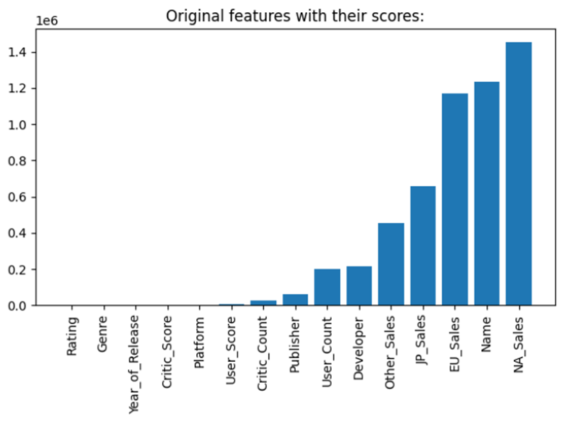
Categorial Feature Selection Using Chi-squared

Avraham Bar Ilan - 205937949, Omer Eckstein - 312350192

February 28, 2022

1 Abstract

Our project deals the automation of machine learning processes on tabular sets. The project allows the user to filter out less important columns and leave the model to practice only with columns and features that are most relevant to the process, there are a variety of methods and statistical calculations on how to do this.
We chose to do the project with a statistical test called Chi-Square on categorical variables - which allows us to estimate whether a particular variable has high P-Value with the variable selected as the Target. The code receives a data set that prints a list of all features and score they received (the score is based on the P-Value result, a higher score means that the feature is more related to the target feature), and presents the user with a graph for more convenient visual display.
For example, here is the graph which displayed to user, the dataset is about video games, and the selected target feature is global sales.



This graph shows all the features, both categorical and numerical and their score.

Then the program chooses which features are less important and removes them from the data set, at the end of the run a limited data set is obtained, a new list of the features and their score is printed for the user, and an updated graph is displayed to the user.

We tested our program on 4 Datasets:

1. Titanic Dataset. (Kaggle)

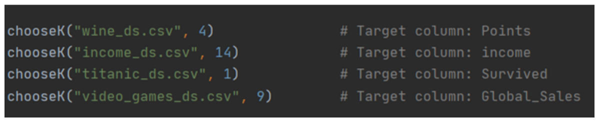
2. Wine reviews Dataset. (Kaggle)
3. Video games sales Dataset. (Kaggle)
4. Income classification Dataset. (Kaggle)

2 Problem Description

Our program designed to solve a problem in the data processing phase, after we have collected the data and arranged it into our structured data set we want to understand what information is essential for our model training, and what information is less important and we can give it up to reduce data size and training times. We found libraries that perform various statistical calculations like chi-sqaured test, but all of these libraries did not allow the columns to be selected automatically but only returned values that represent the strength of the p-value (the output of Chi-squared test).

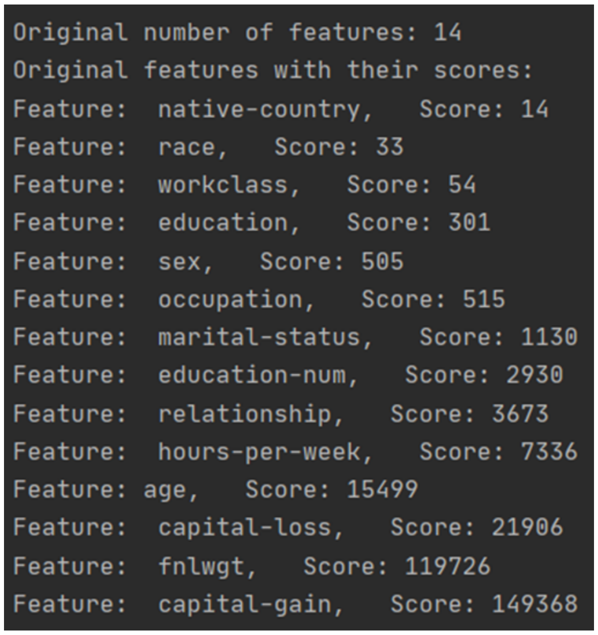
3 Solution Overeview

To execute the program, we first call its main function which called chooseK() with the name of the dataset and the index of the target feature, for example here is an execution for four different datasets:



```
chooseK("wine_ds.csv", 4)              # Target column: Points
chooseK("income_ds.csv", 14)           # Target column: income
chooseK("titanic_ds.csv", 1)           # Target column: Survived
chooseK("video_games_ds.csv", 9)       # Target column: Global_Sales
```

At the first step the main function calls the ds-loader() function, this function loads the csv file into Pandas Data Frame and then removes from it the column we got its index as the index of the target function. To perform chi-squared test we can not leave the categorical variables as a string variable, so we will convert them to an array of integers using Ordinal Encoding, using this encoding we will convert each variable to an array of the number of different values in the column, so that similar values will get the same vector. We used a library called: sklearn.preprocessing.OrdinalEncoder, which belongs to sklearn library. After encoding the dataset, we will also encode the target feature column and return both with a list of feature names. Now function uses sklearn.feature-selection.SelectKBest library of sklearn library. This class receives a data set with a target feature and returns p-value for each feature. Low p-value means that the features are more related to each other, in addition this class can return a score calculated according to the p-value but unlike p-value, a higher score means that the features are more related to target feature. In our program we used the scoring option because it is more convenient to work with graphs. After we got the scores for each feature we call showScores() function. That function will print to console the initial results and will show to user which features got higher score and which features get lower score that he will get first impression of our dataset.

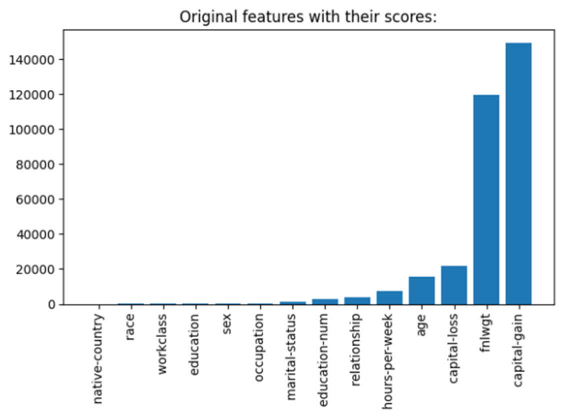


```
Original number of features: 14
Original features with their scores:
Feature:  native-country,   Score: 14
Feature:  race,   Score: 33
Feature:  workclass,   Score: 54
Feature:  education,   Score: 301
Feature:  sex,   Score: 505
Feature:  occupation,   Score: 515
Feature:  marital-status,   Score: 1130
Feature:  education-num,   Score: 2930
Feature:  relationship,   Score: 3673
Feature:  hours-per-week,   Score: 7336
Feature: age,   Score: 15499
Feature:  capital-loss,   Score: 21906
Feature:  fnlwgt,   Score: 119726
Feature:  capital-gain,   Score: 149368
```

Console printing after execution the program with Income classification dataset.

We can see the original amount of features is printed to screen and a sorted f the features according to score they received (a higher score mean the features are more related to each other).

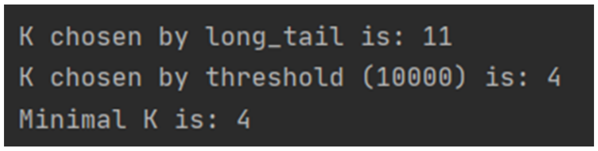
After printing that list, program will present the user a graph which will show that list in a more convenient way that will make it easier to understand the connections between the various columns and the target feature.



This graph was displayed after execution with Income classification dataset.

Once the graph is displayed the program wants to automatically select the important columns we want in our reduced dataset, the program is doing this by selecting the columns with the highest score, the selection is made by selecting K value which represents the number of columns (which have the highest score) we want to keep in our reduced dataset. To select that K which will bring us a small but effective dataset we will send the list of features and the scores of each feature to two functions, both will return K value and finally the program will choose the minimum K from both. The

first function is called: getK-threshold() because it calculates how many columns there are below a certain threshold (whose default value is 10,000) and returns the number of columns that pass this threshold as K. The second function is called: getK-long-tail() and it is answers the problem of a long tail in a graph. Many times there are very large amount of columns whose score is very low and we want to get rid of them, that function looks for an elbow point in the charts graph and returns the number of columns that are above the elbow point.
For example in the graph picture seen above you can clearly see that the first two columns have high score and will probably be able to teach the model a lot during training, and apart from these two columns the other columns have a rather long tail and will probably consume valuable training time and will not train our model effectively. After the two functions return the values of k the program selects the minimum and will print the user information about the selected K

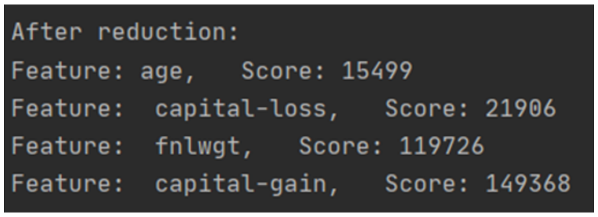


```
K chosen by long_tail is: 11
K chosen by threshold (10000) is: 4
Minimal K is: 4
```

Info about the chosen K which printed to console.

Once our program has selected an optimal K program calls the reduce-features() function, which will receive from us dataset and K value, and then returns a reduced version of our original dataset without the columns with low score as previously decided.

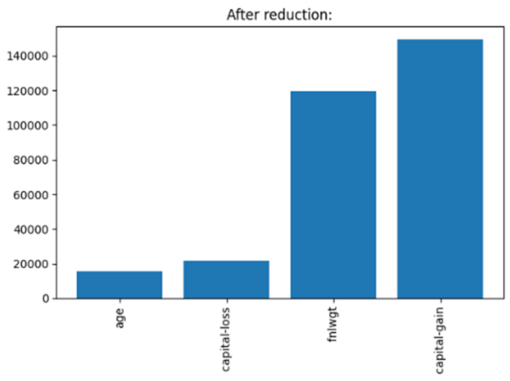
Now that we have a reduced version of the dataset, program will print to console the status of the columns and their score after the reduction, for example we will see what the program prints after its run and after it returns a reduced version of the dataset.



```
After reduction:
Feature: age,   Score: 15499
Feature:  capital-loss,   Score: 21906
Feature:  fnlwgt,   Score: 119726
Feature:  capital-gain,   Score: 149368
```

Console after execution and reduction on Income classification dataset.

Of course, we assume user wants to see the information in more convenient and graphical way, so program will presents a graph shows the column score after the reduction.



An example from execution on Income classification data set.

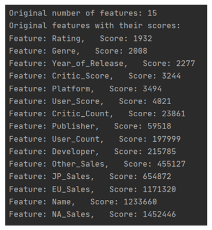
After program has presented the graph to user it has basically finished its run, if the programmer feels the column selection was too aggressive or less he can change the threshold.

The program does not save the file on the computer but only accepts it as a variable, of course adding a save command of the file to the computer is immediate but currently the code does not save the updated data set as a file.

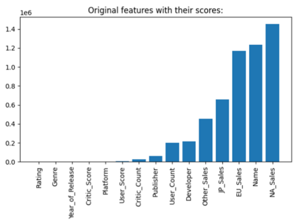
4 Experimental evaluation

To evaluate the program we ran it several times on the four attached datasets, each time a different target feature was selected and we followed the columns it throws, and it could be seen that columns that were less cushioned got a lower score and did go away and did not kept in the reduced version of the dataset.

For example, here is the list and graph we got from program executed with video-games dataset, with global-sales as target featurs,



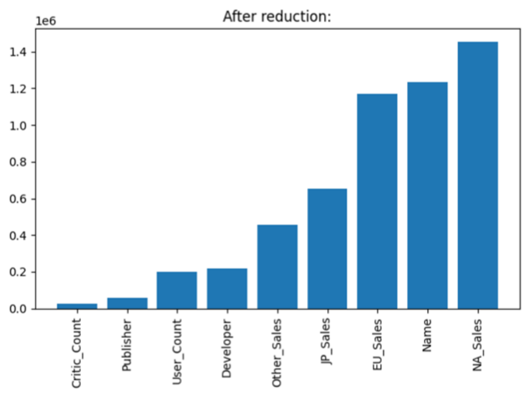
```
Original number of features: 15
Original features with their scores:
Feature: Rating,   Score: 1932
Feature: Genre,   Score: 2008
Feature: Year_of_Release,   Score: 2277
Feature: Critic_Score,   Score: 3244
Feature: Platform,   Score: 3494
Feature: User_Score,   Score: 4021
Feature: Critic_Count,   Score: 23861
Feature: Publisher,   Score: 59518
Feature: User_Count,   Score: 197999
Feature: Developer,   Score: 215785
Feature: Other_Sales,   Score: 455127
Feature: JP_Sales,   Score: 654872
Feature: EU_Sales,   Score: 1171320
Feature: Name,   Score: 1233660
Feature: NA_Sales,   Score: 1452446
```



Scores list and graph

Of course we can see that columns which deal with sales around the world have received a very high score, of course because the amount of global sales depends on sales in different regions of the world. It can also be seen that a column such as Year-of-release received a low score for the reason that of course there is no connection between the year of release of and the amount of sales from it.

We continued to ran the program and after our program selects K and produces a scaled down version of the dataset,list and graph will printed to screen of with the remaining features and their scores, of course you can see that the selected K left the significant columns and throw away the low scoring columns.



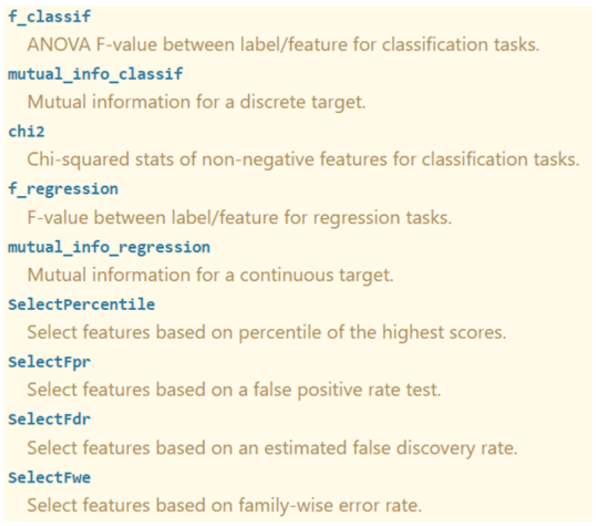
Kept columns and their score.

But we did met few difficulties. Before starting work on the project we thought that the process of reducing the data set would be simpler, but in practice when we tried to think about what considerations need to be included in choosing K value it turns out that in the current situation it is difficult to create automation that will always work on all data sets, each data has its own structure and his own needs, when we tried to make complex K choices (in terms of logic or mathematics) we sometimes found that what worked well for one data set would not necessarily work for another data set, so we created a relatively simple system logically and mathematically.

Moreover sometimes the p-values didn't work as we expected, for example when we checked it on Titanic dataset the program gives low score to gender of the survivor/injured, and we saw that it is one of the most correlative column when we want to check whether a person had a chance to survive titanic. So our program works well in most of cases but maybe more complex statistic calculation need to be added.

5 Related work

We did not find tools on the Internet that perform this automation from start to finish independently, but when we researched tools that perform statistical tests on features (both categorical and not) we came across many libraries, for example there is the library we used called sklearn.feature-selection And it has a big variety of statistical tests, each suitable for a different type of data and with different restrictions. For example there are several classes that each of them try to understand in a different way which are the more important columns and which are less.



`f_classif`	ANOVA F-value between label/feature for classification tasks.
`mutual_info_classif`	Mutual information for a discrete target.
`chi2`	Chi-squared stats of non-negative features for classification tasks.
`f_regression`	F-value between label/feature for regression tasks.
`mutual_info_regression`	Mutual information for a continuous target.
`SelectPercentile`	Select features based on percentile of the highest scores.
`SelectFpr`	Select features based on a false positive rate test.
`SelectFdr`	Select features based on an estimated false discovery rate.
`SelectFwe`	Select features based on family-wise error rate.

From official sklearn library website.

All of these libraries give information about the relationships between the columns but we did not find a library that gets a dataset and reduces it down and presents the user a simple summary of the process.

Our program does not compete with these libraries because they are more general, they are more versatile and can be used for more datasets, our software is more limited in options but performs a set of actions automatically and quickly. We took inspiration from these libraries when we thought of our solution and how it could be implemented.

To get to know better with categorial feature selection we found some articles on the web and read them. Here are some of them:

- Chi-Square Test for Feature Selection in Machine learning.
  towardsdatascience.com

- How to Choose a Feature Selection Method For Machine Learning.
  machinelearningmastery.com

- Ordinal and One-Hot Encodings for Categorical Data.
  machinelearningmastery.com

6 Conclusion

To complete this project we read a lot of articles and papers on the web, we tried various Python libraries and we found a lot of good blogs and Python libraries, for a conclusion we can say that we learned a lot and we learned about the complexity of data science.

We have learned from the project that data science combines a lot of trial and error and there is no definite mathematical way in which everything can be solved, but with the help of learning statistical tools one can get a good idea of how to proceed and improve the data to improve models. And that a deep understanding of the process is important for each and every step in the pipeline in machine learning.

But at the end we think that it is a pretty cool tool, we executed it on various datasets and every time we chose different target column and it was very interesting to see the hidden connections between the columns that we didn't expect them to get high score.