Tabular Data Science - Final Course Project

Naor Samuelov (ID. 307939447),Imry Uzan ID. 209005909)

Submitted as final project report for the TDS course, BIU,
Semester A, 2022

1 Abstract

The problem we are trying to solve is converting a scatter plot that is difficult to read and understand to another graph based on bins division of the dots. For the solution, we implemented a K-Means based algorithm that try to show the tendency of the graph using "smart" division to bins. During our experiment, we compered our algorithm results to another function we saw in class and we came to the conclusion that almost in all the tests we made, our algorithm showed impressive results.

2 Problem description

In this project we've focused on converting a scatter plot into another graph based on bins division of the dots. Until now there wasn't an effective and automatic way to split the graph's data into bins, the commonly used method for bins division was splitting the dots to equal bins. We notice this issue on one of the practical examples in class and that's the reason we chose to focus on this problem in our project.

3 Solution overview

In our project we decided to implement algorithm that's based on K-means, it gets the following parameters: DataFrame, X axis (String) , Y axis (String) and the numbers of bins required by the user. Our algorithm (will be called the main algorithm) is based on running our base algorithm multiple times. We will start with the description of our base algorithm:

As the first step, the algorithm runs the K-means algorithm for the X axis only (input it accepts from the user), while k = the numbers of bins required by the user. After the K-means we accept general division to bins that will be used as a base to our final division. Afterwards, we go over each bin and check the impact of removing the dots located at more than 1.5 standard deviation from the mean on the algorithm performance (if demanded by the main algorithm). Next, we go over each bin (by order) from the bin with the lowest X's average to the bin with the highest X's average. Every bin (except from the last one) we split into 3 thirds (equal). From the main algorithm we get a parameter called third, for convenience we will call it Z. Now, we go over the first third and check if the dots amount that's on the right or left side of the third divided by the total amounts of dots in the third is lower then Z, if it does, we change the right limit of the bin to be the value of the first third. Otherwise, we check if the dots amount that's on the right side of the second third divided by the total amounts of dots in the third is lower then Z, if it does, we change the right limit of the bin to be the value of the second third. After we went over all the bin at the way we described above we finished the basic bins creation process.

Next, we will describe our main algorithm:
It gets the following parameters: the number of bins required, DataFrame, X axis and Y axis. The algorithm splits the DataFrame into train and validation sets (train=0.9 , validation=0.1) and runs the basic algorithm 10 times, such that he changes the Z value (third parameter) and decides whether to use the standard deviation. Throughout the whole process the algorithm keeps the mean distance of the test points from the generated graph (for each execution of the basic algorithm) and prints the graph with the lowest mean distances.

4 Experimental evaluation

We compared our algorithm to the technique we saw in class:
fig, ax = plt.subplots(figsize=(10,4))
dtf.groupby(pd.cut(dtf['YearBuilt'],6))['SalePrice'].mean().plot(kind='line',ax=ax)

From each data set we chose 2 columns and did 3 experiments with 6 bins, 8 bins and 10 bins. In the graphs below we will show for each bin number our result and the results of the algorithm we saw in class. For each graph we measured the mean distances between the test points and our graph. At most we saw that our algorithm reached to better results than the algorithm we saw in class (since our algorithm is based on k-means, which initialize it's centroieds randomly the results can be a bit different each time. For most of the data sets we checked, our algorithm achieved better results in 100% of the time except on the "houses" data set that achieved better results only at 60% percent of the time.

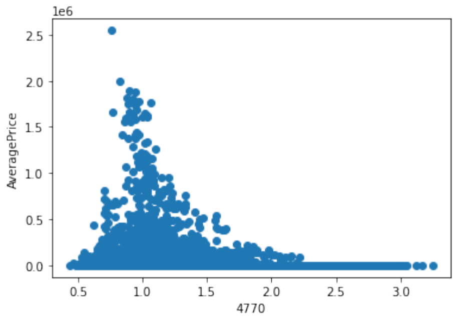
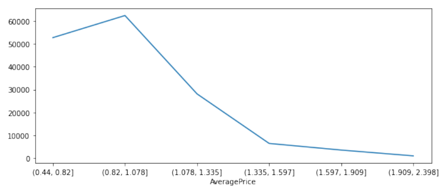
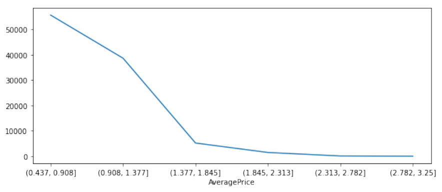


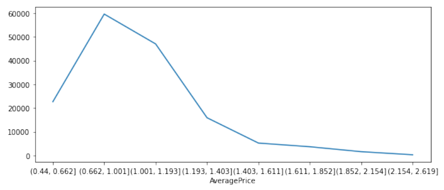
Figure 1: Avocado prices data set



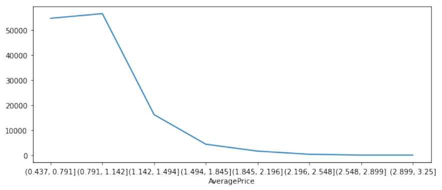
Our algorithm, 6 bins,
average distance: 0.2545



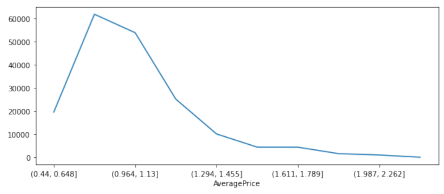
Class algorithm, 6 bins,
average distance: 0.3854



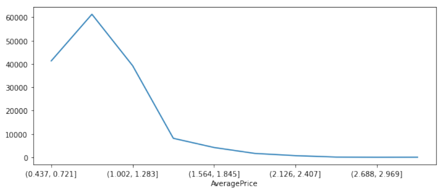
Our algorithm, 8 bins,
average distance: 0.2347



Class algorithm, 8 bins,
average distance: 0.2619



Our algorithm, 10 bins,
average distance: 0.187



Class algorithm, 10 bins,
average distance: 0.2348

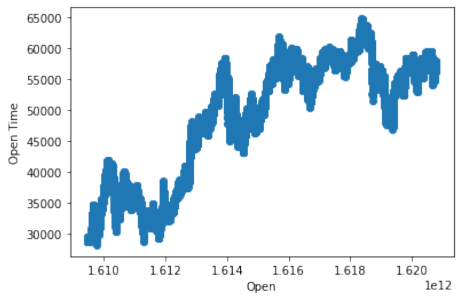
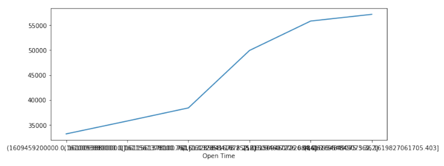
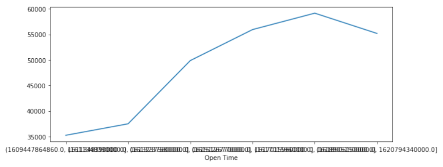


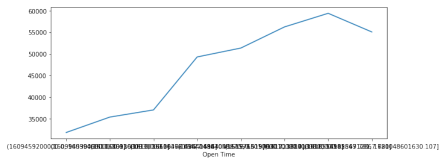
Figure 2: Bitcoin prices data set



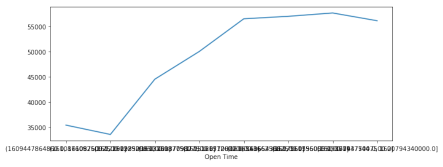
Our algorithm, 6 bins,
average distance: 1636.9496



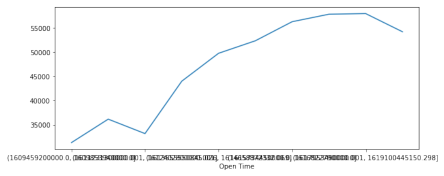
Class algorithm, 6 bins,
average distance: 2794.7228



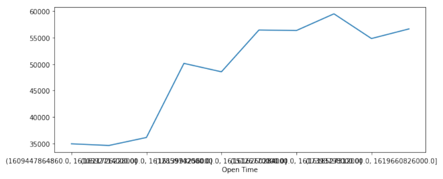
Our algorithm, 8 bins,
average distance: 1397.4262



Class algorithm, 8 bins,
average distance: 2155.9827



Our algorithm, 10 bins,
average distance: 951.5102



Class algorithm, 10 bins,
average distance: 1089.7179

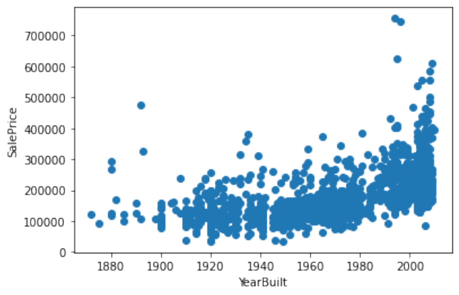
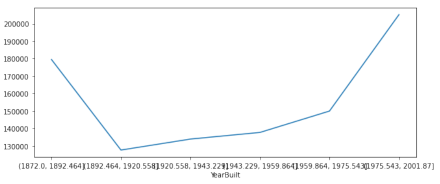
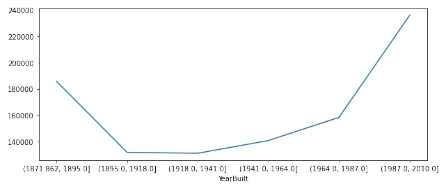


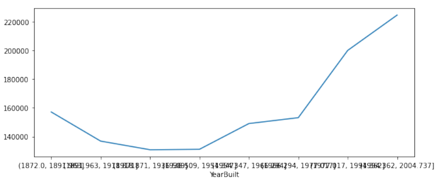
Figure 3: Houses data set



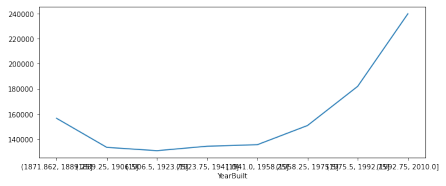
Our algorithm, 6 bins, average distance: 11.975



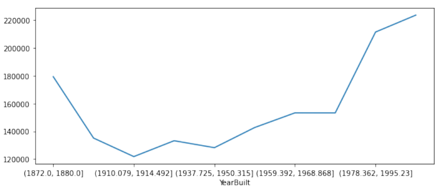
Class algorithm, 6 bins, average distance: 14.86



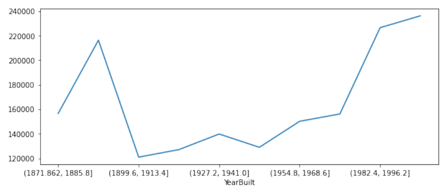
Our algorithm, 8 bins, average distance: 10.645



Class algorithm, 8 bins, average distance: 12.660



Our algorithm, 10 bins, average distance: 7.602



Class algorithm, 10 bins, average distance: 9.952

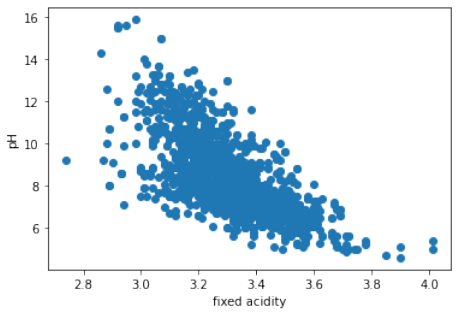
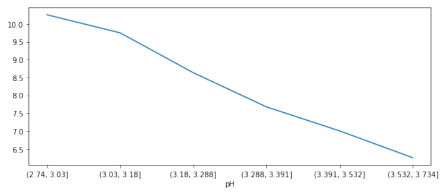
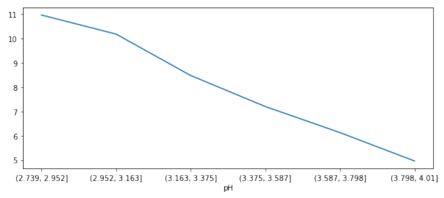


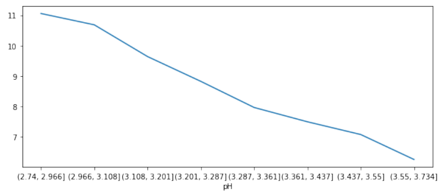
Figure 4: Wine quality data set



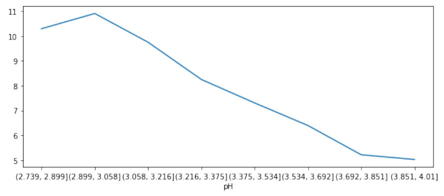
Our algorithm, 6 bins,
average distance: 0.06994



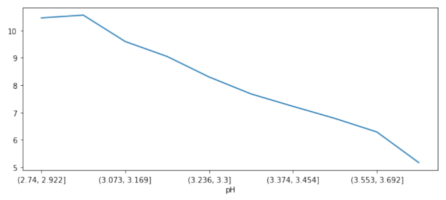
Class algorithm, 6 bins,
average distance: 0.113



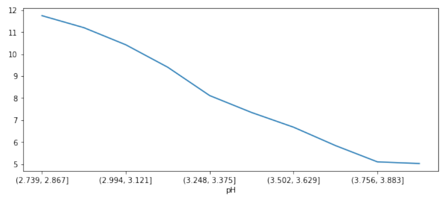
Our algorithm, 8 bins,
average distance: 0.06359



Class algorithm, 8 bins,
average distance: 0.1041



Our algorithm, 10 bins,
average distance: 0.04572



Class algorithm, 10 bins,
average distance: 0.0608

5 Related work

We saw in class a function that converts a scatter plot to a graph based on bins division of the dots and we thought that most of the time this function doesn't show the tendency of the graph properly. In another course (Cognitive Models) we study about the connection between human brain and computer science and we tried to mock the way human would do this mission, taking into account what we studied in our class, in our algorithm.

6 Conclusion

We created a new algorithm that converts a scatter plot to a graph based on bins division of the dots. Our algorithm achieves good results compering to the algorithm we saw in class but it has 2 drawbacks:
1. It based on a randomise algorithm so the graph it generates for the same arguments might be different a bit every time.
2. The run time of our algorithm is longer then the run time of the algorithm we saw in class. For most of the data sets we checked, it wasn't noticeable but for very large data sets it can take a while (the longest run-time we had was around 5 minuets in google colab and even longer when we used jupyter notebook).

7 Code

Link to git repository: `https://github.com/ImryUzan/TDS_Final_Project_2022`
Link to colab notebook example: `https://colab.research.google.com/drive/1dHtmRO11mXB6-kkZ8UxFXv4ZE8qWpZ1f?usp=sharing`