Kanishk Kamalvanshi
CS 242 Project

Abstract:

NFTs have been a major application to blockchain technology, and they are a growing space. Hundreds of collections are made every week. However, many of them tend to fail for many reasons. This project finds ways NFT collection within the top 100 can do to further maximize their profits. Based on the findings, we can see that the blockchain that the project is built on, the newness, and the platform the collection is published matters and can made a significant difference in the market value.

Introduction:

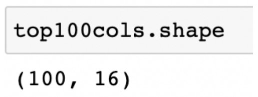
An NFT is a non-fungible token that can be in the form of an image, video, etc. that is stored in the blockchain. The NFT space has been gaining traction since the past 2 years, especially the last year, where sales were approximately $17.7 billion. Based on these numbers, it looks like NFTs are here to stay, and more players will enter this space. The dataset was taken from CryptoSlam API's that track the top 100 NFT collections. This project provides insights on the top 100 NFT collections.

Data:

The dataset was taken from CryptoSlam API's that track the top 100 NFT collections and top 100 NFT's sold in the market. The top 100 NFT collections of the month dataset was saved as the top100cols.

```
> top100cols = pd.read_csv('response_1650837751168.csv', encoding="ISO-8859-1")
```

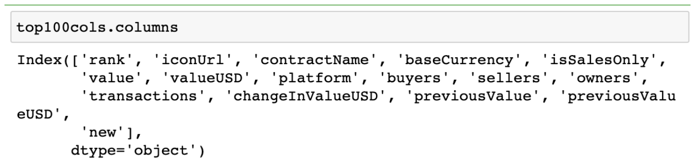
The dataset has 16 columns and 100 rows (since we are looking at top 100 NFT collections). The dataset is 18 KB.



```
top100cols.shape

(100, 16)
```

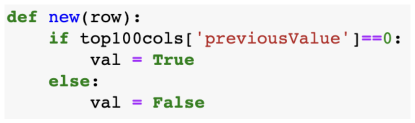
The columns are rank, iconUrl, contractName, baseCurrency, isSalesOnly, value, valueUSD, platform, buyers, sellers, owners, transactions, changeInValueUSD, previousValue, previousValueUSD. **rank** is the is rank of the NFT project for that month. **iconUrl** is the picture of the logo of the NFT collection. **contractName** is the name of the NFT project. **baseCurrency** is the cryptocurrency that the NFT collection is running on. **isSalesOnly** means if the NFT collection contract is meant for sales only and has not other property. **value** is the value of the NFT project based on the baseCurrency. **valueUSD** is the value of the NFT collection in US dollars. **platform** is the the number associated to a specific platform. buyers is the number of buyers of the NFT in the NFT Collection that month. **sellers** is the number of sellers of the NFTs in the NFT collection that month. **owners** is the number of owners of the NFT in the NFT Collection that month. **changeInValueUSD** is the **change** in the value of the NFT collection in terms of US Dollars. **previousValue** is value of the NFT project based on the baseCurrency from the previous month. **previousValueUSD** is previousValue but in terms of US Dollars.



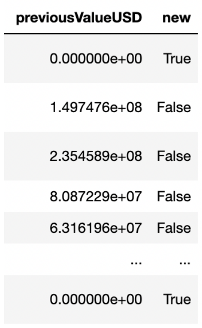
```
top100cols.columns
```

```
Index(['rank', 'iconUrl', 'contractName', 'baseCurrency', 'isSalesOnly',
       'value', 'valueUSD', 'platform', 'buyers', 'sellers', 'owners',
       'transactions', 'changeInValueUSD', 'previousValue', 'previousValu
eUSD',
       'new'],
      dtype='object')
```

Preprocessing with the dataset was also done to find better insights and create a better machine learning model. For example, some rows had previousValueUSD equalling to 0, meaning that the collection was new this month, so the variable new was added to the columns of the dataset which stores a boolean value depending on previousValueUSD.

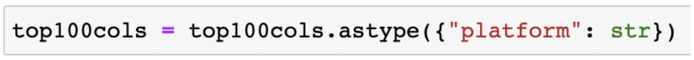


```
def new(row):
    if top100cols['previousValue']==0:
        val = True
    else:
        val = False
```



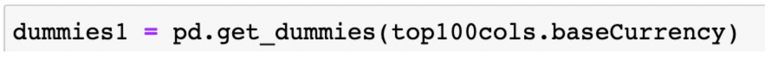
previousValueUSD	new
0.000000e+00	True
1.497476e+08	False
2.354589e+08	False
8.087229e+07	False
6.316196e+07	False
...	...
0.000000e+00	True

Also, the platform column was converted into str for analysis to create boxplots for each platform.

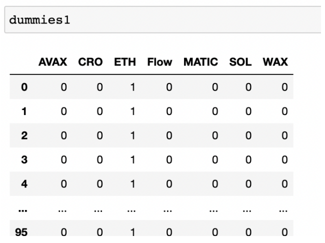


```
top100cols = top100cols.astype({"platform": str})
```

For the model to read categorical variables within the dataset, one-hot encoding was used to create dummy variables from the categorical variables. First, we create dummies1 which creates dummy variables from the baseCurrency variable. For example, if a project is running on Ethereum (ETH), we store 1 for the ETH variable and store 0 for the rest of the currencies such as AVAX, CRO, Flow, MATIC, SOL, WAX.



```
dummies1 = pd.get_dummies(top100cols.baseCurrency)
```



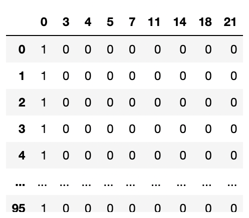
```
dummies1
```

	AVAX	CRO	ETH	Flow	MATIC	SOL	WAX
0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...
95	0	0	1	0	0	0	0

We then create dummies2 which creates dummy variables from the platform variable variable based on the number of the platform variable.

```
dummies2 = pd.get_dummies(top100cols.platform)
```

```
dummies2
```



	0	3	4	5	7	11	14	18	21
0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
95	1	0	0	0	0	0	0	0	0

Lastly, boolean variables such as isSalesOnly and new were changed from True and False to 1 and 0.

```
final['isSalesOnly'] = (final['isSalesOnly'] == True).astype(int)
```

```
final['new'] = (final['new'] == True).astype(int)
```

Problems and Methods:

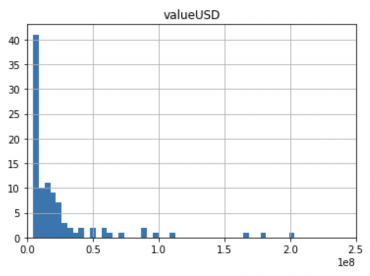
We want to explore ways to maximize profits for NFT projects. First, we want to find the distribution in valueUSD based on different factors such as the coin the NFT collection has been built on top of, the newness of the NFT Collection (if it dropped since the past 30 days), and the platform the NFT collection was dropped on. Then, we create models in order to predict the best ways to maximize valueUSD. In addition, to find the driving factors in the dataset that maximize the valueUSD, anova tests will be performed to find significant variables within the dataset.

Results and Discussion:

First, we look at the distribution of the valueUSD within the dataset. Based on the histogram, the distribution in valueUSD is heavily skewed to the right. The minimum value is approximately 4.66e+06, and the max is approximately 4.36e+08.

```
top100cols.hist(column = 'valueUSD', bins = 100)
plt.xlim([0,250000000])
```

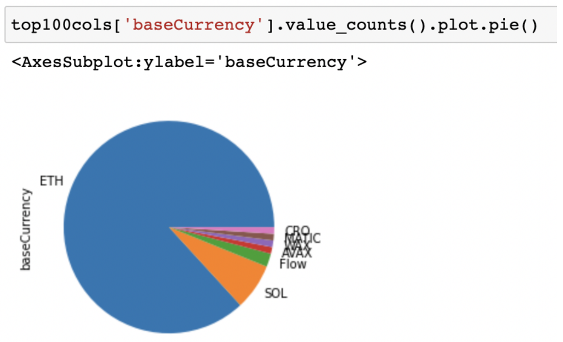
```
(0.0, 250000000.0)
```



```
top100cols['valueUSD'].describe()
```

```
count    1.000000e+02
mean     2.869996e+07
std      5.402230e+07
min      4.664112e+06
25%      6.465920e+06
50%      1.310916e+07
75%      2.238525e+07
max      4.368491e+08
Name: valueUSD, dtype: float64
```

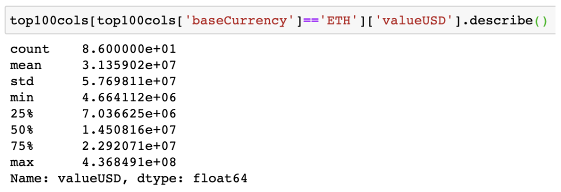
Now, we will look at the dataset with respect to the baseCurrency. Based on this pie chart, it appears that a majority of the projects are Ethereum (ETH) based. The coin with the second-most projects is Solana (SOL), and the runner ups are Flow, AVAX, MATIC, WAX, and CRO.



```
top100cols['baseCurrency'].value_counts().plot.pie()
```

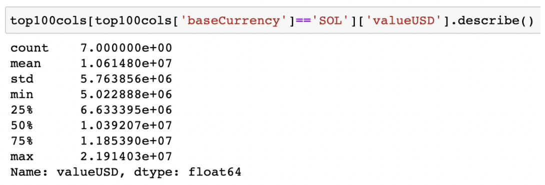
```
<AxesSubplot:ylabel='baseCurrency'>
```

We can also see the distribution of valueUSD based on baseCurrency. 86 projects in the top 100 collections are running on ETH. It is also skewed to the right as ETH is the majority of the collections. With SOL, since the mean is higher with median, the distribution of valueUSD is also skewed to the right. Since Flow only has 2 rows, the distribution for valueUSD is normal. Other projects that are using other coins have only one row, so the describe function wouldn't work properly for those subsets.



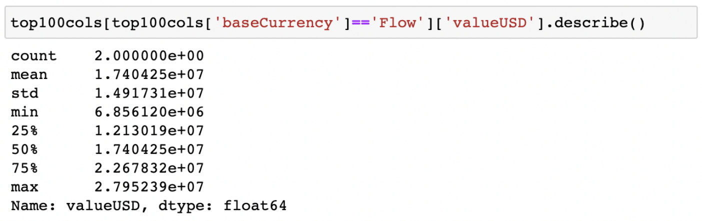
```
top100cols[top100cols['baseCurrency']=='ETH']['valueUSD'].describe()
```

```
count    8.600000e+01
mean     3.135902e+07
std      5.769811e+07
min      4.664112e+06
25%      7.036625e+06
50%      1.450816e+07
75%      2.292071e+07
max      4.368491e+08
Name: valueUSD, dtype: float64
```



```
top100cols[top100cols['baseCurrency']=='SOL']['valueUSD'].describe()
```

```
count    7.000000e+00
mean     1.061480e+07
std      5.763856e+06
min      5.022888e+06
25%      6.633395e+06
50%      1.039207e+07
75%      1.185390e+07
max      2.191403e+07
Name: valueUSD, dtype: float64
```

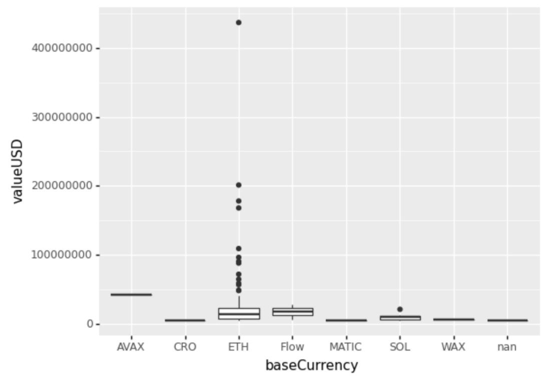


```
top100cols[top100cols['baseCurrency']=='Flow']['valueUSD'].describe()
```

```
count    2.000000e+00
mean     1.740425e+07
std      1.491731e+07
min      6.856120e+06
25%      1.213019e+07
50%      1.740425e+07
75%      2.267832e+07
max      2.795239e+07
Name: valueUSD, dtype: float64
```

Based on the boxplots of all coins, it looks like the project running on Avalanche (AVAX) is outperforming other projects that are running on other coins based on their interquartile range. It also looks like ETH projects have many outliers.

```
boxplot = ggplot(top100cols,aes('baseCurrency','valueUSD')) + geom_boxplot(
boxplot
```



Now, we take a look at how the newness of projects can affect its valueUSD. Based on the description and the boxplots of the distribution of valueUSD based on newness of the project, it appears that the new projects have a greater mean and median than that of older projects. However, projects that are old also appear to have far more outliers compared to that of new. In addition, there is an NA value meaning that the coin has not been recorded properly.

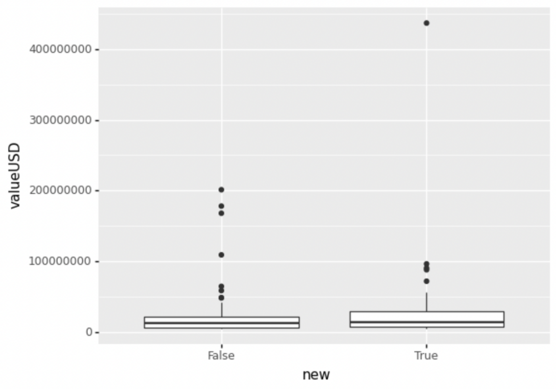
```
top100cols[top100cols['new']==True]['valueUSD'].describe()
```

```
count    3.200000e+01
mean     3.743368e+07
std      7.762092e+07
min      4.664112e+06
25%      7.362158e+06
50%      1.406307e+07
75%      2.968038e+07
max      4.368491e+08
Name: valueUSD, dtype: float64
```

```
top100cols[top100cols['new']==False]['valueUSD'].describe()
```

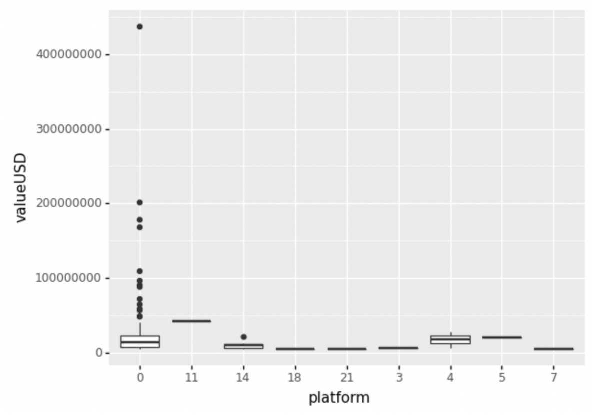
```
count    6.800000e+01
mean     2.458998e+07
std      3.835377e+07
min      4.711450e+06
25%      6.390523e+06
50%      1.280732e+07
75%      2.193803e+07
max      2.012193e+08
Name: valueUSD, dtype: float64
```

```
boxplot = ggplot(top100cols,aes('new','valueUSD')) + geom_boxplot()
boxplot
```



The platform of where the NFT is being sold is very important, and this can be one of the driving factors for NFT sales. The coin that is being used can also be a determinant when it comes to platform. For example, the biggest platform as of today is OpenSea, which accepts mainly Ethereum (ETH), and more recently (3 weeks ago) Solana (SOL). Here, we can look at how the platform can affect the valueUSD for a project. It appears that platform 0 has many outliers, meaning that the most popular NFT collections are hosted in platform 0. Platform 11's NFT project seems to be outnumbering that of many others based on their interquartile ranges. Also, platform 4 has a few NFT collections that are perhaps being used through another coin as platform 0 is only compatible with ETH and SOL.

```
boxplot = ggplot(top100cols,aes('platform','valueUSD')) + geom_boxplot()
boxplot
```



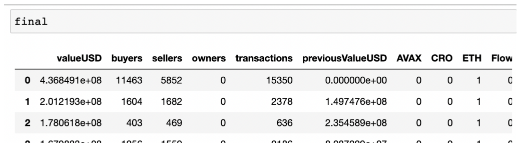
For the model, a merged dataframe is needed to merge the original dataframe, and the dummy variables dataframe (dummies1 and dummies2).

```
merged = pd.concat([top100cols,dummies1, dummies2],axis = 'columns')
```

Then, a final dataset was created where specific columns such as rank, iconUrl, contractName, value, previousValue, changeInValueUSD were dropped. In addition, baseCurrency and platform were dropped since they were converted into dummy variables through one hot encoding and merged.

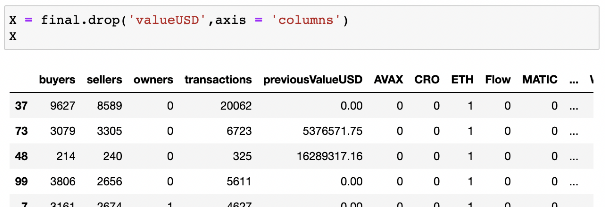
```
> final =
merged.drop(['baseCurrency','platform','rank','iconUrl','contractName','value',
'previousValue','changeInValueUSD'],axis = 'columns')
```

Before going to the models, we need to define the inputs (X) and outputs (Y) and the train/test split.



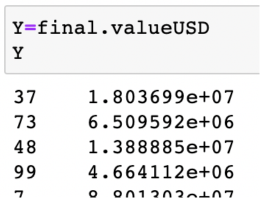
```
final
```

	valueUSD	buyers	sellers	owners	transactions	previousValueUSD	AVAX	CRO	ETH	Flow
0	4.368491e+08	11463	5852	0	15350	0.000000e+00	0	0	1	0
1	2.012193e+08	1604	1682	0	2378	1.497476e+08	0	0	1	0
2	1.780618e+08	403	469	0	636	2.354589e+08	0	0	1	0



```
X = final.drop('valueUSD',axis = 'columns')
X
```

	buyers	sellers	owners	transactions	previousValueUSD	AVAX	CRO	ETH	Flow	MATIC	...
37	9627	8589	0	20062	0.00	0	0	1	0	0	...
73	3079	3305	0	6723	5376571.75	0	0	1	0	0	...
48	214	240	0	325	16289317.16	0	0	1	0	0	...
99	3806	2656	0	5611	0.00	0	0	1	0	0	...



```
Y=final.valueUSD
Y
```

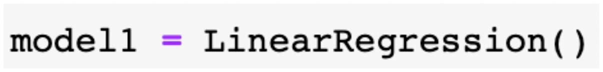
```
37     1.803699e+07
73     6.509592e+06
48     1.388885e+07
99     4.664112e+06
```

Kanishk Kamalvanshi
CS 242 Project

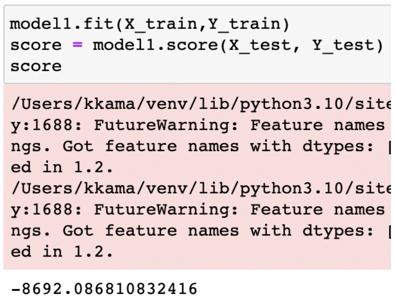
Train/Test Split:

```
>X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.
3,random_state=101)
```

First, the Linear Regression Model was used.



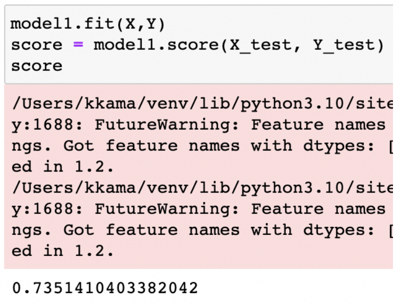
```
model1 = LinearRegression()
```



```
model1.fit(X_train,Y_train)
score = model1.score(X_test, Y_test)
score
```

```
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
```

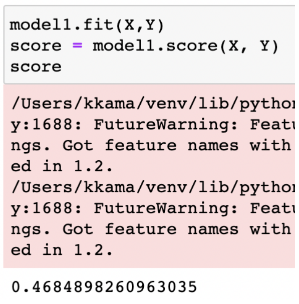
```
-8692.086810832416
```



```
model1.fit(X,Y)
score = model1.score(X_test, Y_test)
score
```

```
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
```

```
0.7351410403382042
```

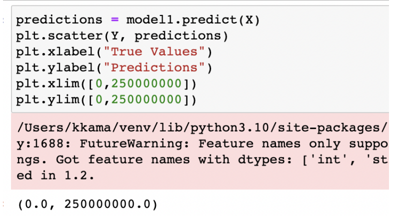


```
model1.fit(X,Y)
score = model1.score(X, Y)
score
```

```
/Users/kkama/venv/lib/pytho
y:1688: FutureWarning: Featu
ngs. Got feature names with
ed in 1.2.
/Users/kkama/venv/lib/pytho
y:1688: FutureWarning: Featu
ngs. Got feature names with
ed in 1.2.
```

```
0.4684898260963035
```

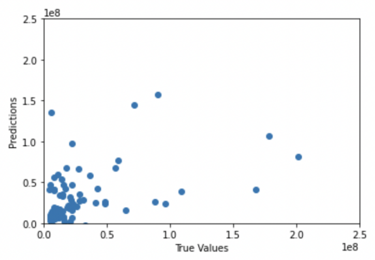
With train/test split, the model accuracy is -869200%; however, when evaluating only test sets by fitting entire model, accuracy is 73.51%.



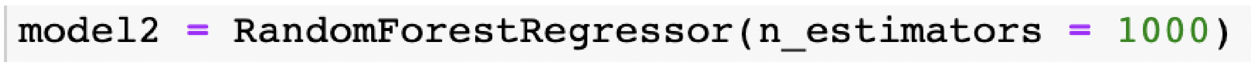
```
predictions = model1.predict(X)
plt.scatter(Y, predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.xlim([0,250000000])
plt.ylim([0,250000000])
```

```
/Users/kkama/venv/lib/python3.10/site-packages/
y:1688: FutureWarning: Feature names only suppo
ngs. Got feature names with dtypes: ['int', 'st
ed in 1.2.
```

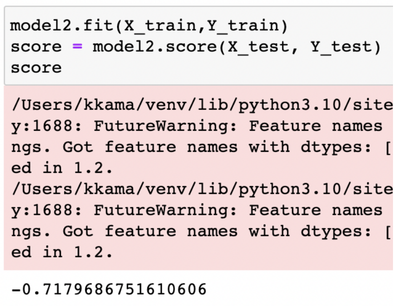
```
(0.0, 250000000.0)
```



Based on first model.



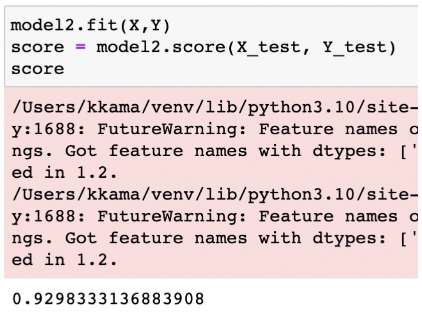
```
model2 = RandomForestRegressor(n_estimators = 1000)
```



```
model2.fit(X_train,Y_train)
score = model2.score(X_test, Y_test)
score
```

```
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
/Users/kkama/venv/lib/python3.10/site
y:1688: FutureWarning: Feature names
ngs. Got feature names with dtypes: [
ed in 1.2.
```

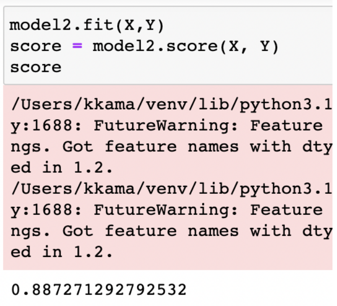
```
-0.7179686751610606
```



```
model2.fit(X,Y)
score = model2.score(X_test, Y_test)
score
```

```
/Users/kkama/venv/lib/python3.10/site-
y:1688: FutureWarning: Feature names o
ngs. Got feature names with dtypes: ['
ed in 1.2.
/Users/kkama/venv/lib/python3.10/site-
y:1688: FutureWarning: Feature names o
ngs. Got feature names with dtypes: ['
ed in 1.2.
```

```
0.9298333136883908
```

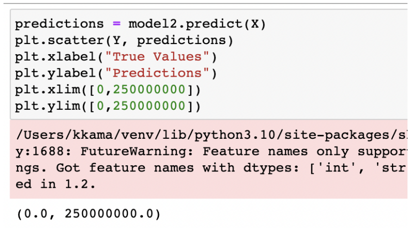


```
model2.fit(X,Y)
score = model2.score(X, Y)
score
```

```
/Users/kkama/venv/lib/python3.1
y:1688: FutureWarning: Feature
ngs. Got feature names with dty
ed in 1.2.
/Users/kkama/venv/lib/python3.1
y:1688: FutureWarning: Feature
ngs. Got feature names with dty
ed in 1.2.
```

```
0.887271292792532
```

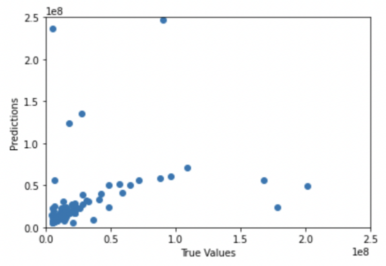
With train/test split, the model accuracy is -71.79%; however, when evaluating only test sets by fitting entire model, accuracy is 92.98%.



```
predictions = model2.predict(X)
plt.scatter(Y, predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.xlim([0,250000000])
plt.ylim([0,250000000])
```

```
/Users/kkama/venv/lib/python3.10/site-packages/s
y:1688: FutureWarning: Feature names only suppor
ngs. Got feature names with dtypes: ['int', 'str
ed in 1.2.
```

```
(0.0, 250000000.0)
```

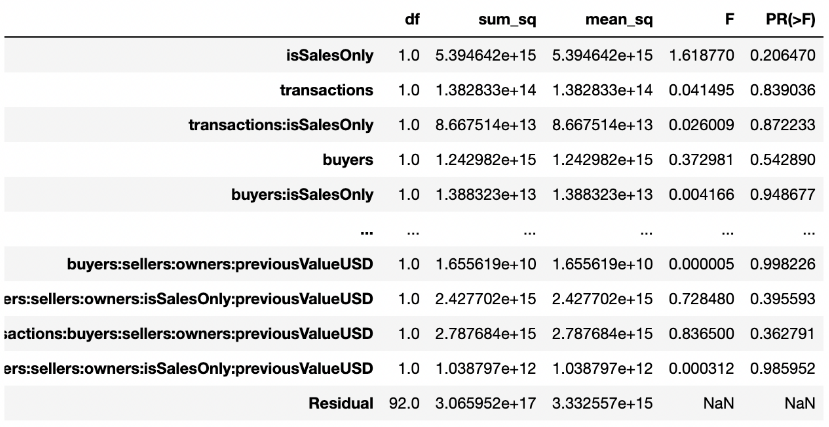


This is based on the first model. The true values vs predictions appear to have some correlation minus the outliers.

To find significant variables and interactions within the dataset, we conduct ANOVA tests to find what are the driving factors and interactions that lead to a greater valueUSD. We also couldn't use categorical variables such as platform and baseCurrency the model wasn't allowing them due to memory.

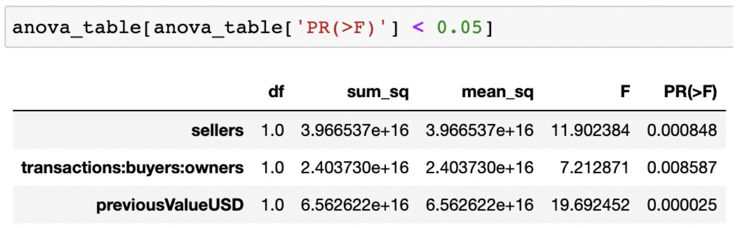
```
>model =
ols('valueUSD~transactions*buyers*sellers*owners*isSalesOnly*pre
viousValueUSD', data=top100cols).fit()
anova_table = sm.stats.anova_lm(model)
```

To find this, we create a model in which transactions, buyers, sellers, owners, and isSalesOnly are interacting with each other.



	df	sum_sq	mean_sq	F	PR(>F)
isSalesOnly	1.0	5.394642e+15	5.394642e+15	1.618770	0.206470
transactions	1.0	1.382833e+14	1.382833e+14	0.041495	0.839036
transactions:isSalesOnly	1.0	8.667514e+13	8.667514e+13	0.026009	0.872233
buyers	1.0	1.242982e+15	1.242982e+15	0.372981	0.542890
buyers:isSalesOnly	1.0	1.388323e+13	1.388323e+13	0.004166	0.948677
...	...	...	...	...	...
buyers:sellers:owners:previousValueUSD	1.0	1.655619e+10	1.655619e+10	0.000005	0.998226
ers:sellers:owners:isSalesOnly:previousValueUSD	1.0	2.427702e+15	2.427702e+15	0.728480	0.395593
;actions:buyers:sellers:owners:previousValueUSD	1.0	2.787684e+15	2.787684e+15	0.836500	0.362791
ers:sellers:owners:isSalesOnly:previousValueUSD	1.0	1.038797e+12	1.038797e+12	0.000312	0.985952
Residual	92.0	3.065952e+17	3.332557e+15	NaN	NaN

The anova table shows the p-values of. Based on the p-values, our null hypothesis would be that there is insufficient evidence that there is any interaction within said variables, and the alternative hypothesis is that there is significant interaction within said variables. In this case, we will assume that alpha = 0.05. In the anova table, the p-values are checked if they are less than alpha, and are filtered out from the table.



```
anova_table[anova_table['PR(>F)'] < 0.05]
```

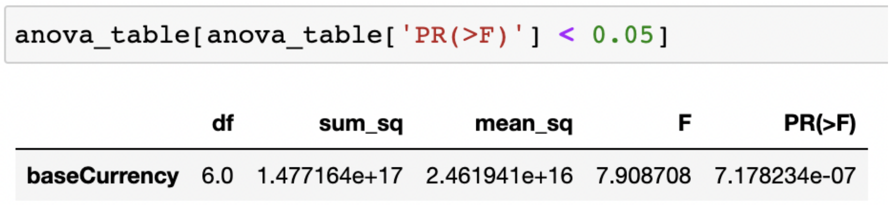
	df	sum_sq	mean_sq	F	PR(>F)
sellers	1.0	3.966537e+16	3.966537e+16	11.902384	0.000848
transactions:buyers:owners	1.0	2.403730e+16	2.403730e+16	7.212871	0.008587
previousValueUSD	1.0	6.562622e+16	6.562622e+16	19.692452	0.000025

Based on this filtered table, we see that sellers and previousValueUSD have p-values that are less than alpha. Hence, we reject the null hypothesis and can conclude that there is sufficient evidence that both sellers and previousUSD are significant variables within the table. Also, since the p-value between the interaction between transactions, buyers, and owners has a p-value less than alpha, we can reject the null hypothesis and conclude that this interaction was significant to the model.

This model can still be bias since the categorical variables such as baseCurrency and platform are not used for the model.

Another anova test was done only with platform and baseCurrency due to the lack of memory the model can handle.

```
> model = ols('valueUSD~platform*baseCurrency',
data=top100cols).fit()
anova_table = sm.stats.anova_lm(model)
```



```
anova_table[anova_table['PR(>F)'] < 0.05]
```

	df	sum_sq	mean_sq	F	PR(>F)
baseCurrency	6.0	1.477164e+17	2.461941e+16	7.908708	7.178234e-07

Since the p-value between the variable baseCurrency has a p-value less than alpha, we can reject the null hypothesis and conclude that this interaction was significant to the model.

Conclusion:

Based on the analysis and the plots, that baseCurrency, newness, and the platform used to trade these NFTs are relevant factors as they make a significant difference to the valueUSD. Looking at the linear and random forest regression forest models, it is extremely evident that models cannot be built on top of them. This can be perhaps due to the the size of the data (100 rows), it may not be possible to build a reliable machine learning model due to the current lack of availability of large NFT datasets. For further research, it would be better to use a larger dataset more than 1000 rows. Lastly, the anova test was used to find more driving factors that can maximize valueUSD. Main drivers were sellers, previousValueUSD, baseCurrency, and the interaction between transactions, buyers, and owners.