Python Project: Online News Popularity

By Aksel Yilmaz

Summary of the problem

Given a dataset composed of heterogenous properties of online news articles, the goal was to predict the popularity of said articles. However, considering the discrepency in the data distribution as well as in the information itself, it was necessary to first preprocess the data before attempting to predict using an AI model.

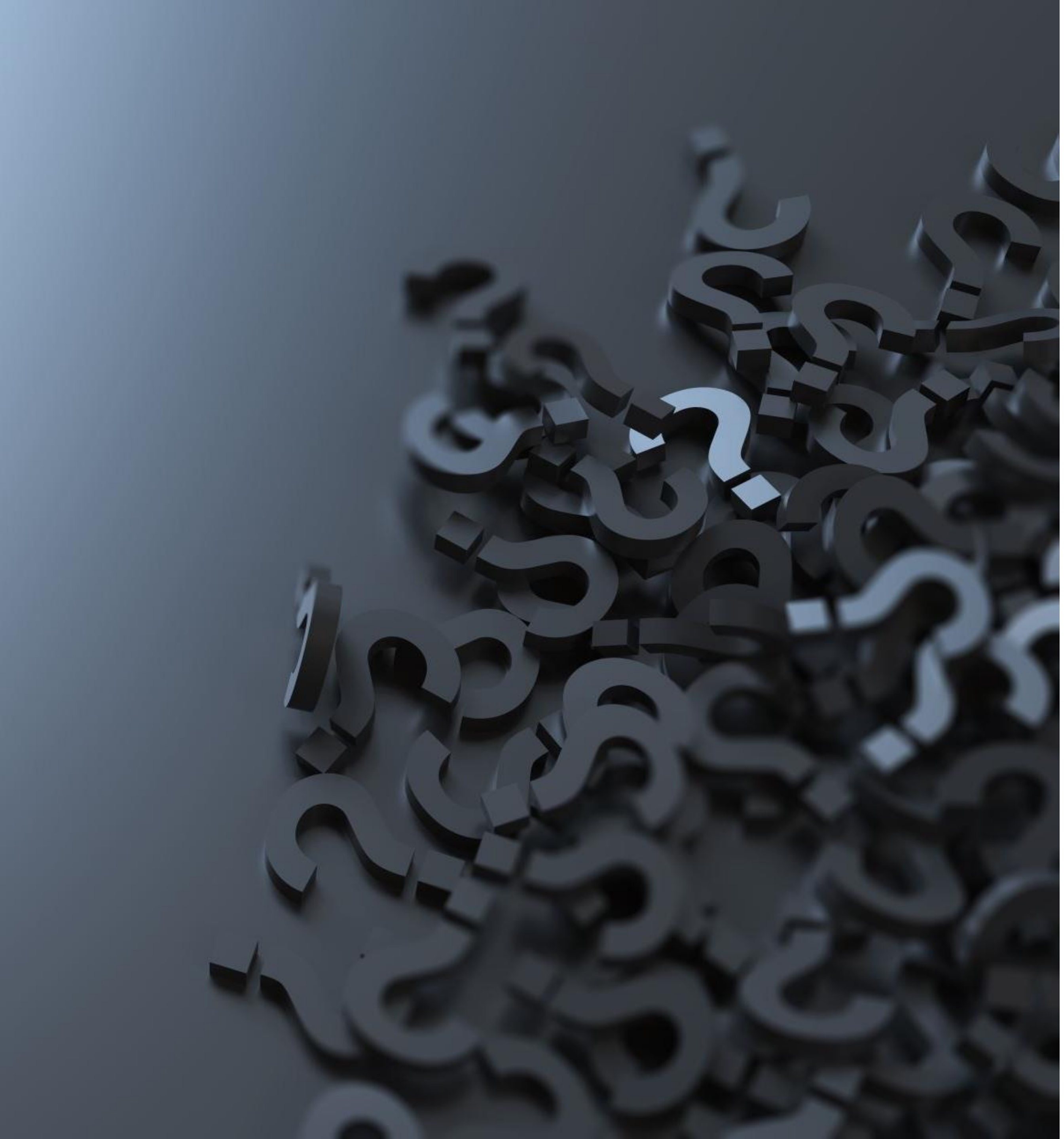
Why is this project interesting?

Nowadays, a lot of things can be predicted thanks to AI algorithms (e.g. market value, stock exchange, product demand, population evolution). News websites can also benefit from this practice. Indeed, understanding what makes an article popular among readers can help online-news companies produce better read articles. Furthermore, it can allow these companies to get a first impression on the articles they write in order to determine whether some points and aspects of those articles need to be reviewed in order to satisfy a larger demographic.

Issues with the dataset

The first issue I saw was the high number of columns. Indeed, too many values can hinder the efficiency of the prediction process, which we don’t want in this context.

The second issue that was more difficult to apprehend was the heterogenous data distribution. For example, a lot more articles are publish on weekdays rather than on the weekend. Furthermore, the number of articles that had huge success is extremely low compared to that of other popularity levels.



First step: grouping and rearranging the columns

One of the first things I noticed was that some columns consisted of boolean variables when they could be grouped into one column of type string.

What I also did was regroup the ‘ shares’ values into a column named ‘popularity’ consisting of categories (‘Very Poor’, ‘Poor’…) to prevent the model from having to be too precise to get a good score.

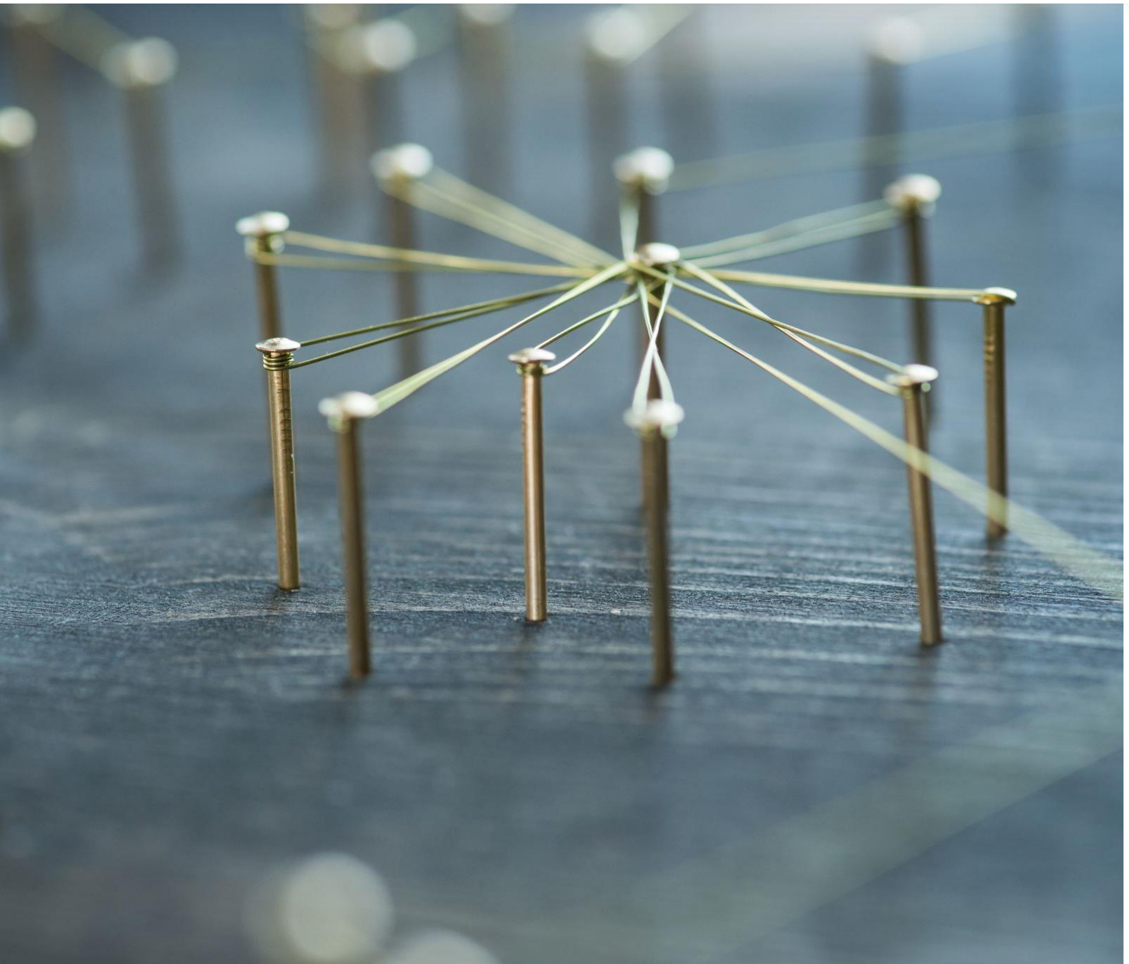
For example: the weekdays were ‘ weekday_is_Monday’ etc. so I grouped them into a column ‘weekday’ which had values like ‘Monday’, ‘Tuesday’ etc. and I dropped the original columns. I did the same to depict the channel the article was published in (‘Businnes’, ‘Lifestyle’…).

Second step: finding correlations between variables

My goal was to limit the number of columns to try and optimize the efficiency of the predictions. Therefore I computed the correlation matrix of the dataset (excluding the column I wanted to predict), selected each couple of columns that had a correlation value greater than 0.8 (or smaller than -0.8) and dropped one of the two each time.

This step left me with 39 predictive columns.

Testing several models



After the preprocessing was done, I tested different models with different parameters to determine which model with which parameters was the best for our problem.

I tested four models: Logistic Regression, Random Forest Classifier, KNN and SVC.

What I ended up as the best result was RandomForestClassifier(n_estimators=1000, n_jobs=-1, max_depth=50, random_state=0) with an accuracy of 51.22%.

Conclusion

An accuracy of 51.22% isn’t catastrophic but is far from good. Therefore, maybe some more complex models should be used and these models might work better if they were specifically trained on this dataset. Furthermore, the heterogenous data makes it difficult for models to achieve high accuracy thus some finer preprocessing might be required to get satisfying results.