NFT Valuation Test Documentation

This document contains the description of my approach of in solving the problem and the assumption I made.

1. Merging Data

I started by merging the 3 dataset (`token_sales.csv, token_metadata.csv, eth_usd_fx_rates.csv`). However, before doing this I was able to transform the date into a perfect datetime format for time series.

2. Preprocessing

The missing values were checked and for the **traits** features, we fill with none. The reason for this is that '**we cannot fill missing values based on mode. It is better we fill with none to make every feature unique.'** While the categorical variables I used one-hot encoding for encoding categorical variables.

3. Data Splitting

I dropped the **token_index** that has less than 3 transactions as it will cause abnormalities in the modeling. If we have enough data or features we can add them.

We splitted the data in a unique way by selecting the newest data based on date for each group for **testdata** and the rest was used for the **traindata**.
This will help us predict on the latest data,

4. Training

We adopted **pycaret** for machine learning modeling. The reason for this is for us to be able to find the potential model easily before applying hyper-parameter turning. Below is the model performance and we can observe that Xgboost outperforms other tree-based models. Therefore we can invest our time on Xgboost for hyperparameter turing.

	Model	MAE	MSE	RMSE	R2
xgboost	Extreme Gradient Boosting	5.4631	774.9217	19.8690	0.8223
lightgbm	Light Gradient Boosting Machine	5.6920	828.8517	20.6154	0.8094
et	Extra Trees Regressor	5.5814	929.3758	22.3323	0.7752
gbr	Gradient Boosting Regressor	8.6257	914.0412	23.8538	0.7440
dt	Decision Tree Regressor	7.5542	1082.4813	26.5027	0.6808
ada	AdaBoost Regressor	16.6766	1305.8931	31.5821	0.5310

5. Prediction

We predicted on the test data and using our best model Xgboost and below is the performance of just first 10 **token_index**

	token_index	last_sale_price	observed	Accuracy
9467	4989	1.582349	1.589945	0.997605
1866	1537	1.346560	1.336490	0.996247
18532	9086	3.736599	3.837754	0.986645
10517	5471	1.290090	1.336490	0.982334
18810	9219	1.246154	1.185199	0.974930
14313	7158	0.686315	0.651798	0.974205
15621	7807	0.514510	0.546760	0.969612
1294	1244	1.419777	1.311984	0.960541
9439	4976	0.849789	0.783293	0.959281
9808	5153	1.388281	1.277217	0.958333

How to Approach the problem Given more time and resources

Here are few of my thoughts on how to improve the model performance:

1. **Image Analysis**: Extraction of features using pretrained CNN model from the NFT images and adding it to the features before training will enrich the model.
2. **Twitter Data**: Getting more data related to each token on twitter will influence the performance of the model.
3. **Medium gas price**(Gwei): It is observed that the price of Gwei influences the performance of different tokens at a particular time. Therefore the addition of this feature will help measures the volatility of the index and help better the model.