Dataflow


Kaggle Dataset
Load to HDFS
Data Exploration
Data Cleaning
SandBox Creation

HYPOTHESIS

Is the helpfulness correlated with the review order?

- More specificaly, within a single book, is the time order of a review related with its helpfulness?

HYPOTHESIS

Is the helpfulness correlated with the review order?

```
ipeline_remove = {'$match':{
                    'Title':{'$exists':True},
                    'review/time':{'$exists':True},
                    'N_helpful' :{'$exists':True},
                    'Tot_votes' :{'$exists':True, '$ne':0}
                    }
            }

pipeline_project = {'$project':{
                        'review/time':1,
                        'Title':1,
                        'review/helpfulness_rate':{'$multiply':[
                                                    {'$divide':['$N_helpful','$Tot_votes']},
                                                    {'$sqrt':'$Tot_votes'}
                                                    ]
                                                },
                        'N_helpful':1,
                        'Tot_votes':1,
                        '_id':0,
                            }
            }
```

```
pipeline_group = {
        '$group': {
            '_id': '$Title',
            'totalCount': {'$sum': 1},
        }
    }

pipeline_remove2 = {'$match':{
            'totalCount': {'$gt': 10}
        }
}

pipeline_project2 = {'$project':{
            'Title': "$_id",
            "_id":0,
            "totalCount":1
        }
}
```

- Creation of a SandBox environment to test the Hypothesis
- MongoDB query to get data ready for analysis

HYPOTHESIS

Is the helpfulness correlated with the review order?

Data transformation

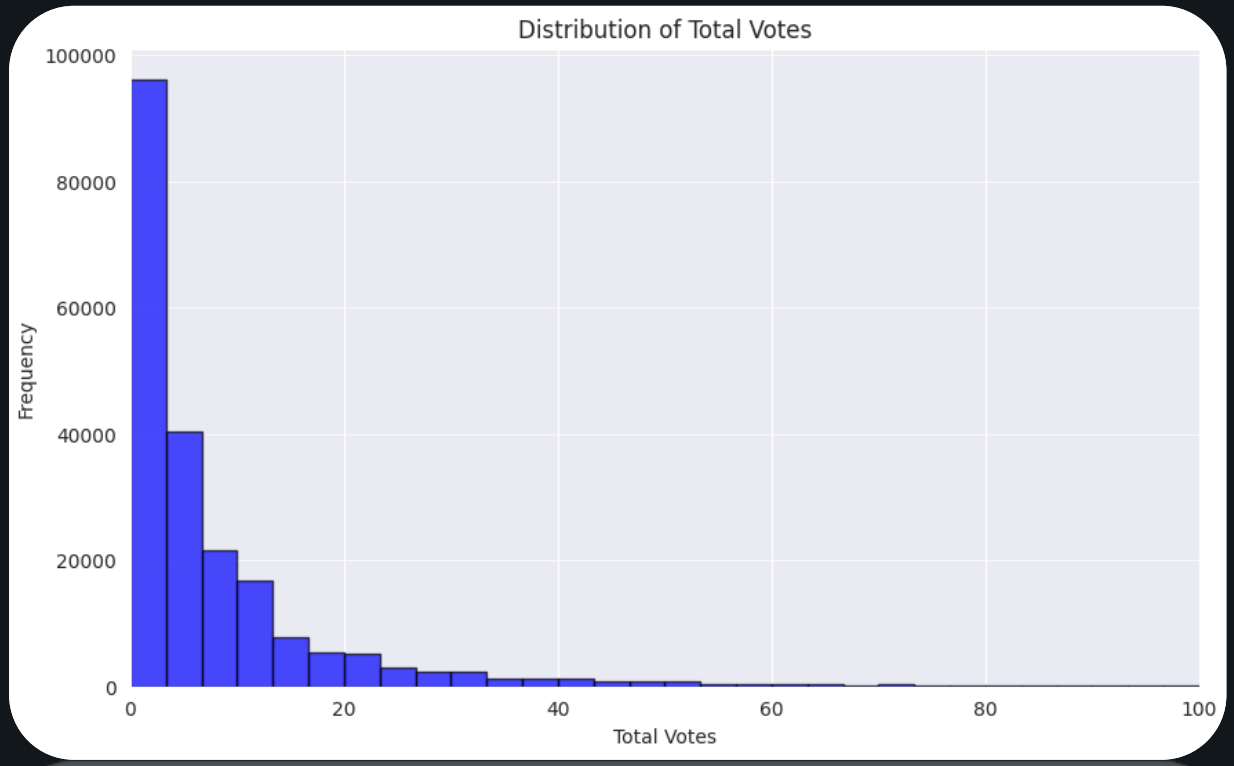
- Group reviews by Title
- Sort each group by time
- Substitute the time with a sorted integer array

```
# Group by Title
grouped = books_rating.groupby("Title")
grouped = grouped[['review/helpfulness_rate', 'review/time']]

sorted_groups = []

# Order each group by time and substitute the time with an ordered inte
for name, group in grouped:
    sorted_group = group.sort_values(by='review/time', ascending=True)
    sorted_group["review/time"] = range(0, len(sorted_group))
    sorted_groups.append(sorted_group)

# Concatenate all sorted DataFrames into a single DataFrame
sorted_df = pd.concat(sorted_groups)
```

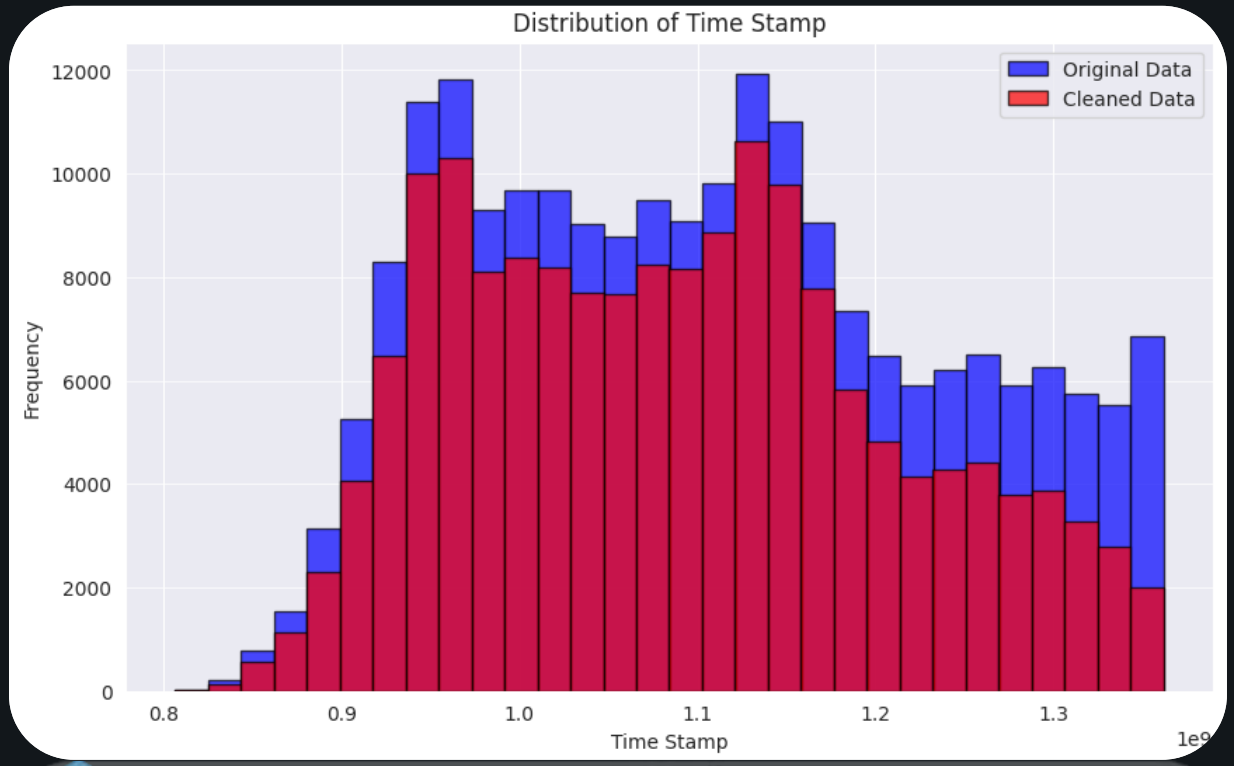
HYPOTHESIS

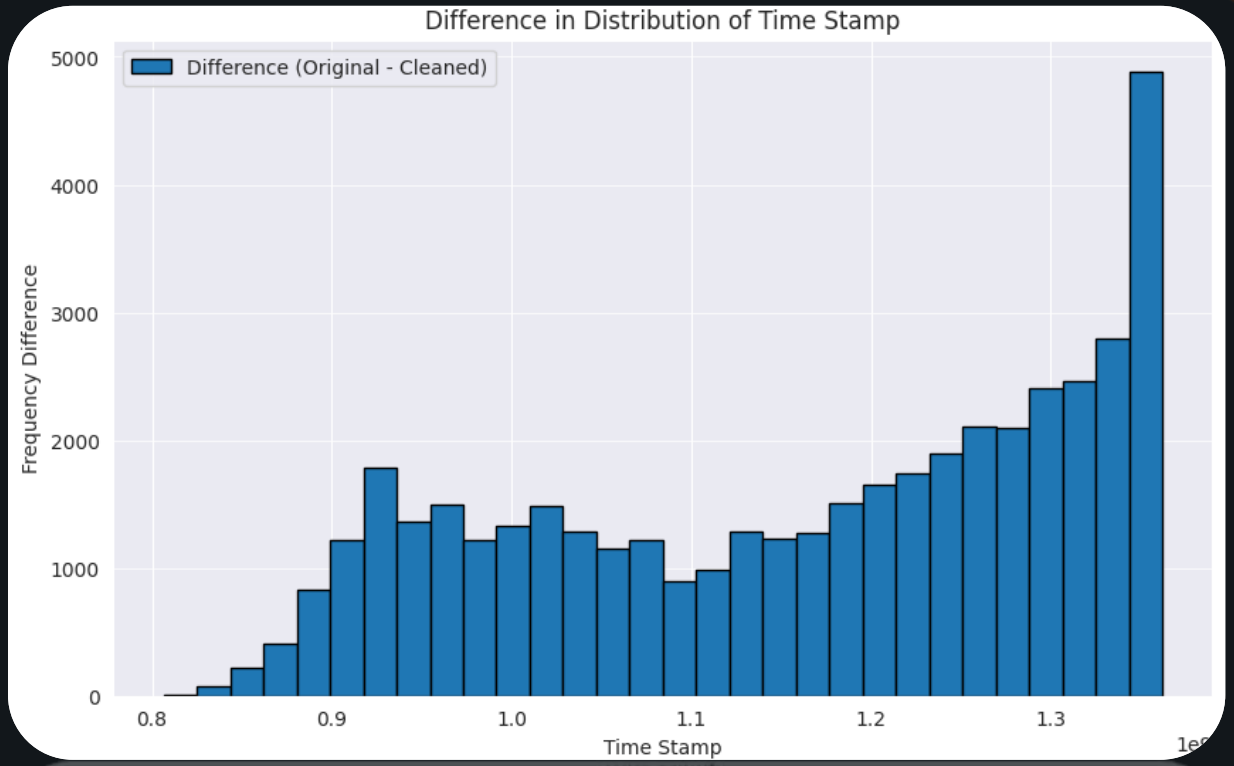
Total Votes Distribution

- Power law distribution
- Removed less significant values
- Tail not removed

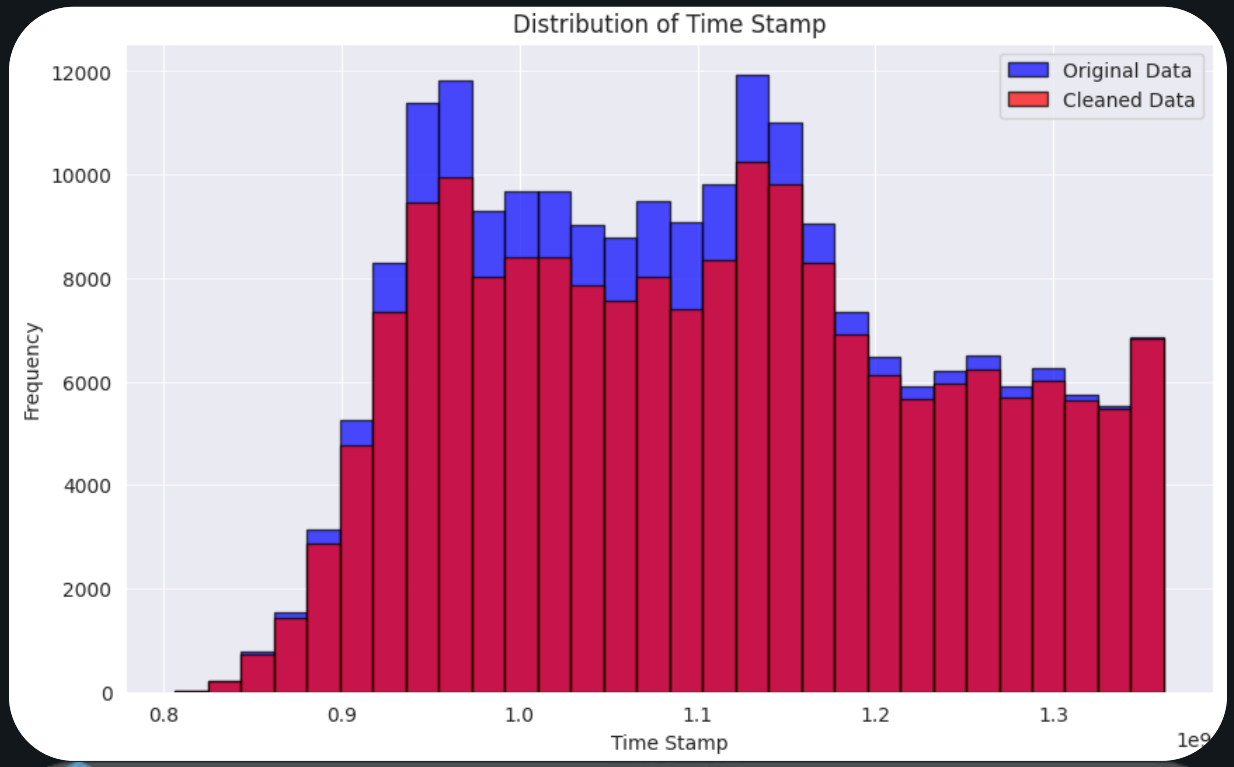


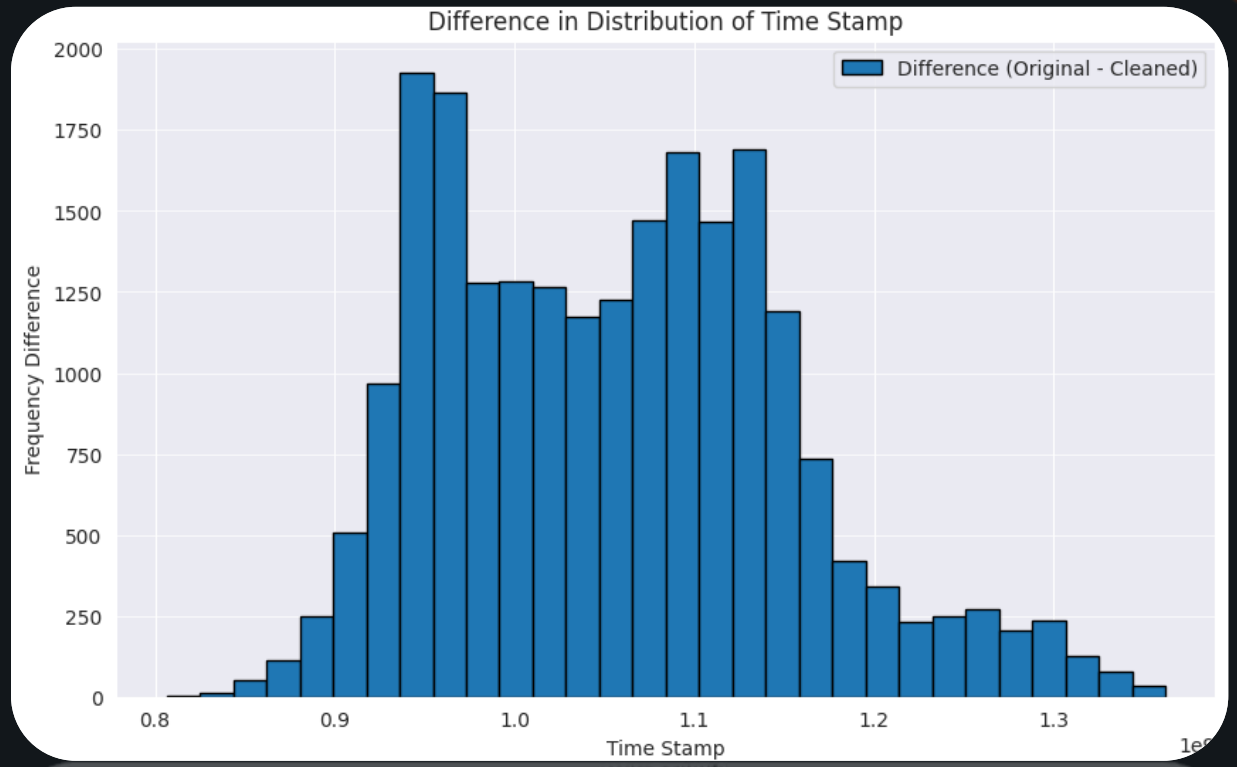
HYPOTHESIS


Distribution of Time Stamp
Original Data
Cleaned Data
Frequency
Time Stamp
1e9


Difference in Distribution of Time Stamp
Difference (Original - Cleaned)
Frequency Difference
Time Stamp

HYPOTHESIS

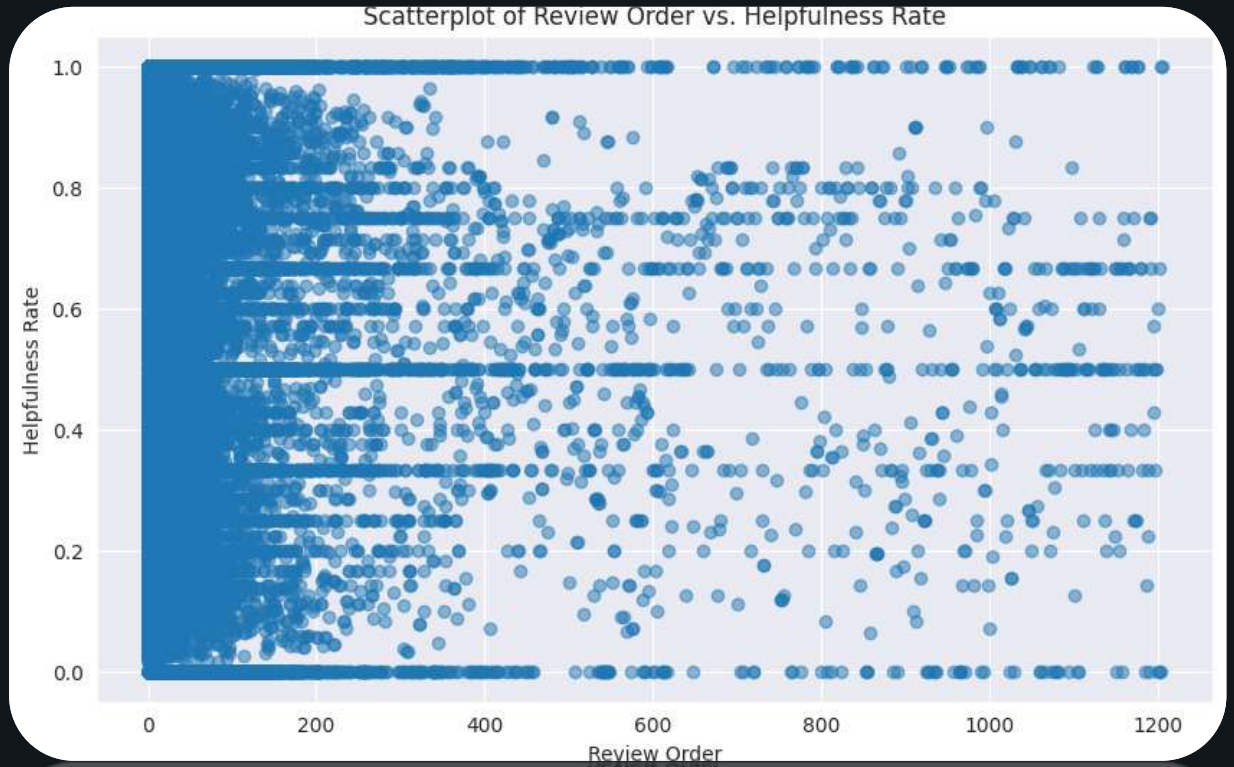

Distribution of Time Stamp
Original Data
Cleaned Data
Frequency
Time Stamp
1e9
0
2000
4000
6000
8000
10000
12000
0.8
0.9
1.0
1.1
1.2
1.3


Difference in Distribution of Time Stamp
Difference (Original - Cleaned)
Frequency Difference
Time Stamp
0
250
500
750
1000
1250
1500
1750
2000
0.8
0.9
1.0
1.1
1.2
1.3

HYPOTHESIS

Statistical test

- Scypy normaltest
  Not normal distribution
- Spearman correlation
  value = -0.2178
- P-value < 0.05



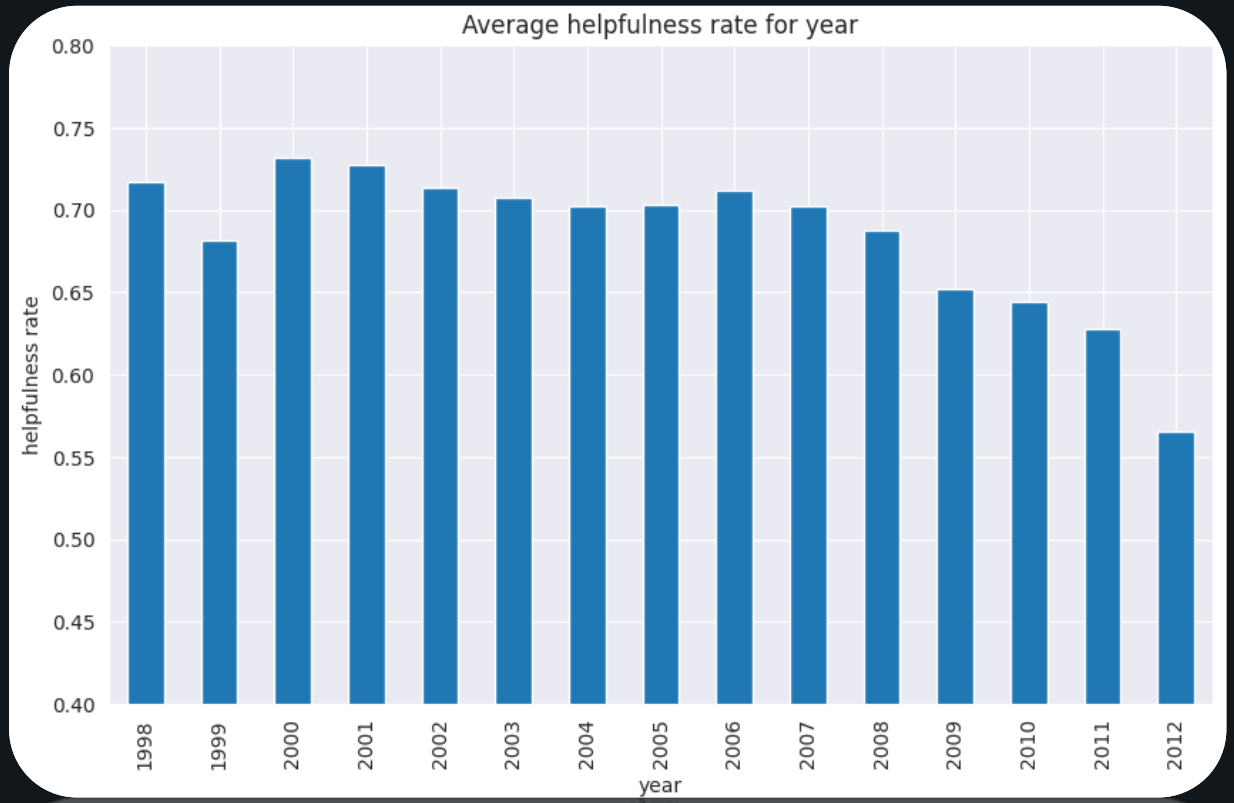
TREND ANALYSIS

Data cleaning and transformation

- Timestamp conversion to datetime
- Removed years with few reviews

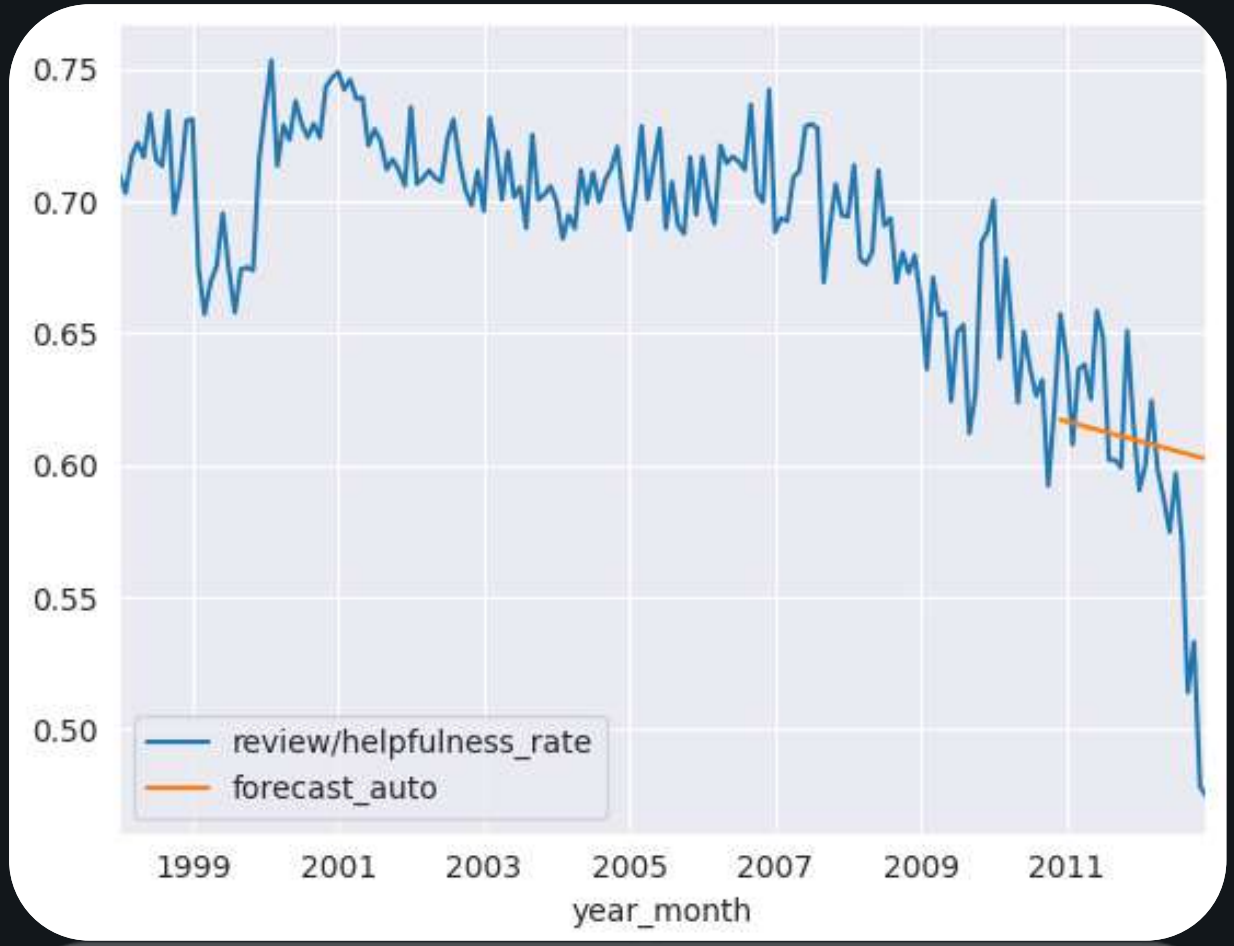
TREND ANALYSIS

- No daily trends
- No monthly trends
- Possible annual trends

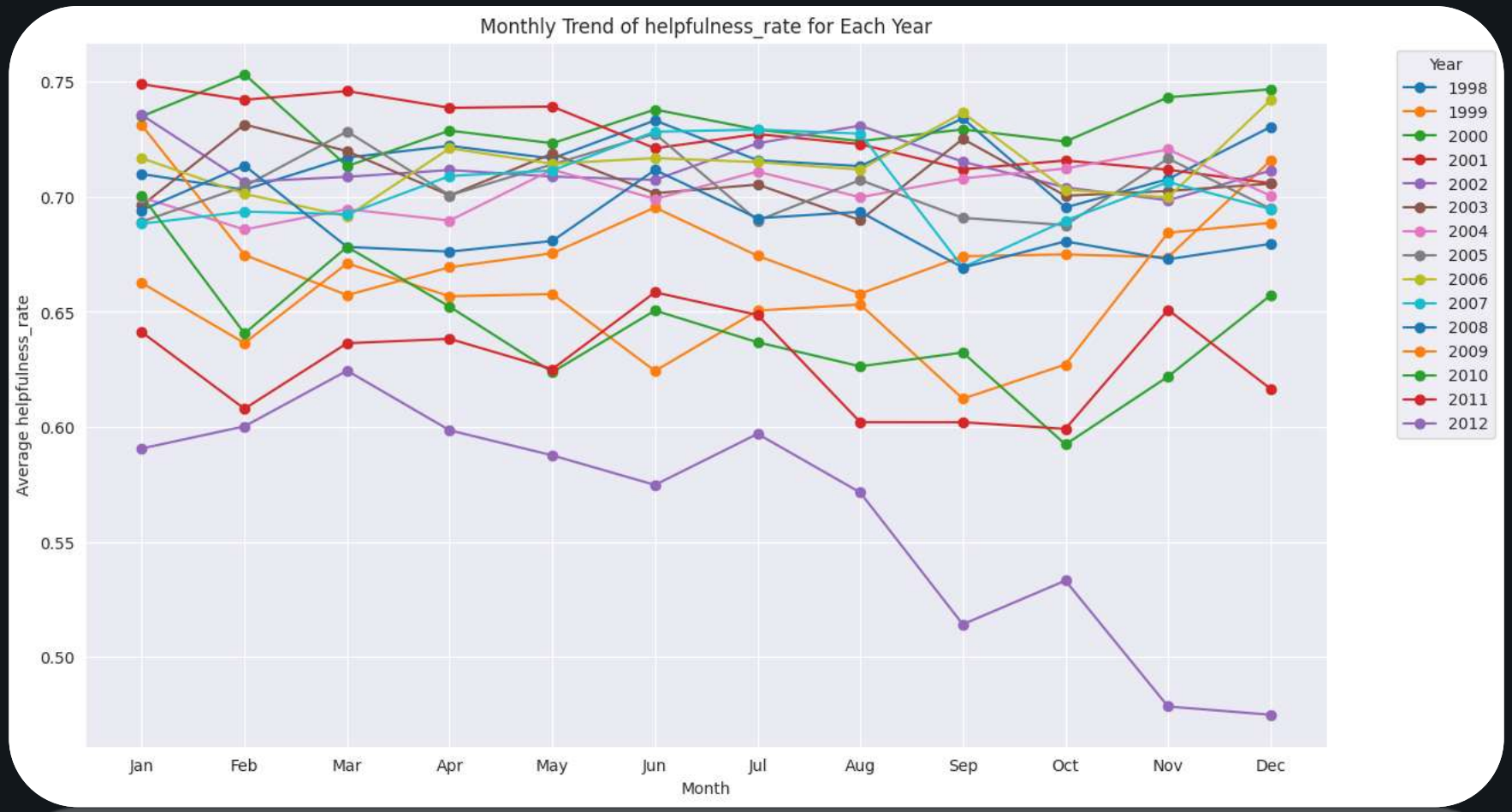


TREND ANALYSIS

- Negative helpfulness trend
- Collapse in the last year (2012)
- ARIMA Forcasting



TREND ANALYSIS



HYPOTHESIS

Final results:

- Using spark to handle a larger dataset (more then 2 milion of reviews) we obtained a slighly higher correlation:
  Spearman Correlation = -0.2577

Conclusions:

- Earlier reviews tend to be more useful, probably due to the fact that amazon show you just 10 reviews in the main page and amazon tend to push up the most usefull reviews. So the most useful reviews is the first useful review that arrive in term of time.