Filling missing value first.

Data exploration:

1. histogram: find sparsed data, look at discrete & categorical data distribution

2. correlation map: find multicollinearity

3. clustering (kmeans, DBSCAN, hierarchical)

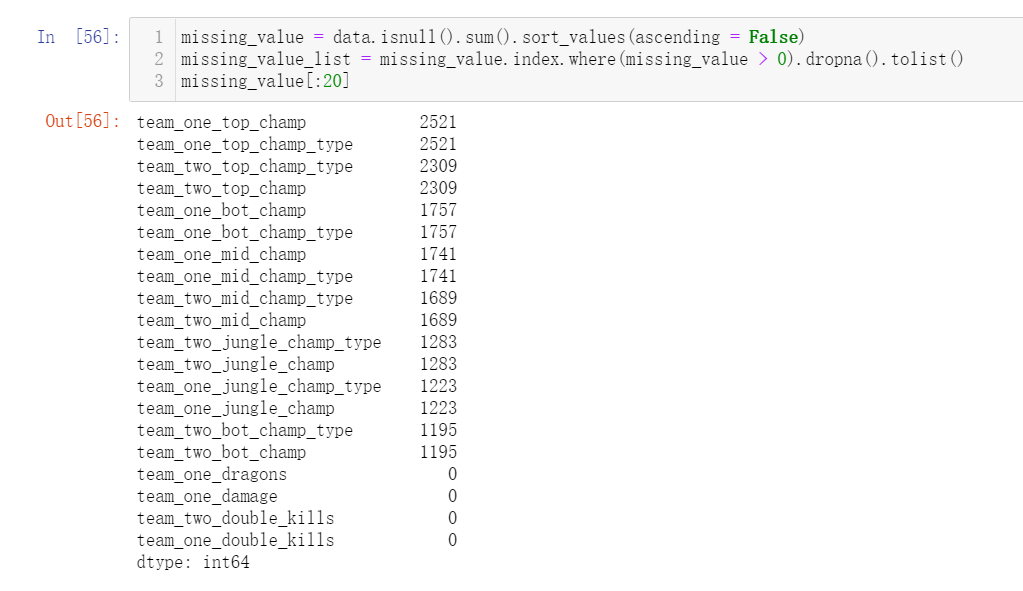
4. feature selection (by models, score methods,forward feature selection)

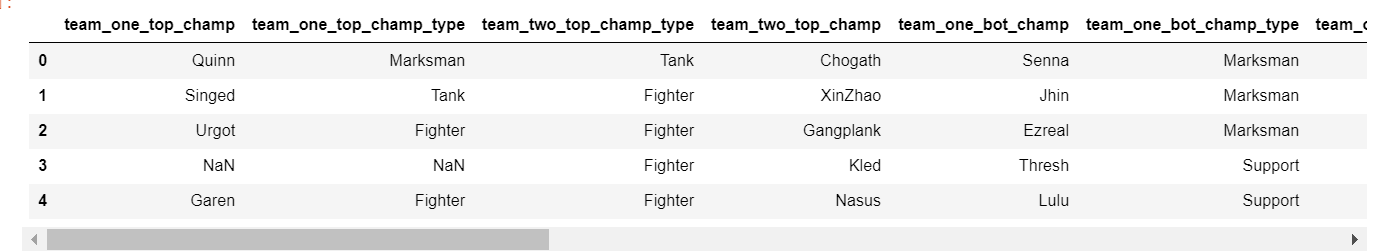
5. scaling continous variable to 0-1

Data preprocessing:

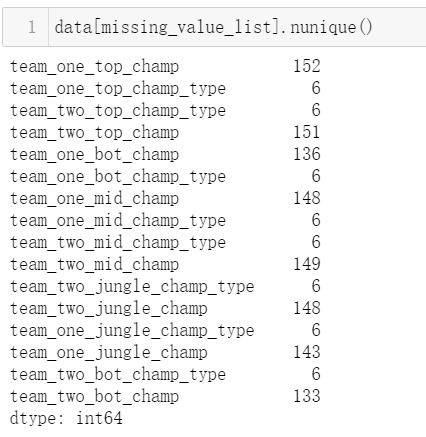
1. distinguish continuous numeric, discrete numeric, categorical columns
2. filling missing value by IterativeImputer, with method decision tree. (if there is numerical missing value, use other method)
3. onehog encoder for unique value <=20, label encoder for 20-50, grouping when > 50
4. visualize columns with unique values > 50, then choose binning methods
5. normalize numerical data with big values

Filling missing value:





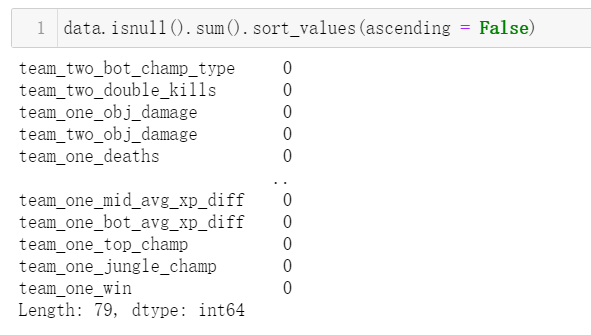
By look at the missing columns, they are all categorial words with names.



Check numbers of unique values make sure that they are not random names.

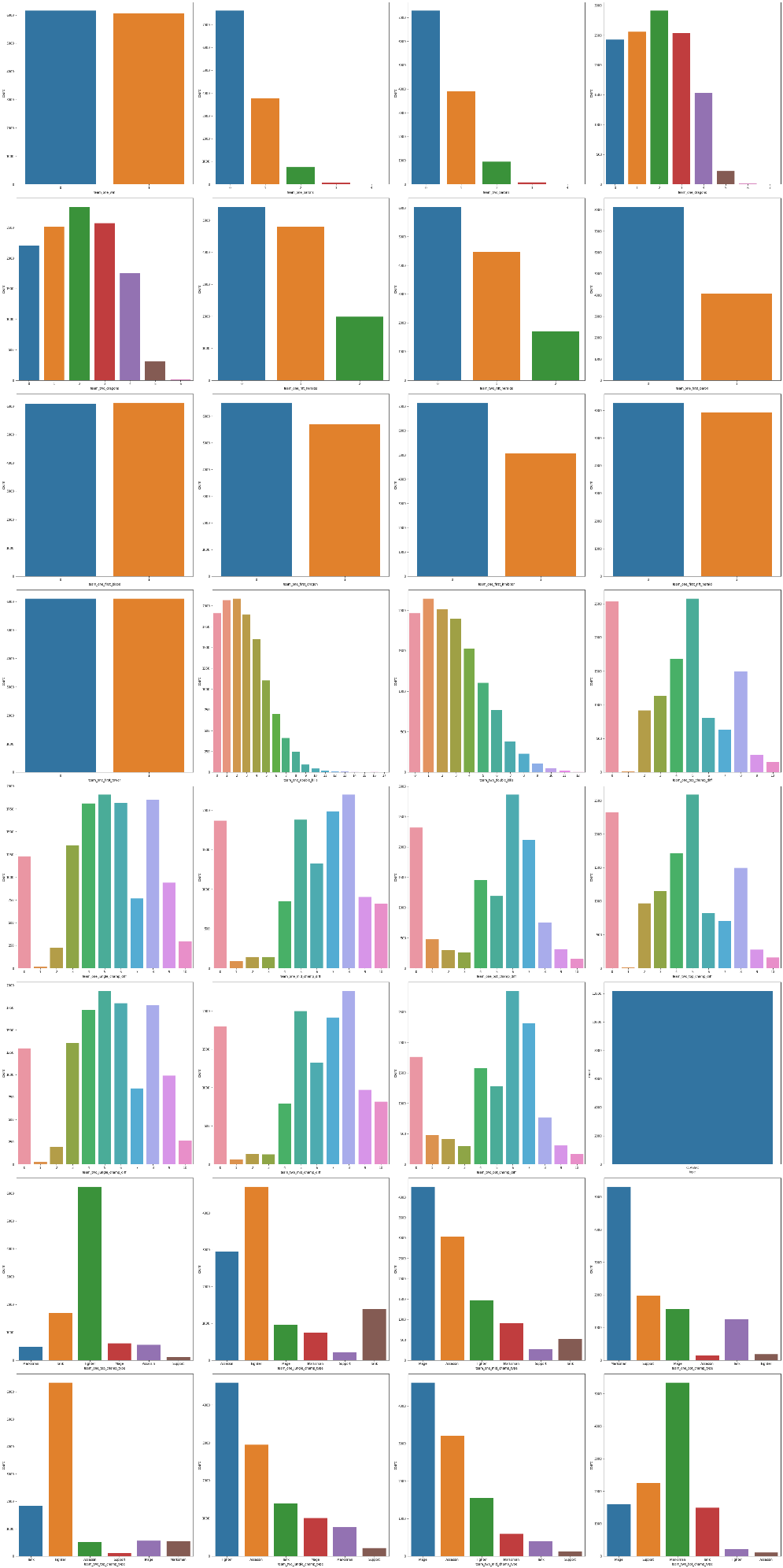
Since each game are independent than others, we cannot fill values with nearby values. I want to use IterativeImputer, a strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion.

I choose estimator = DecisionTreeClassifier(), initial\_strategy = "most\_frequent"

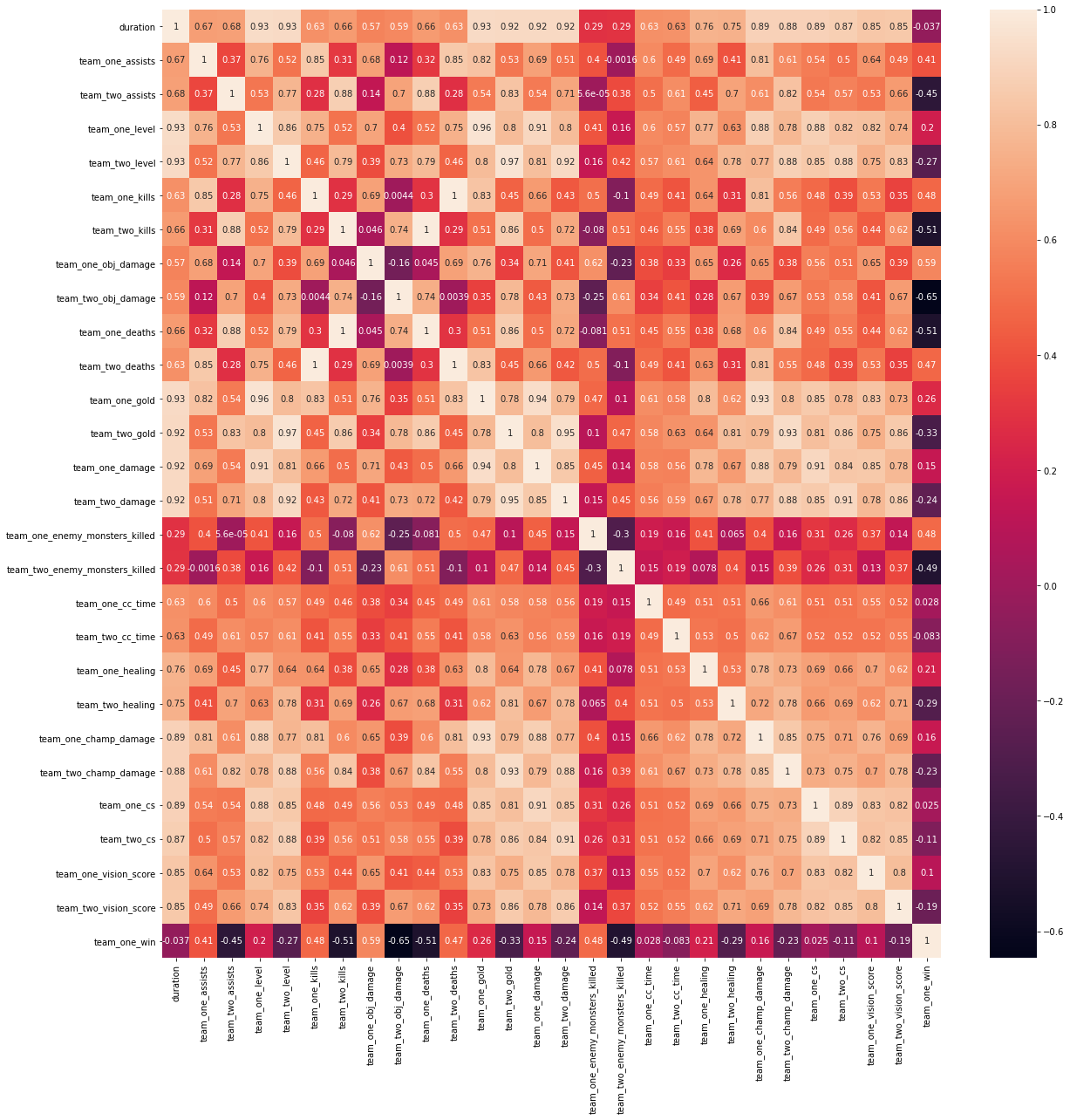


Now finishes the imputation.

Data exploration: histogram (next page)

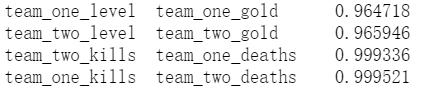
Looks like type is sperased data, delete it.

1. correlation map: find multicollinearity



Since there are too many attributes, use python to find pairs of correlations > 0.95.

Multicollinearity refers to a situation in which more than two explanatory variables in a multiple regression model are highly correlation.



Here is the result of pairs. Delete one of the feature for each pair to reduce multicollinearity.

1. clustering (kmeans, DBSCAN, hierarchical)

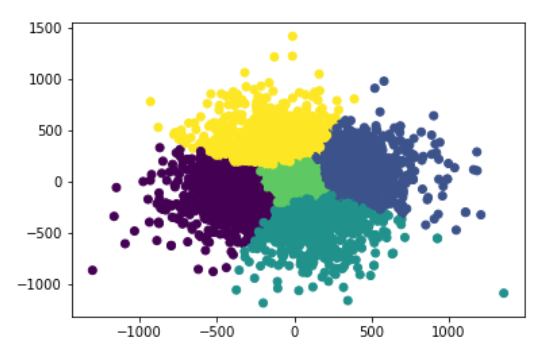
For the visualization purpose, dimensionally reduction to 2 columns by PCA. Randomly sample some of the data points.

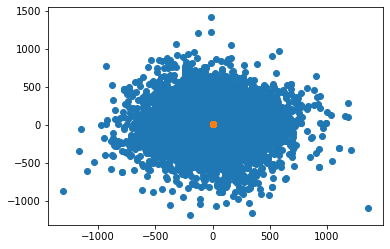
Kmeans



It has a clear boundary.

(one possible way is minimum intra cluster distance and maximum inter cluster distance.)





DBSCAN: not a good result, exclude this.

(also apply silhouette\_score to the dataframe.

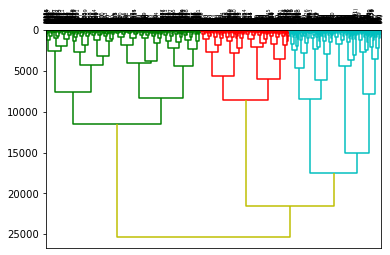
Silhouette Score = (b-a)/max(a,b) where a= average intra-cluster distance i.e the average distance between each point within a cluster. b= average inter-cluster distance i.e the average distance between all clusters.

1: Means clusters are well apart from each other and clearly distinguished.

0: Means clusters are indifferent, or we can say that the distance between clusters is not significant. -1: Means clusters are assigned in the wrong way.

Our score is -0.22, which is not good.)

Hierarchical clustering:



1. feature selection (by models, score methods, forward feature selection)

I’m using python sklearn select from model with decision tree, random forest, ada boosting and ridge classifier, and select k best by f\_classif score function.

Then, combine the selected columns and apply min max scaler. I’m consider to use step-wise feature selection, but it has high computational cost, not fit with our dataset.

1. scaling continous variable to 0-1

By using min-max scaler

Data preprocessing:

1. distinguish continuous numeric, discrete numeric, categorical columns

We have string type, numerical type and float type data. Numerical type data might has little unique attribute (e.g columns with 0 and 1) So, for discrete numeric variables, when unique values > 50, consider as continous variable. Such as duration is the total time for the game, it is numeric type but should be consider as continuous numeric variable.

There are some categorical types of data, such as jungle type, champion type for the team. (mage, Assassin) By looking at numbers of unique values for categorial types data, make sure no columns with random names (people’s name, strange words, …)

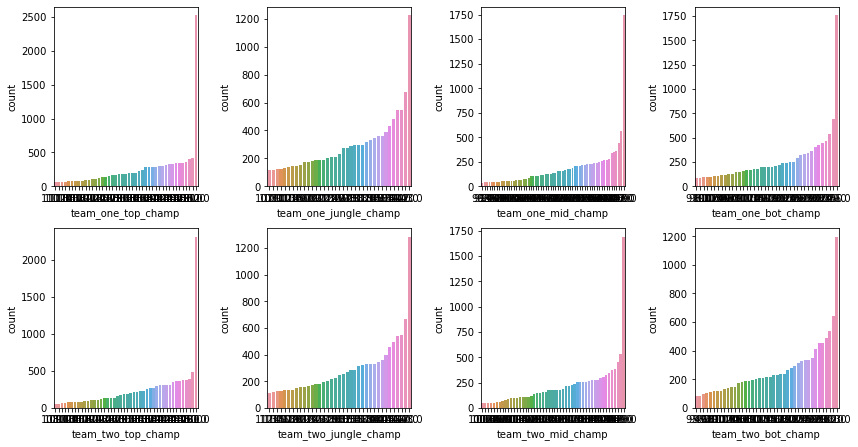
1. onehog encoder for unique value <=20, label encoder for 20-50, grouping when > 50

This is because onehog works well with columns for less unique values.

Use label encoder because onehog will generate too many columns when unique value > 20. Sometimes onehog should works better since it works well for not ordinal data. (without ranking) But there is no column has unique value between 20-50.

Use grouping since the column has too many unique values. This time I choose to group by frequency.

1. visualize columns with unique values > 50, then choose binning methods



I choose to group with 7 groups, then use onehog encoder again convert to 0s and 1s.

1. normalize numerical data with big values

The normalizer don’t works well for our datasets. But I uses minmax scaler to scaling continuous data to 0-1.