in LassoCV: -150780.862347
38420.10681044345
-150780.86234651148
Lasso picked 2 variables and eliminated the other 5 variables
```



EMA(Exponential Moving Average)

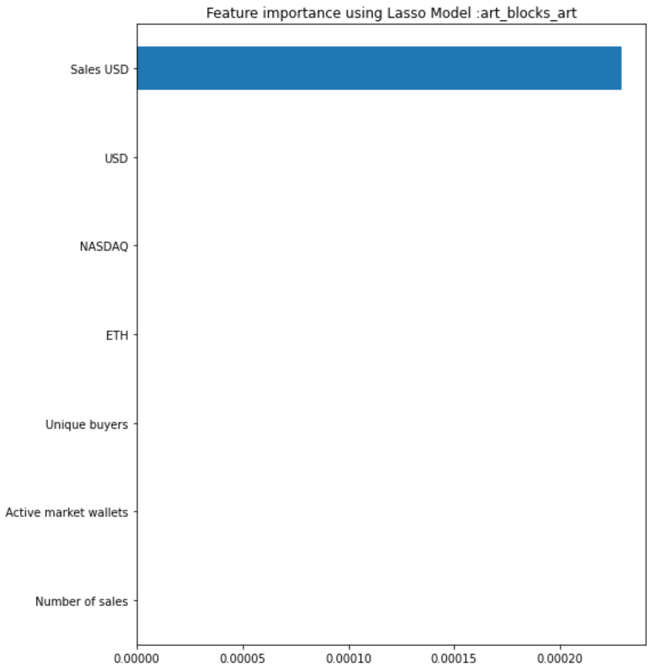
- pandas_ta library ema function
- EMA10



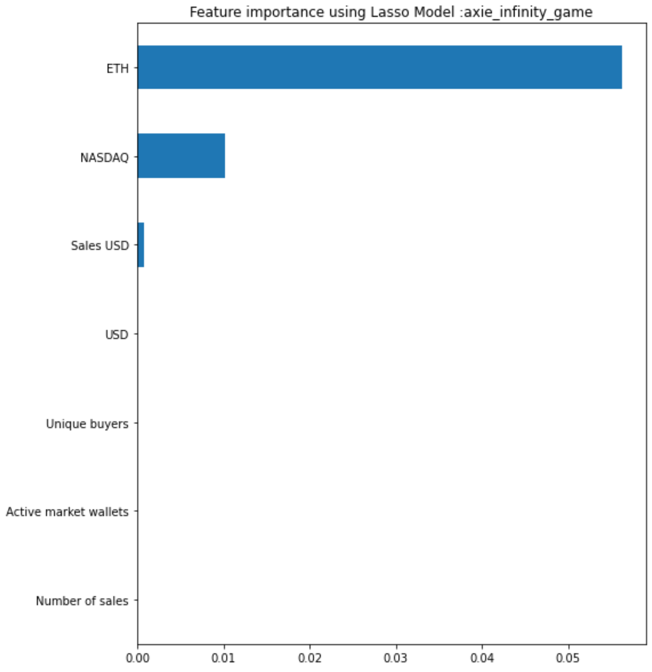


Linear regression for EMA10

Best alpha using built-in LassoCV: 678526102.898727
Best score using built-in LassoCV: 0.028498
934.3928054105266

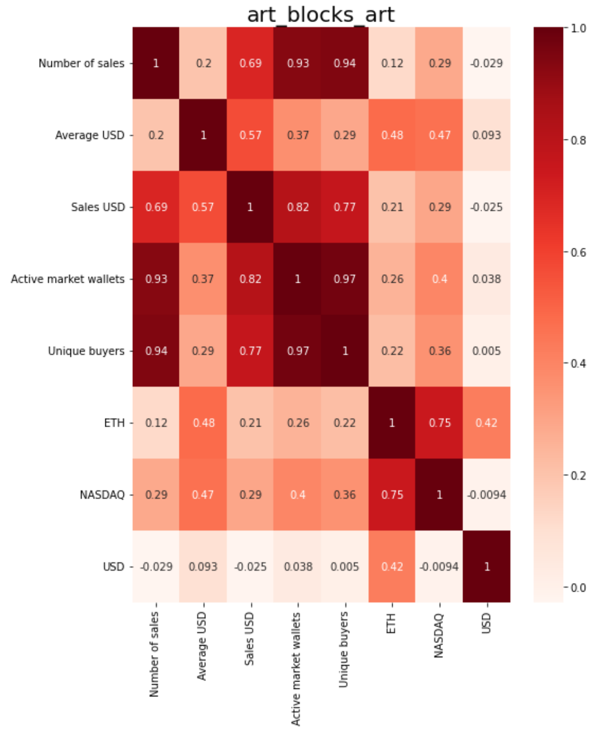


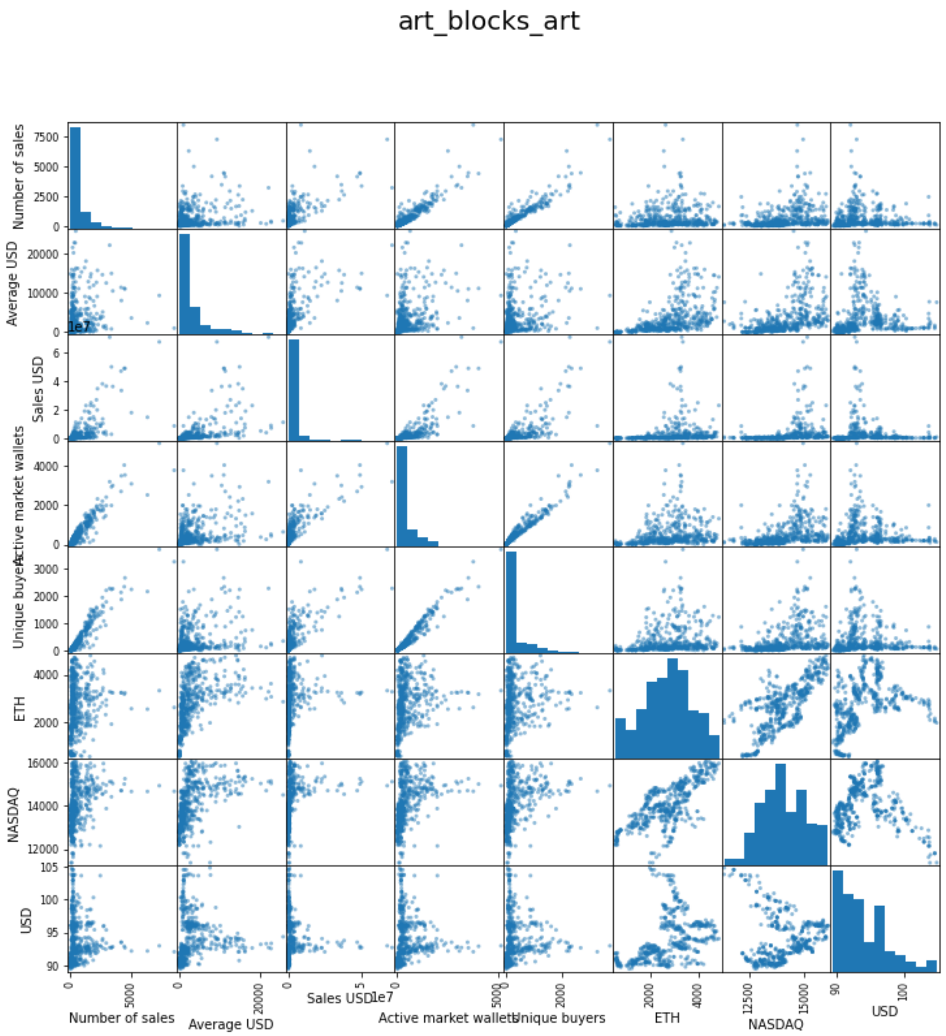
Best alpha using built-in LassoCV: 10753.052293
Best score using built-in LassoCV: -12027.379186
10465.256472109817



why?

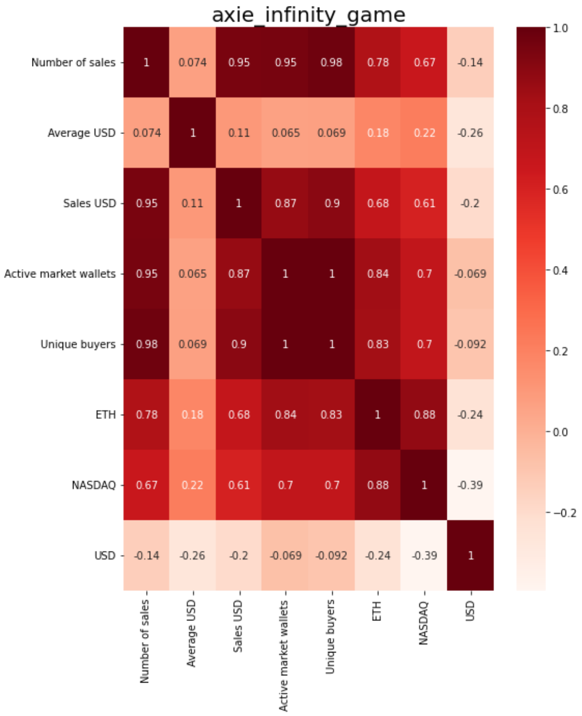
correlation between features

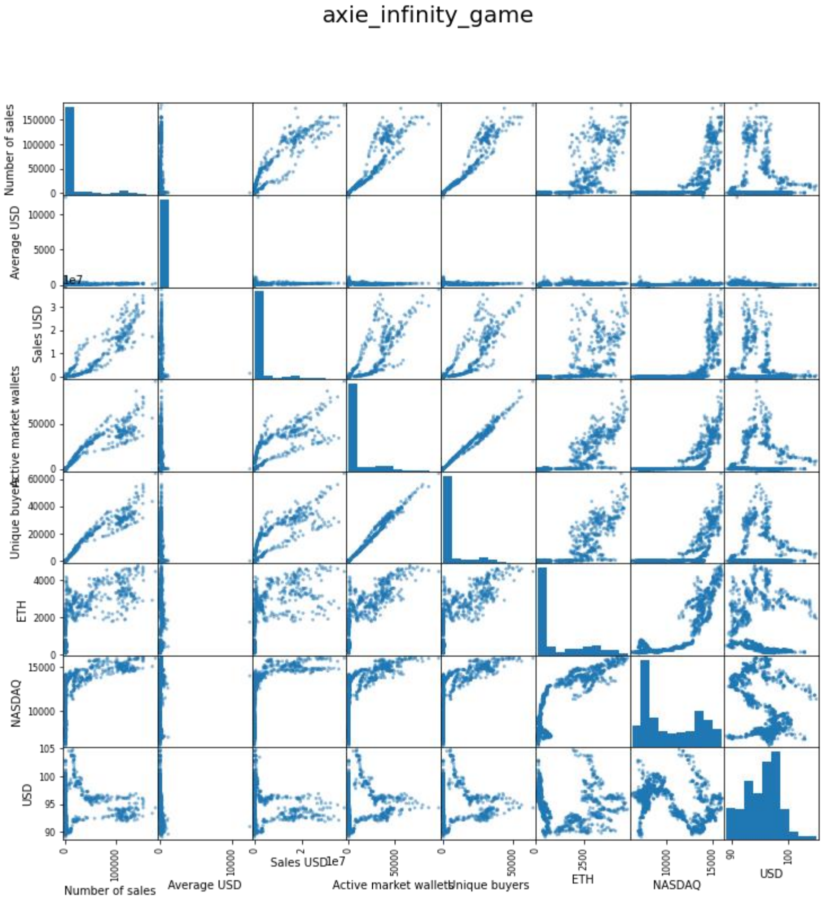




why?

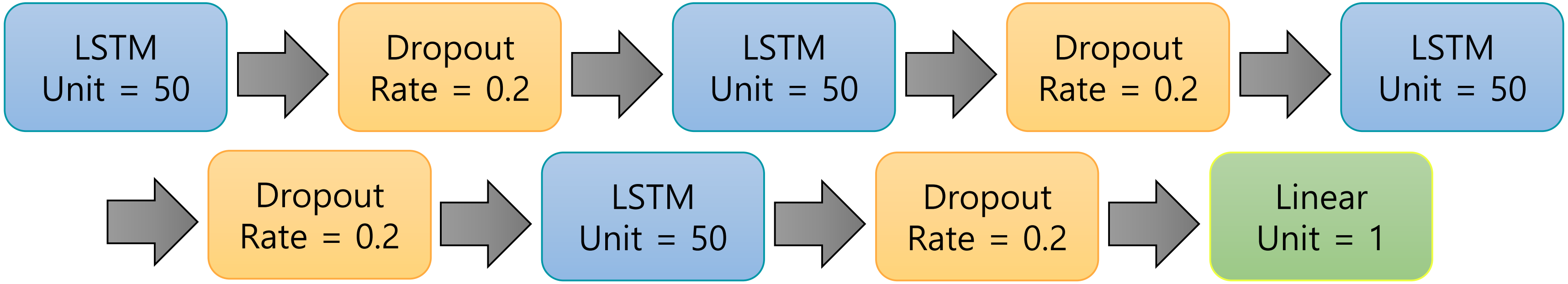
correlation between features



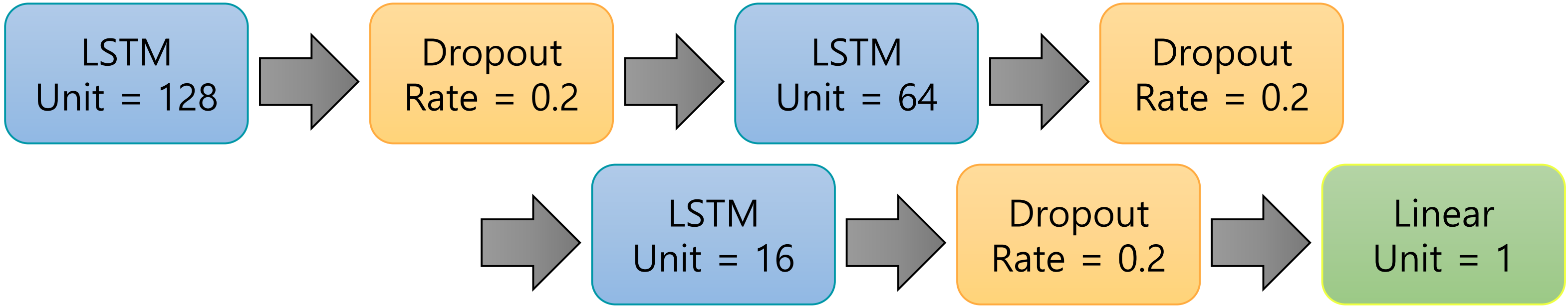


Stacked LSTM

Architecture 1



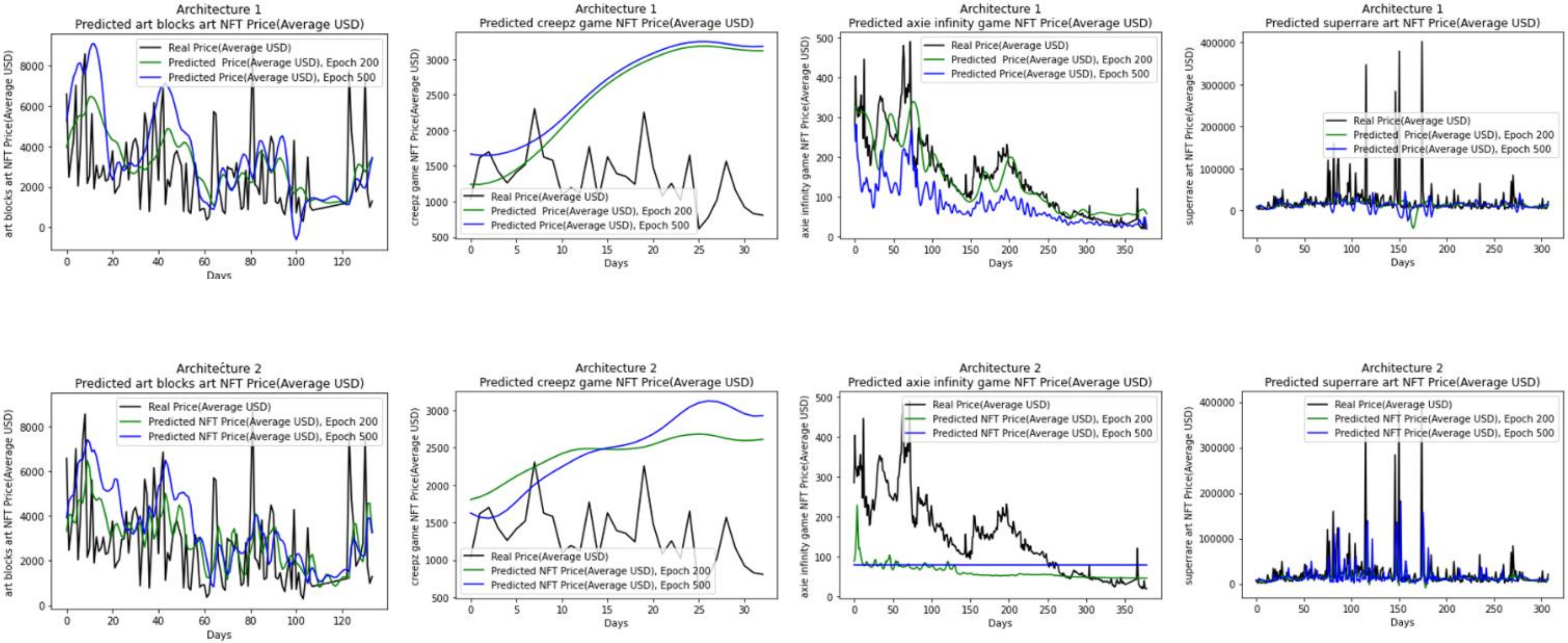
Architecture 2



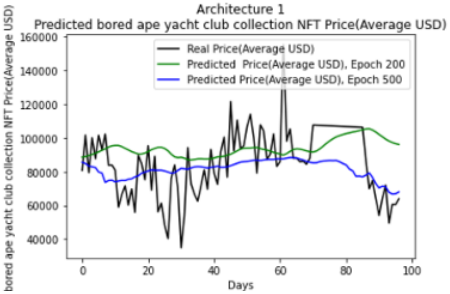
Optimizer: ADAM, Loss: Mean squared Error

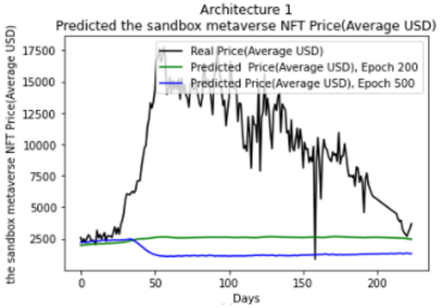
Epoch: 200, 500 each

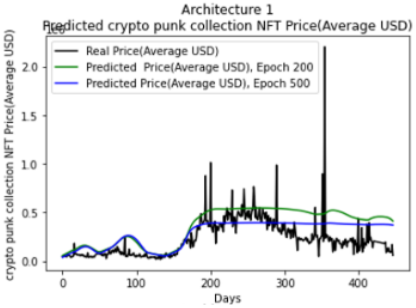
Prediction Graph (1)

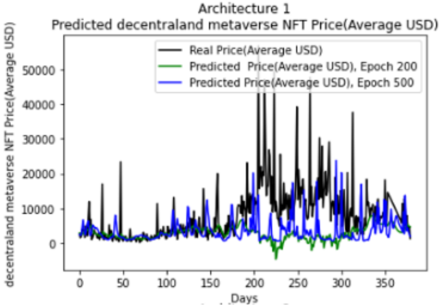


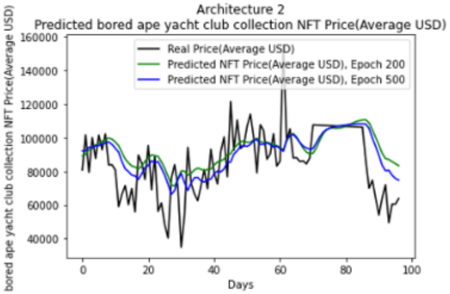
Prediction Graph (2)

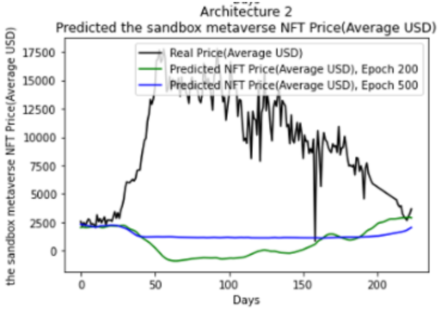


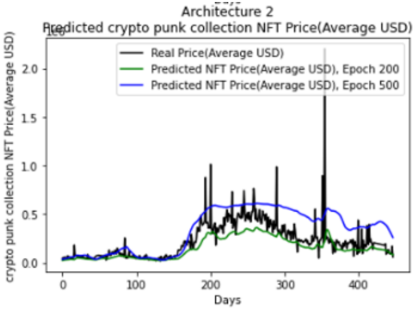


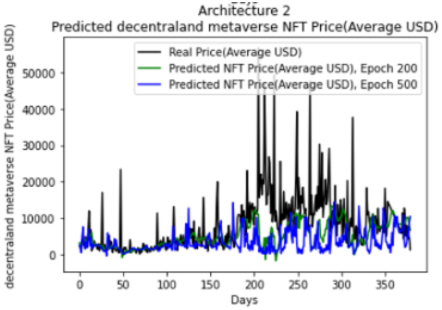








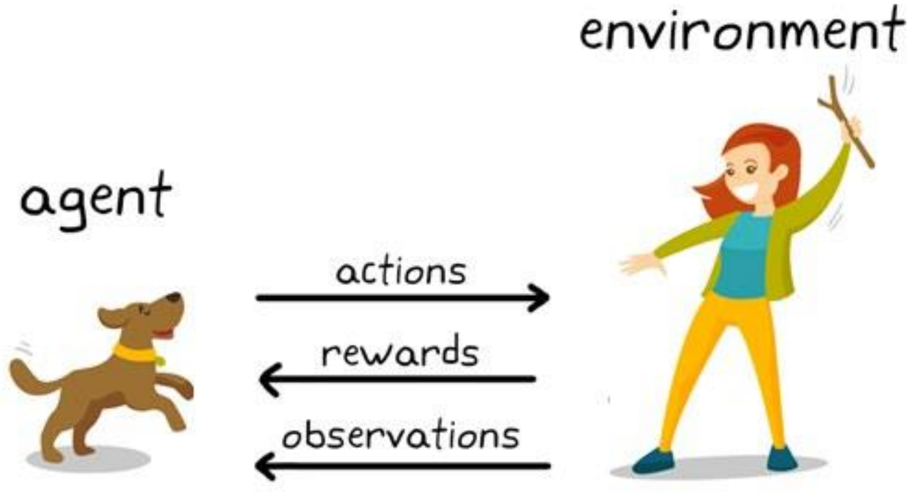




RMSE (Based on Real, unscaled value)

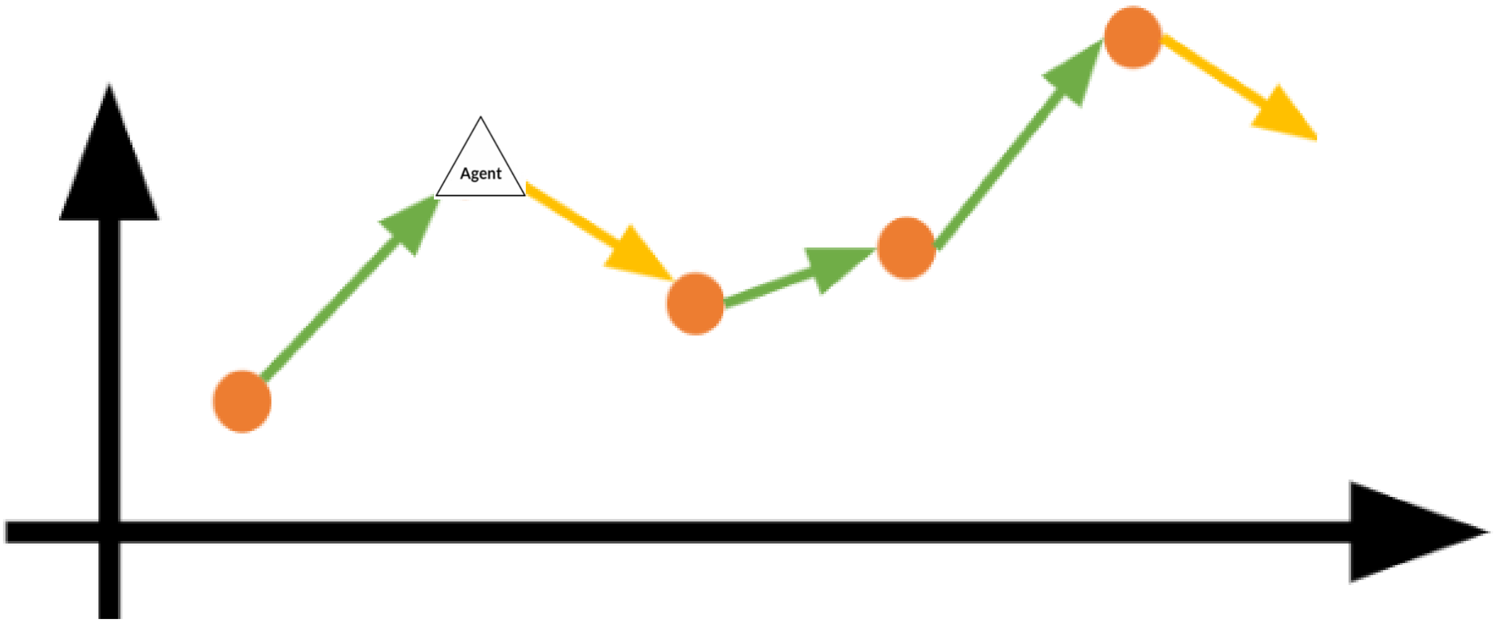
Data Category	RSME				Lowest RMSE
	Architecture 1		Architecture 2		
	Epoch 200	Epoch 500	Epoch 200	Epoch 500	
art blocks art	2035	2589	1879	2249	1879
Creepz game	1464	1521	1206	1355	1206
The sandbox metaverse	8112	9279	10262	9284	8112
Superrare art	43569	45175	44324	48966	43569
Bored ape yacht club collection	21582	18245	18262	16691	16691
Axie infinity game	39	42	40	120	39
Crypto punk collection	204144	163752	151422	191073	151422
Decentraland metaverse	9702	9448	7703	9238	7703

Reinforcement Learning

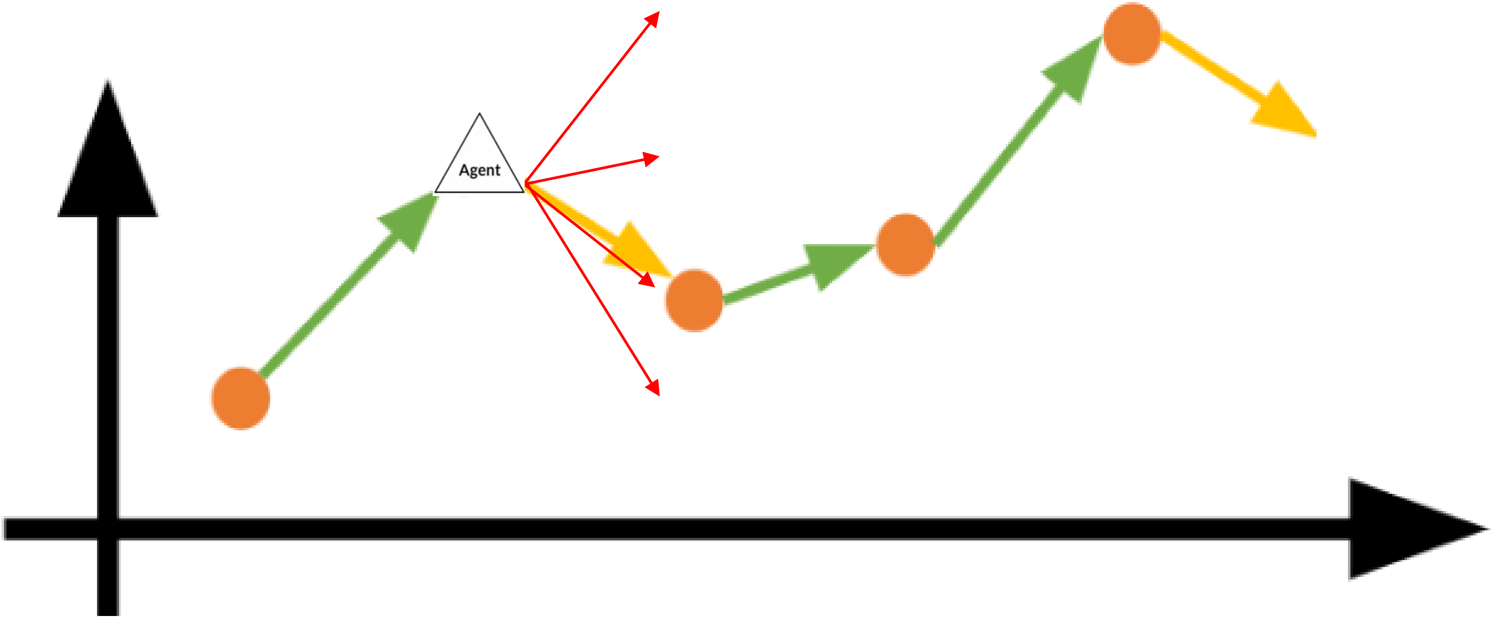


	Number of sales	Average USD	Sales USD	Active mark
4/25/2021	7	183.62	1285.33	8
4/26/2021	70	198.59	13901.4	18
4/27/2021	46	204.71	9416.47	20
4/28/2021	49	215.45	10557.25	19
4/29/2021	68	219.25	14909.21	31
4/30/2021	297	221.81	65877.04	97
5/1/2021	11159	337.48	3765957.9	149
5/2/2021	1706	2301.68	3926670.5	114
5/3/2021	1142	2657.02	3034317	88
5/4/2021	350	2819.33	986765.68	41
5/5/2021	269	2357.26	634103.49	33
5/6/2021	191	2473.23	472386.45	26
5/7/2021	155	2055.3	318572.1	21
5/8/2021	97	2657.96	257822.49	15
5/9/2021	119	2566.07	305362.86	16
5/10/2021	62	2570.01	159340.91	98
5/11/2021	35	1577.69	55219.31	63
5/12/2021	41	2198.69	90146.39	72
5/13/2021	48	1734.96	83277.97	84
5/14/2021	43	2277.31	97924.37	70

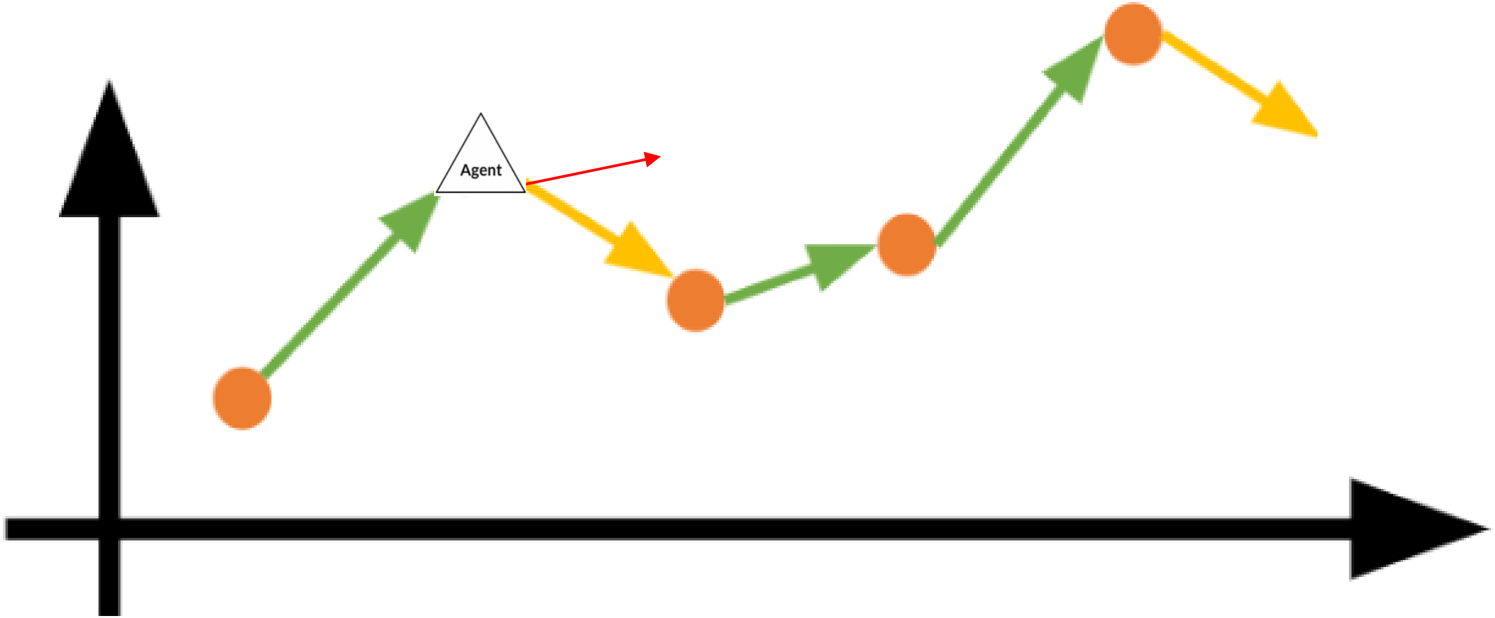
Reinforcement Learning



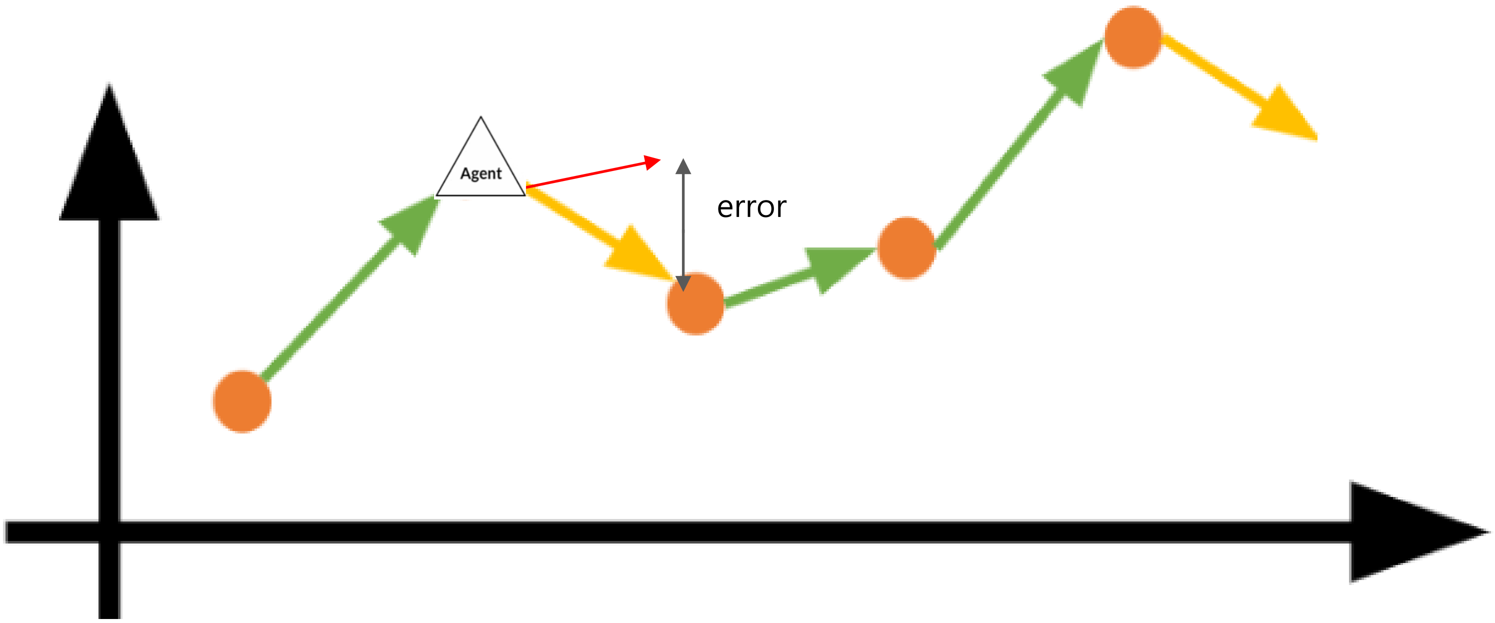
Reinforcement Learning



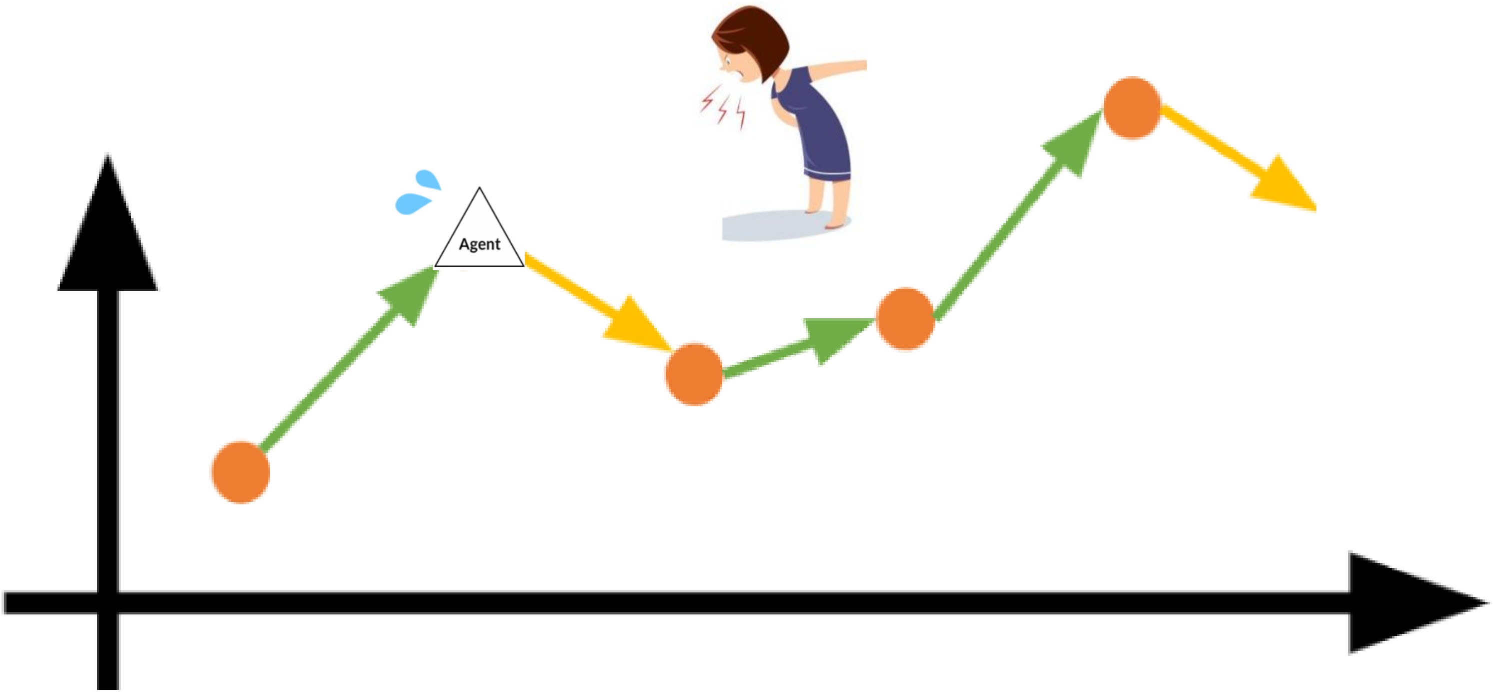
Reinforcement Learning



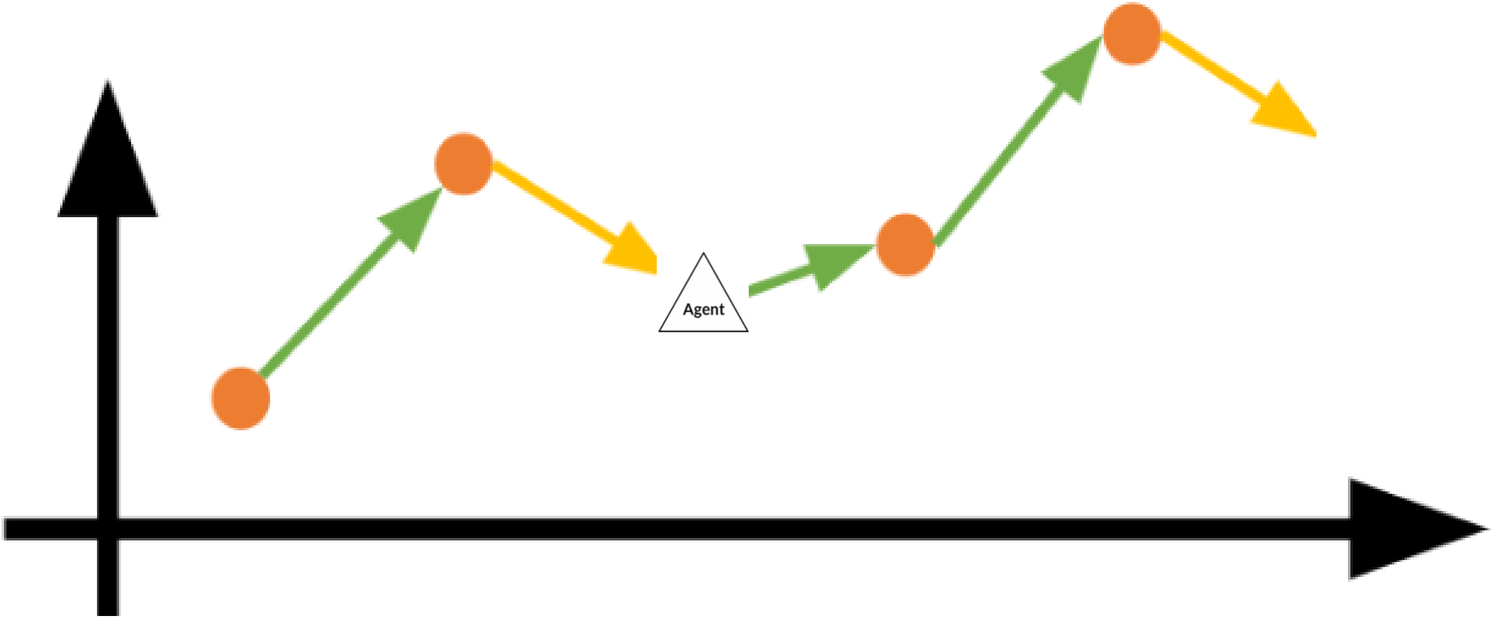
Reinforcement Learning



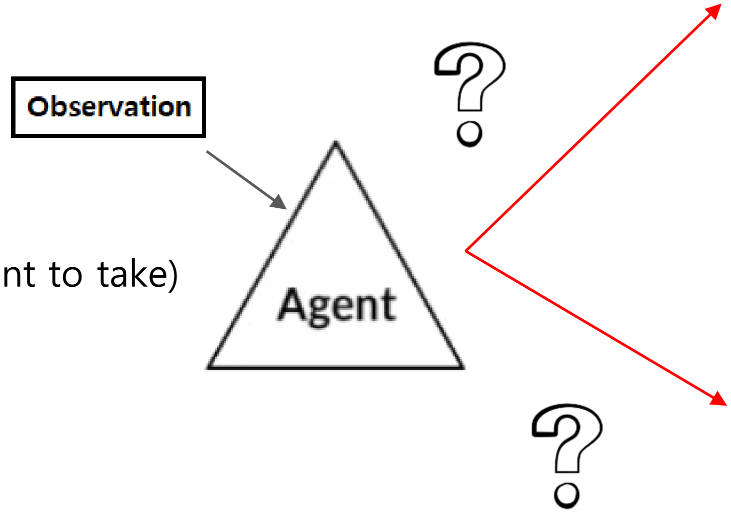
Reinforcement Learning



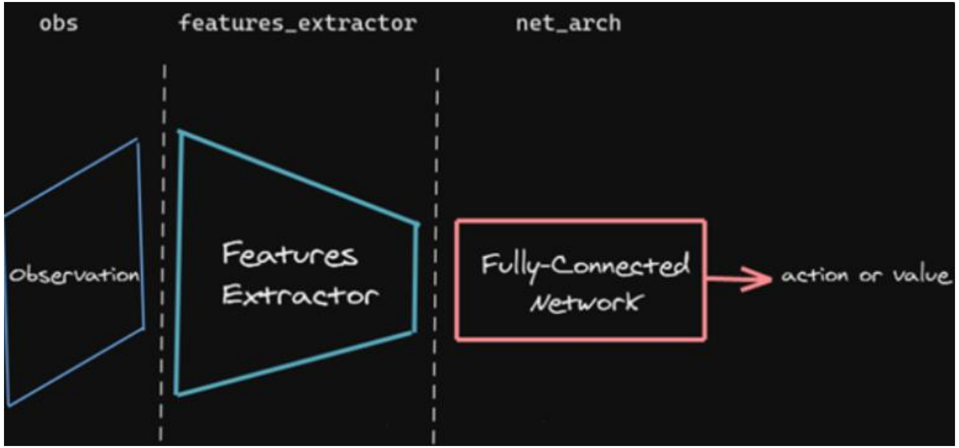
Reinforcement Learning



Policy Network/ Agent



- Inputs an observation, outputs an action (for the agent to take)
- The brain of the agent
- MLP policy: 64 input feature extractor, 64 input FCN



Observation Space

- Basic NFT data features as the observations
    - 75/25 training/test split
- We also tried adding the following values to the observation:
    - The agent’s most recent prediction value
    - Each feature of the most recent observation

Algorithms

Actor-critic RL algorithms that could support continuous observation and action spaces

- PPO (Proximal Policy Approximation)
- A2C (Advantage Actor Critic)

Normalization

- In order for Actor critic algorithms to learn well, the observation, reward and especially the actions, should be normalized to reasonable values

10/20/202	93.56	4053.9	15,121.68
10/21/202	93.77	3971.55	15,215.70
10/22/202	93.64	4168.56	15,090.20
10/23/202	93.64	4082.64	15,090.20
10/24/202	93.64	4220.93	15,090.20
10/25/202	93.81	4131.47	15,226.71
10/26/202	93.95	3923.94	15,235.71
10/27/202	93.8	4288.26	15,235.84
10/28/202	93.35	4421.23	15,448.12
10/29/202	94.12	4324.67	15,498.39
10/30/202	94.12	4290.16	15,498.39
10/31/202	94.12	4322.89	15,498.39
11/1/2021	93.88	4595.01	15,595.92

Reward: 2502342

Normalization of Action

For PPO and A2C, it is critical that the action space be normalized to [-1, 1]

=> Use Tanh as the activation function in the Policy Network to achieve this.

Normalization of Observation Space

Normalizing the observation was not feasible

=> The agent would be able to predict values only in the normalized environment

=> Not able to predict the actual price values

Problem: Action space is limited to [-1, 1], whereas the observation is not

Solution: Using the action value as a **ratio** to be multiplied to the agent’s current location to make a prediction

ex) ratio: 1 + (0.5 * action) => ratio can vary from 0.5 to 1.5!

current agent’s location: $2000, action = 1 => predicted value: 2000 * 1.5 = 3000

Normalization of Reward

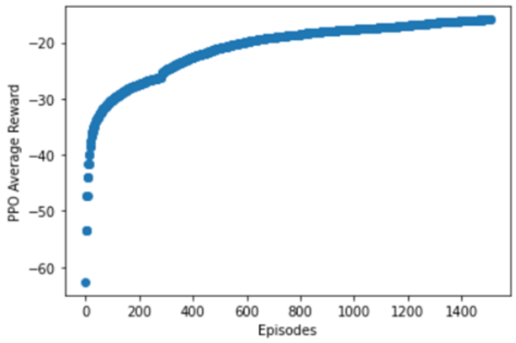
- We used the negative absolute difference as initial reward

- Normalize it to [-2,0] using moving averages and variances

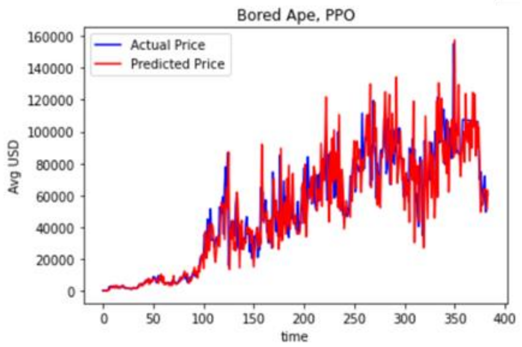
  => we did not want to give positive rewards in order to get the agent to predict as close to the actual value possible

Results

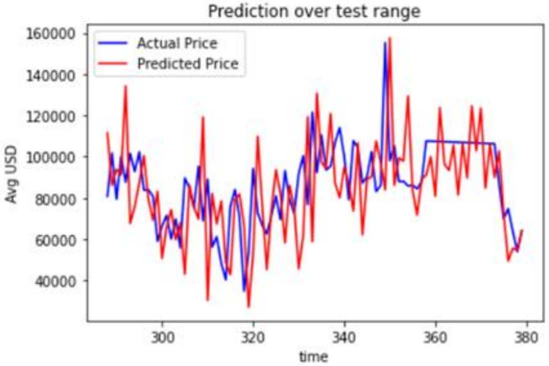
- Bored Ape Yacht Club Collection, PPO, 150000 timesteps



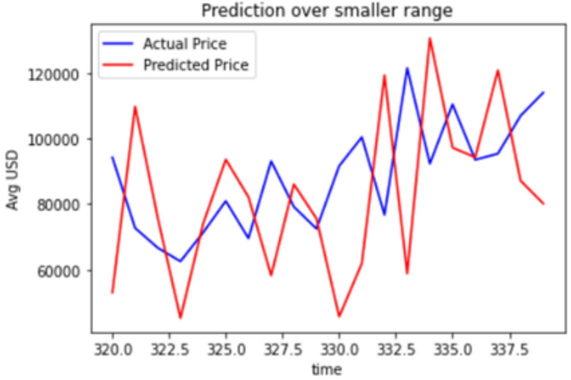
< Training Process >



< Entire dataset >



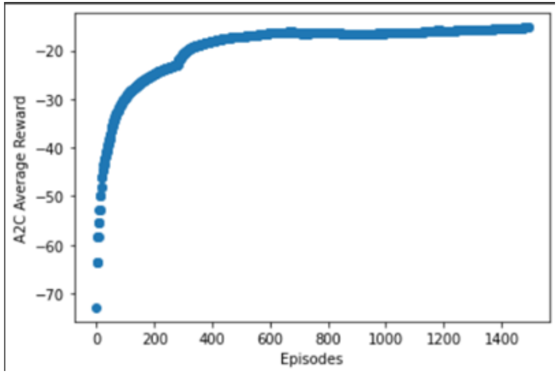
< Test set >



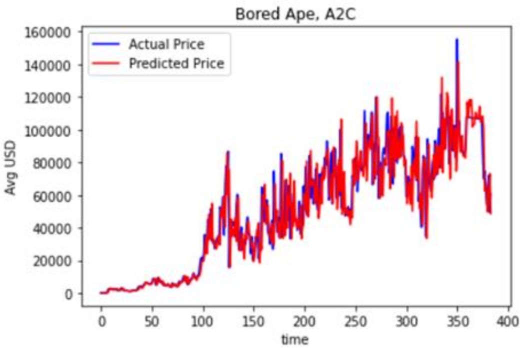
< Smaller set>

Results

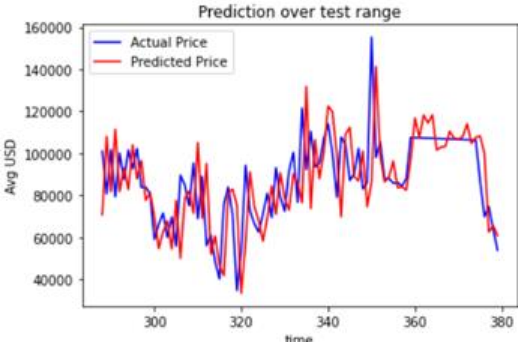
- Bored Ape Yacht Club Collection, A2C, 150000 timesteps



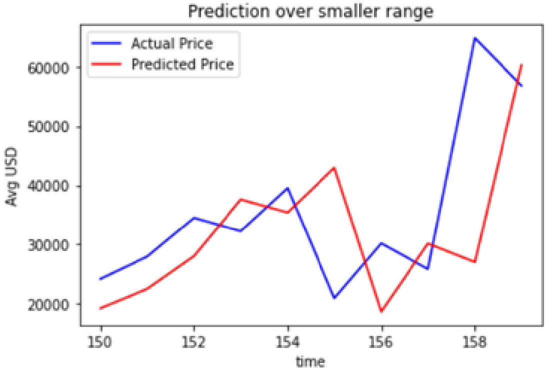
< Training Process >



< Entire dataset >



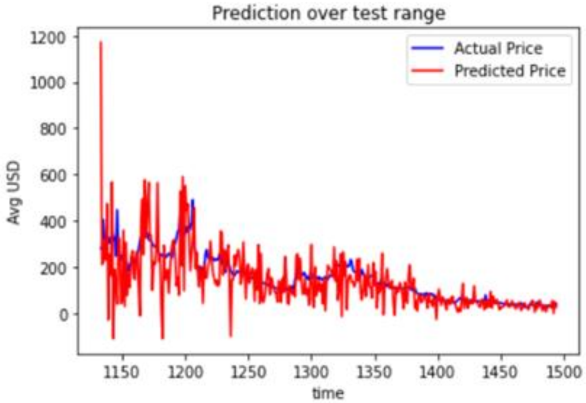
< Test set >



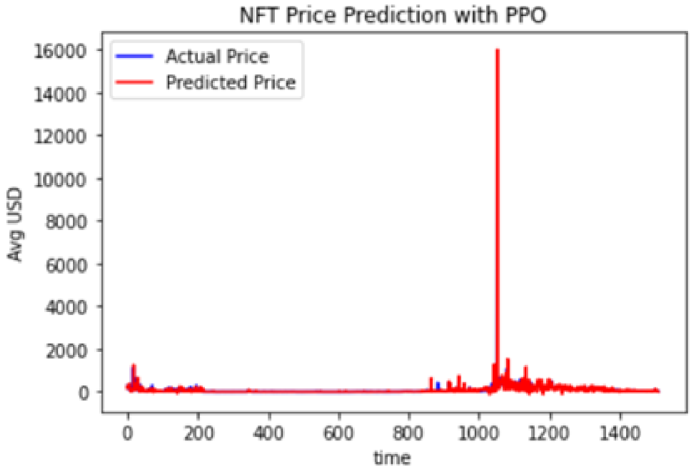
< Smaller set >

Results

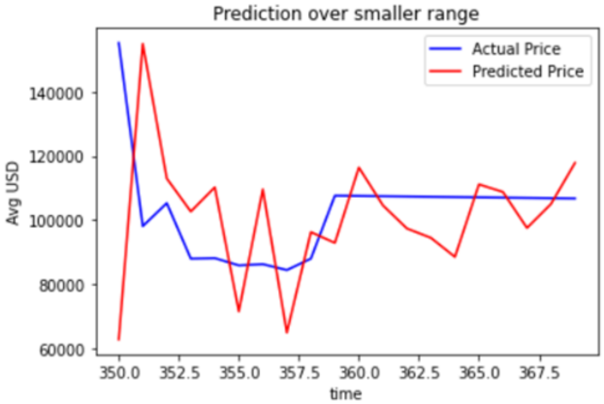
- Axie Infinity Game, PPO, 150000 timesteps



< Test set >



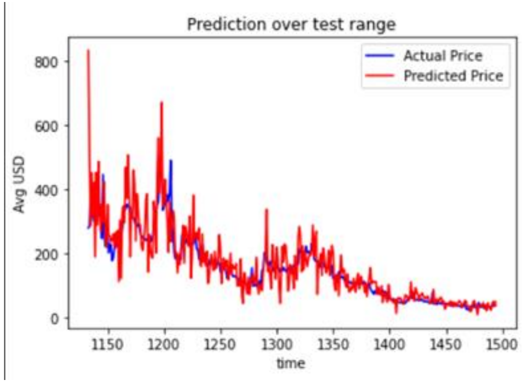
< Entire dataset >



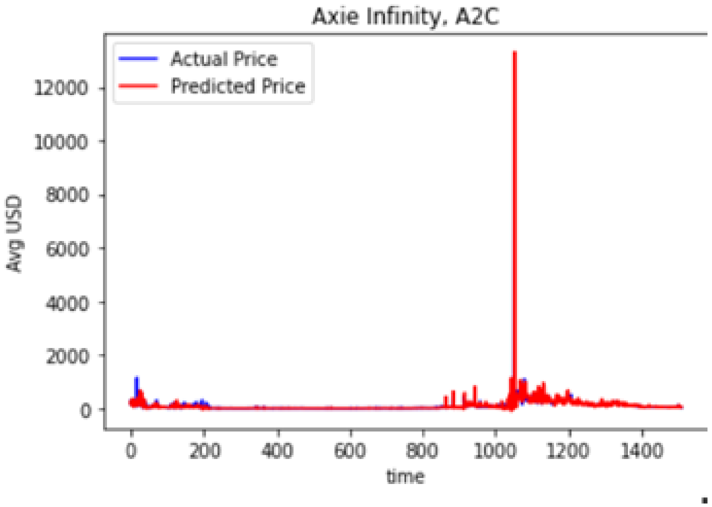
< Smaller set >

Results

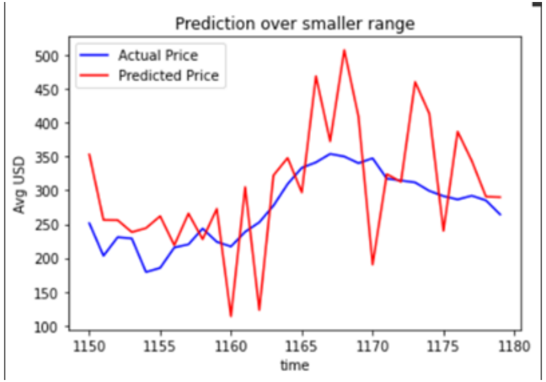
- Axie Infinity Game, A2C, 150000 timesteps



< Test set >



< Entire dataset >



< Smaller set >

Evaluation Metrics: RMSE, R2

```
Bored Ape, PPO
RMSE: 24523.273
R2: -0.525
```

```
Axie Infinity, PPO
RMSE: 100.979
R2: -0.041
```

```
Bored Ape, A2C
RMSE: 19662.734042581218
R2: 0.11082038653062165
```

```
Axie Infinity, A2C
RMSE: 60.950
R2: 0.620
```

Conclusion

	Linear Regression (Baseline)	Stacked LSTM	Reinforcement Learning
RMSE	26837.011415394	40.8	100.979
R^2 score	-73390.40486717	0.8296477128246	0.62

- LSTM showed the best results => best handles sequential data
- Future work: Combining models - Reinforcement learning model that uses LSTM feature extractors!

1) Used Axie_infinity_game data