# Determining Project Success Ratio Depending On Complexity Of Projects:SmartShark

Kaushik Jagarlapudi[#1], Kameswara Pullabhotla[*2]

Dr. Mona Rahimi[#3]

[#]*Computer Science Department, Northern Illinois University*
*DeKalb, IL*

[1]rjagarlapudi1@niu.edu

[2]kpullabhotla1@niu.edu

***Keywords— project complexity, project success ratio, SmartSHARK analysis, SmartSHARK dataset***

## Abstract

SmartShark mining data contains data project, code as well as deployment related details. Predicting the projects that are more complex and measuring the success ratio of the projects can be only achieved by analyzing the issue details like priority, duration, status and in addition success of the projects can be determined using number of builds, their status and the duration of run; this paper presents two fold methodologies. Amongst the two models presented one which provides insights about the project complexity and the other which elaborates on the success ratio of the projects. This research elaborates and presents a clear segregation of projects with respect to the issues related as well as the build success ratio amongst the projects. As we are using a small dataset, the data is limited and contains information about only 78 projects for which project complexity is determined and 45 projects which are used for analyzing their success ratio depending on build state, duration and event type of the builds.

## Introduction and statement of the problem

This paper deals with the smartshark dataset which contains end-to-end project related data like commits, issues, builds, version control system details etc. The SmartSHARK dataset is a collection that contains project, version source control (VCS) details and continuous integration and deployment related data about the development of software projects that is complete and in detail. The contains diversified data that covers different characteristics about change requests, issue tracking details, continuous integration and deployment data, pull requests, commit details and code reviews, including other things. The databases(limited to small dataset) that we have considered are,

- Travis builds logs and commits information for all projects.
- Issues, issue system , and changes to issues made.
- Manually validated links between issue, event, issue, system as well as the type of issues labeled as bug for 78 projects.
- Manually validated links between travis build, commit, VCS system which includes the build state, duration and event type for 45 projects.

The database doesn't provide a comprehensive view of the information and lacks details as to how a project is performing with regards to its complexity and success ratio. So, with the help of the above manually validated links we formulated a model which predicts the complexity and also the success ratio of the projects. To do this, our research aims to develop the following research questions.

**RQ1**. How to determine complexity of issues and extend the same in finding the complexity of a project?

We gathered issue type, issue priority, issue hours, project id and issue id's of 78 projects from the SmarkSHARK dataset. We collected 5,294 issue level records which are mapped to these projects. After analyzing these records we only considered those with valid time durations and derived the final dataset with 4,327 records. These records are further processed in order to derive the complexity of the project.

**RQ2**. How to determine the success ratio of projects by analyzing build success rates?

We gathered duration, event type, build state, project id of 45 projects from the SmarkSHARK dataset. We got an overall of 23,620 commit level records which are mapped to these projects. Upon further analysis we were able to categorize the projects into k buckets and thereby clustered these projects based on their success ratio results.

### LIMITATIONS

The amount of data available for projects that is available in the SmartSHARK's data is the biggest drawback. The dataset needs significant computing power in order to host it on a server containing MongoDB. This dataset restricts the information that is provided in the sample or small dataset. It is also found that most of the projects present in the small dataset haven't got its dependent records in other tables. This is one of the challenges faced in analyzing the models described in this project. By expanding the database with a significant amount of relevant data projects—in the future, we want to get around this restriction by increasing the number of projects. This can be achieved by analyzing the same or different methodologies on the full dataset provided in the SmartShark data releases.

Another issue to consider was the lack of documentation for the tools, libraries, and API descriptions that were published. In addition it also lacks in providing the column level definitions for the tables present in the database. This made our task very challenging to understand and reuse them in selecting dependent and independent fields. Therefore, to overcome this problem the researchers should consider increasing the quality of the documentation for getting a deeper understanding of this data.

Model-based extraction and transformation framework DECENT is used as the footing for SmartSHARK dataset. DECENT offered a near perfect framework for amalgamating information. As the Linux kernel and Firefox have enormous memory requirements, this poses a danger to the data collection's capacity to scale. Nevertheless, Scheidgen et al. suggested a solution to the issue. This method demonstrated that it is better to transparently separate the various models into smaller pieces and store these pieces, for example, in a MongoDB.

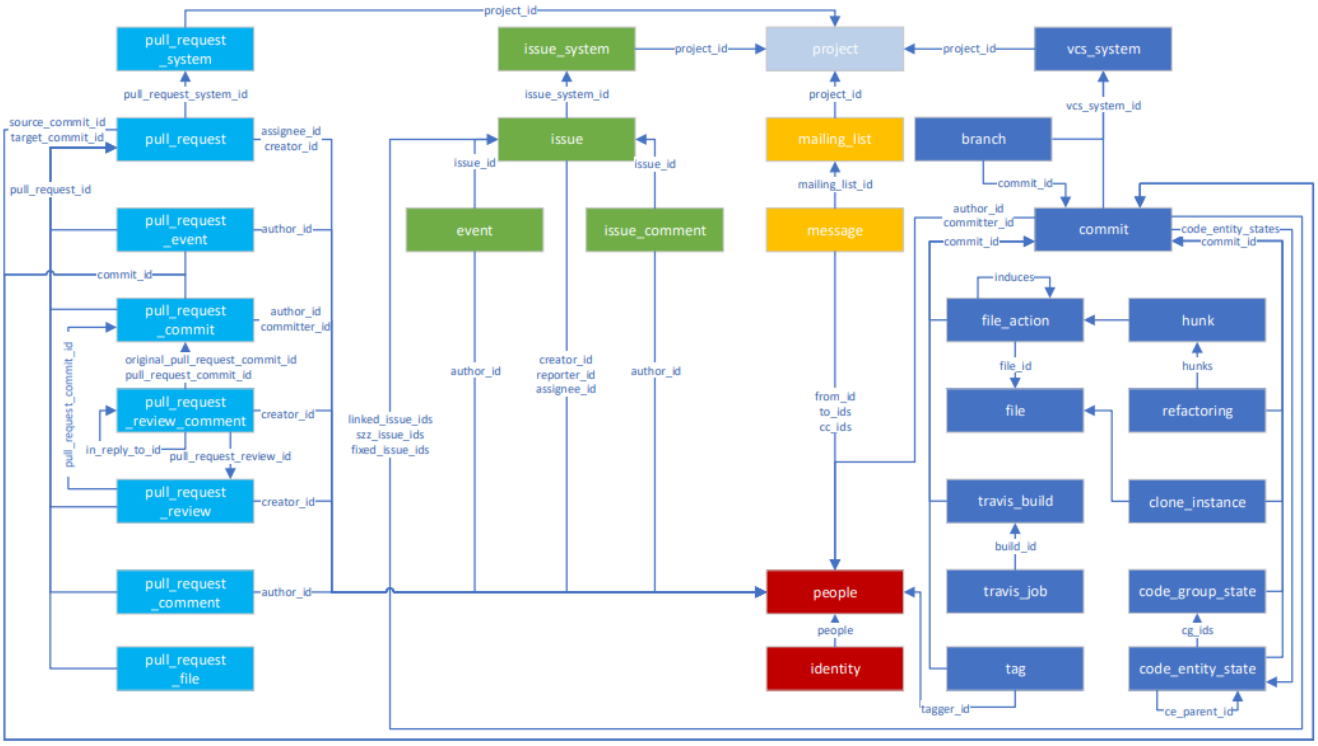
We now have used MongoDB to host the database, which is very hard to denormalize the data and can be only by creating multiple lookups as MongoDB is not a relational database and using lookup is even costlier on this big data given the scale of records that we have.

## METHODOLOGY

### SMARTSHARK DATA

As the data(see Figure 1) is not in a normalized format and each of the tables aren't related we had to spin up the entire dataset in MongoDB and build up relations between tables in bottom-up fashion so that we have consolidated information about,

(a) Issue id, issue type, issue priority, issue hours , project id as a single database
(b) Commit id, duration, event type, build state, project id

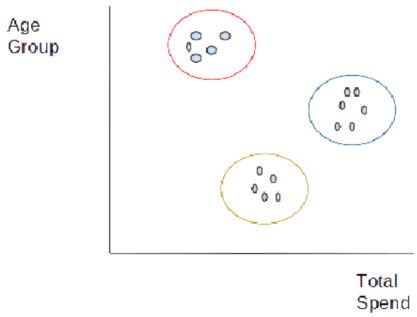



**K-Means Clustering** (Unsupervised Learning Technique) As the data is diversified and unlabeled we had to employ a clustering algorithm to get the predictions about the project complexity and the project success ratio.

Unsupervised learning techniques like clustering require the use of unlabeled datasets for the references. In general, it is utilized to record the grouping, underlying processes, and meaningful structure of a dataset. When clustering, the population must be split up into a number of groups so that the data points within each group are more similar to one another than the data points within other groups. It is, in essence, a grouping of items based on their similarities and differences.

We can find intrinsic grouping within unlabeled data using clustering. Although there are no set standards for good clustering, how a user chooses to use it for their particular purposes is entirely up to them. It can be used to identify unknown properties to locate a suitable grouping in the dataset or to find uncommon data points/outliers in the data.

Let's use a Walmart store manager as an example. You would like to better understand your customers so that you may grow your company by implementing fresh and more effective marketing techniques. Manually segmenting your customer base is challenging. If you have data on your clients' ages and past purchases, clustering might assist you organize them based on their purchasing habits. Following client segmentation, you can specify several marketing plans for each of the categories in accordance with target markets.




The above example(see Figure 2) is confined to two features but for the denormalized data that we have formulated we needed a multi dimensional plot which encapsulates all the features with which we formed the clustered.

## Inferring from Clusters

So, for each of the research questions which are the project complexity and the project success ratio once we have denormalized the data, we started to cluster the data basis the above mentioned methodology and for both the models that we have built we started to infer as to how the model has clustered the data by looking at issue type, issue priority, issue hours, project id for the project complexity and duration, event type, build state, project id for the project success ratio. Further we have also utilized a multi-dimensional plot to look at the clusters in an in depth manner.

## Literature Review

As part of literature review, we went through an in depth dive into the SmartSHARK dataset which is dedicated to mining software repository data. The idea behind SmartSHARK is to develop different tools for data gathering, processing, and analysis focused around a primary data source. A single website serves as the hub for all gathered data and also includes intermediate outcomes.

Powerful synchronous effects across the various tools in the SmartSHARK ecosystem are the approach's first advantage. For instance, tagging commits can be made more accurate by using the outputs of a source code-based refactoring detection tool rather than just the commit messages alone.

The second advantage of SmartSHARK is the high reproducibility of the work done within the software system. The entire software workflow is accessible as open source with thorough documentation. Every tool needed for experimenting with the pipeline is available and open to the public. In addition to that, the SmartSHARK plug-in technology enables easy expansions by both ourselves and other researchers. Since the plug-ins are essentially command line tools, this increases their availability to other researchers and allows them to be used beyond the SmartSHARK ecosystem.

## Outcomes

Based on our review, what we have found is that the dataset lacks analytic findings as the entire dataset isn't a relational database and needs key based access across all the tables. We finally arrived at the consolidated data by denormalizing the data by selecting multiple tables used in this research and performing further analysis.

## Experiment

### Data Preparation

In order to answer the research questions RQ1 and RQ2 mentioned above, we have analyzed the SmartSHARK dataset and collected a few tables which are used in our analysis. Below are the tables used in this entire research:

- **event**

  Contains data about the status of the issue id's their previous and current states. In this work we have considered

  - event_id
  - issue_type
  - old_value
  - new_value

- **issue**

Contains different issues that are associated with various issue systems, the priority of the issues and the different issue types they are related to. In this work we considered fields like

  - issue_system_id
  - issue_system
  - priority
  - status

- **issue_system**

This is one of the master tables that holds the parent keys used in this research. This table is used for creating relationships between issue systems and the project associated with them. Below are the fields used from this table:

  - issue_system_id
  - project _id

- **project**

This project is the master table that contains the project id's present in the SmartShark (small) dataset. Over all this table has 178 projects out of which 98 projects only have end-to-end details. Fields considered from this table are:

  - project_id

- **travis_build**

This table holds the details about continuous deployments of the projects present in the dataset. In this research we consider the below fields:

  - vcs_system_id
  - build_status
  - duration
  - event_type

- **vcs_system**

Using this we can find all the vcs systems associated for a given project. Fields extracted from this table:

  - vcs_system_id
  - project_id

As most of the above-mentioned tables are not relational, we transformed each of these tables and created a denormalized table one for each of the models using the features that we have mentioned above.

# MODELS

**RQ1**

Below is the denormalization deduced from the above elements that is considered as an input data for clustering projects with respect to their complexities.

| priority_val | issue_type_val | derived_value_conv_hrs |
|---|---|---|
| 2 | 1 | 60 |
| 2 | 2 | 4320 |
| 2 | 1 | 30 |
| 3 | 1 | 5 |
| 2 | 1 | 120 |
| 2 | 1 | 30 |
| 2 | 1 | 0 |
| 3 | 1 | 5 |
| 3 | 1 | 120 |
| 3 | 1 | 2880 |

In the above table, as we observe priority_value and issue_type have categorical values and derived_value_cov_hrs has a continuous value set.

The mapping for priority_val is as follows:

- ➔ 'Blocker': 1
- ➔ 'Critical': 1
- ➔ 'Major': 2
- ➔ 'Trivial': 3
- ➔ 'Minor': 3

The mapping for issue_type is as follows:

- ➔ 'Bug': 1
- ➔ 'Epic': 1
- ➔ 'Improvement':1
- ➔ 'New Feature':1
- ➔ 'Story': 1
- ➔ 'Task': 1
- ➔ 'Sub-task': 1
- ➔ 'Technical task': 1
- ➔ 'Question': 2
- ➔ 'Test': 3
- ➔ 'Wish': 4
- ➔ 'Brainstorming': 5
- ➔ 'Documentation': 6
- ➔ 'Dependency upgrade': 7

### Independent Variables

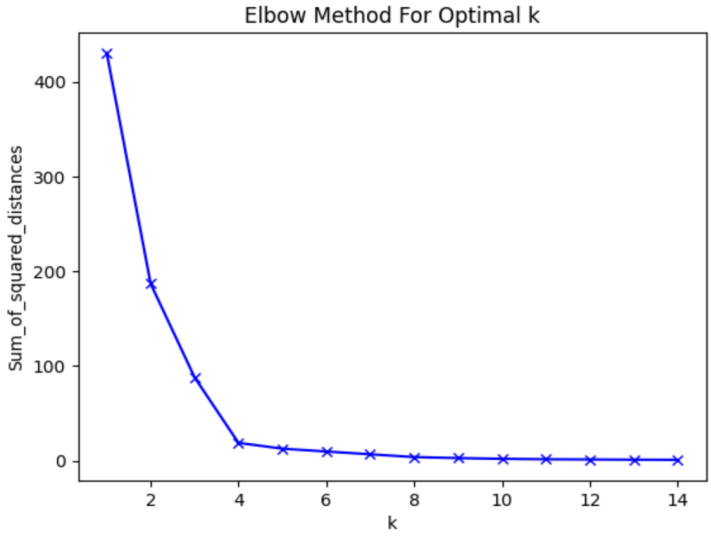
- issue_type
- issue_priority
- issue_hours

Applied **MinMaxScalar** to scale the entire dataset with a range of 0 to 1 as there is a presence of a continuous variable which might make the model biased towards categorical variables negating the effect of derived_value over the clusters formed. Below is the scaled data,

```
array([[5.00000000e-01, 0.00000000e+00,
2.97619048e-04],
       [5.00000000e-01, 1.66666667e-01,
2.14285714e-02],
       [5.00000000e-01, 0.00000000e+00,
1.48809524e-04],
       ...,
       [1.00000000e+00, 0.00000000e+00,
2.97619048e-04],
       [1.00000000e+00, 0.00000000e+00,
0.00000000e+00],
       [5.00000000e-01, 0.00000000e+00,
0.00000000e+00]])
```

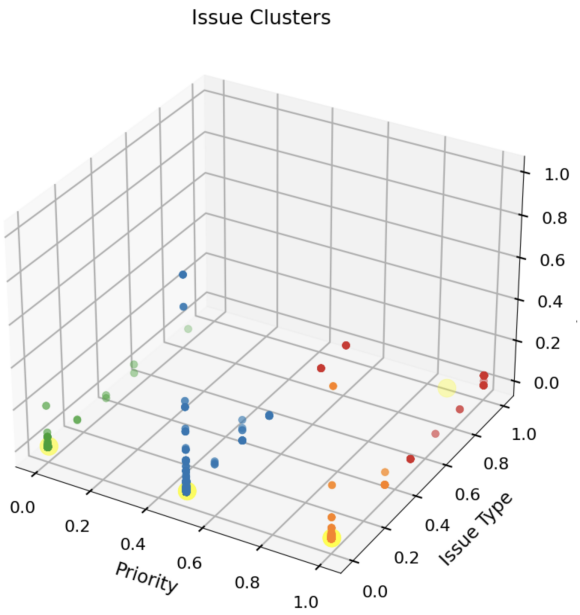
### Elbow Method

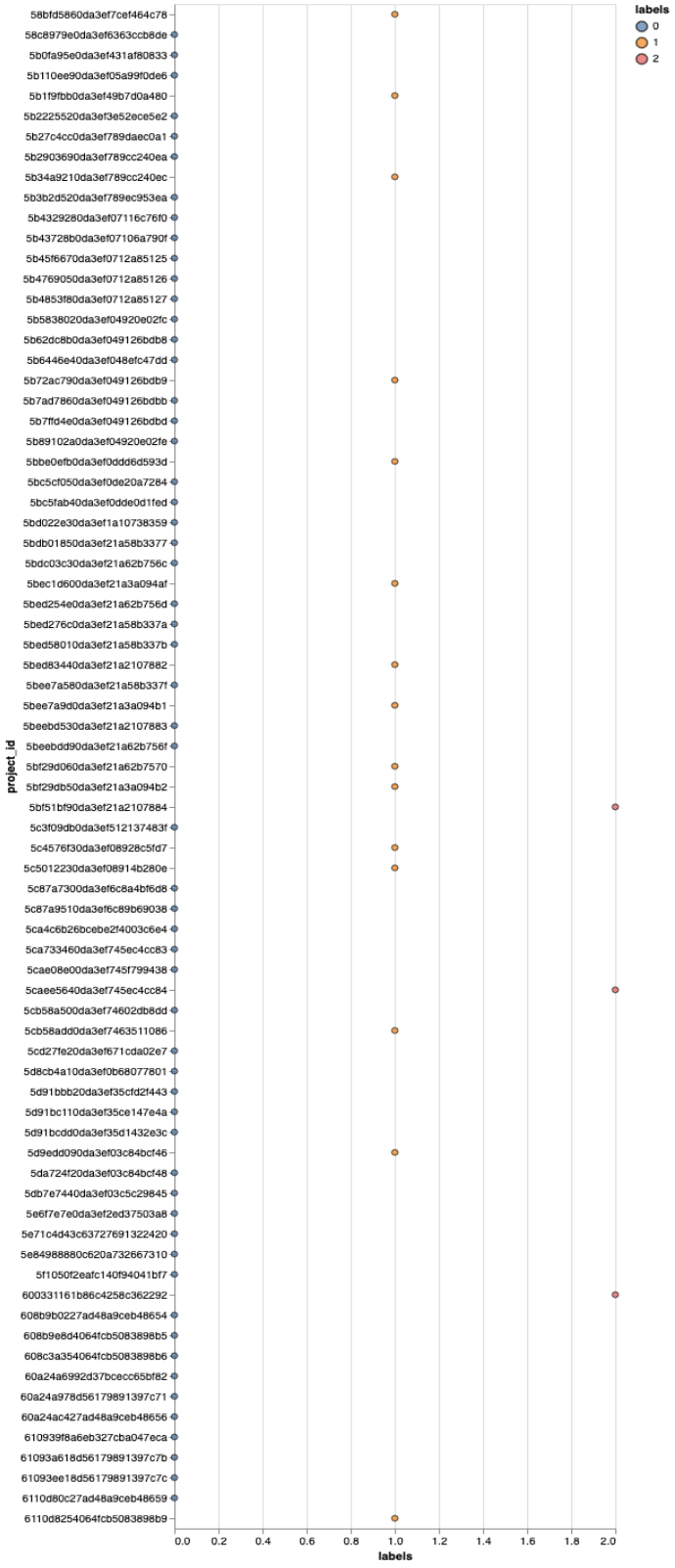
In order to find the optimal “k” value with which we can bucket or cluster the dataset, a variety of values are collected, and applied using the elbow method using K-Means. This analysis helps us in choosing the ideal k-value for defining the number of clusters. The point in the "elbow" curve from which the line becomes linear is a solid sign that the underlying model fits well at that point and the line resembles an arm. Post this point the line gets linear which states there is no change that could be observed beyond the elbow point.




Using this ‘K’ value i.e., “4” we have applied the K-means clustering technique on the data that is derived from the MinMaxScalar.

```
from sklearn.cluster import KMeans
kmeans = KMeans(4, random_state=0)
labels =
kmeans.fit(data_transformed).predict(dat
a_transformed)
```




Once we plotted the multi dimensional graph(Figure 3.) for all the features and their respective labels we then took counts for each of the labels for each project and the label with the highest count for each of the project and plotted them. Now when we refer both the figures what we can infer is that

- Most of the projects are labeled 0 which means that they have low complexity.
- Few of the projects were labeled 1 which means that they have medium complexity.
- There are less number of projects that were labeled 2 which have higher complexity.

**RQ2**

Extending the above work, we derived the success ratio of the projects based on the efficiency of the project builds. For this we have leveraged the data resulted from the previous model and extracted the data from travis_job, vcs_system and project tables. Below is the denormalized data.

| duration | event_type | state |
|---|---|---|
| 31 | 1 | 1 |
| 538 | 1 | 1 |
| 28 | 1 | 1 |
| 446 | 1 | 1 |
| 31 | 1 | 1 |
| 746 | 1 | 1 |
| 727 | 1 | 1 |
| 823 | 1 | 2 |
| 822 | 1 | 2 |
| 812 | 1 | 2 |

The mapping for event_type is as follows:

- ➔ 'Push': 1
- ➔ 'Cron': 2
- ➔ 'Api': 3

The mapping for state is as follows:

- ➔ 'Failed': 1
- ➔ 'Passed': 2
- ➔ 'Errored': 3
- ➔ 'Canceled':4

**INDEPENDENT VARIABLES**

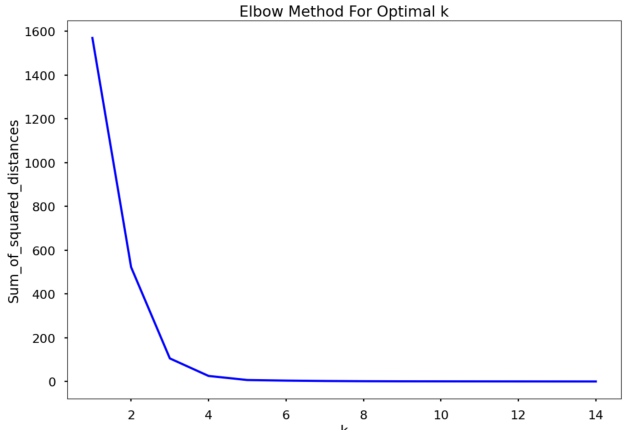
- build_state
- event_type
- duration

As there is a continuous variable present, we used **MinMaxScalar** to scale the entire dataset within a range of 0 to 1, as not performing this could cause the model to be biased towards categorical variables, negating the effect of duration over the clusters created. The scaled data are shown below,

```
array([[0.9656268 , 0.        , 0.
],
[0.96633025, 0.        , 0.        ],
[0.96562264, 0.        , 0.        ],
       ...,[0.96672568, 0.        ,
0.33333333],
[0.96672429, 0.        , 0.33333333],
[0.96671042, 0.        , 0.33333333]])
```

**ELBOW METHOD**

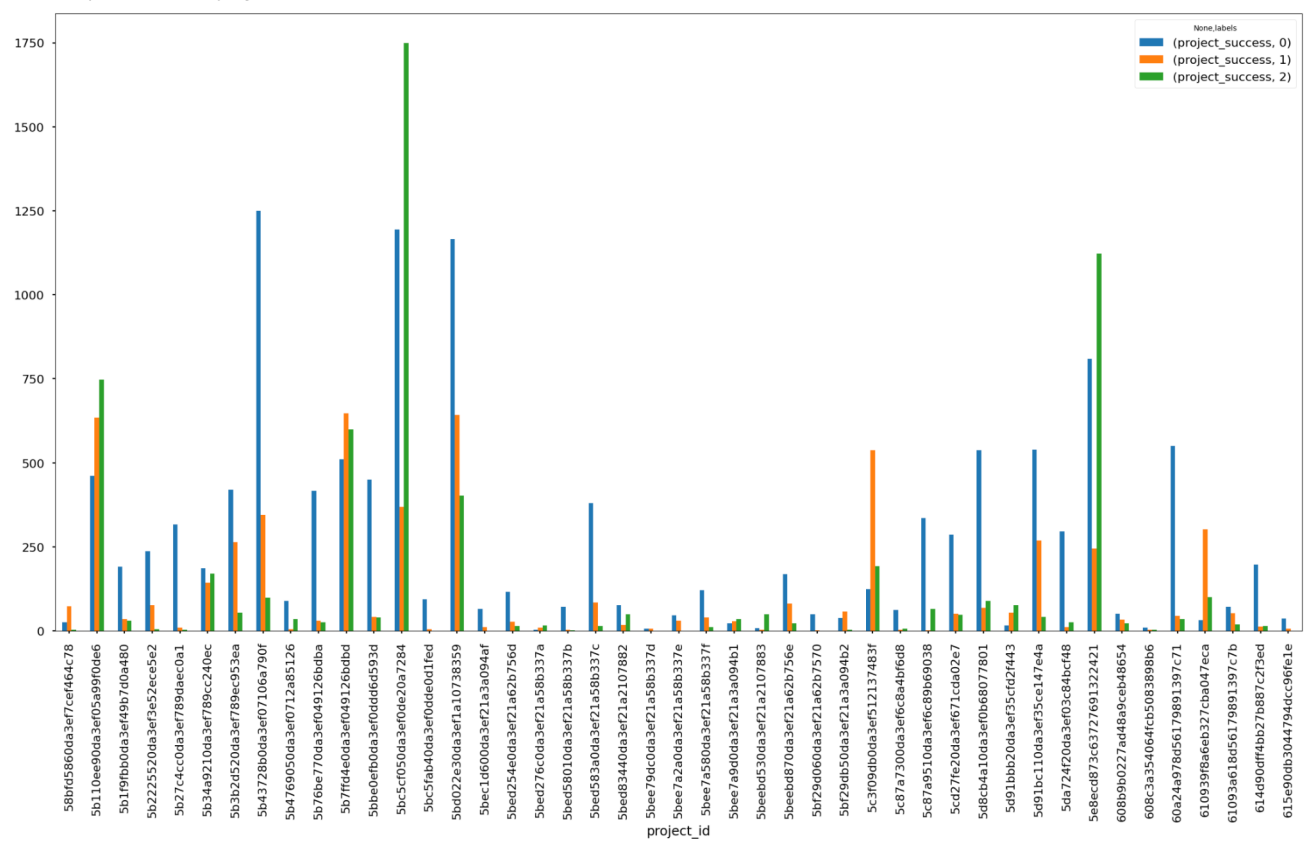
A range of variables are gathered and applied utilizing the elbow approach with K-Means in order to identify the best "k" value with which we can cluster the dataset. Using this technique, we may select the best k-value to specify the number of clusters. A clear indication that the underlying model fits well and the line resembles an arm is the point in the "elbow" curve from where the line turns linear. After this point, the line becomes linear, indicating that no change can be seen past the elbow point.



We have used the K-means clustering algorithm on the data that is obtained from the MinMaxScalar using this "K" number, which is "3".

```
from sklearn.cluster import KMeans
kmeans = KMeans(3, random_state=0)
labels =
kmeans.fit(data_transformed).predict(dat
a_transformed)
```

So, we can infer that project who have more number of labels as “0” are categorized as successful projects. Count of “1” indicated average performing projects, followed by “2” which are poorly performing projects.




Above displayed is the entire statistics of the projects that are clustered based on the success ratio of the builds. Here,

- ➔ 'Build State - Passed: 0
- ➔ 'Build State - Failed': 1
- ➔ 'Build State - Errored/Canceled':2

## THREATS TO VALIDITY

The main issues with the results' external validity worked as the main motivation behind our work on SmartSHARK. We examined the effects a dataset like SmartSHARK can have on these issues, how it can in turn help to address them, or

what additional work might be required based on our experience.

(a) **High reutilization of datasets.**

A platform like SmartSHARK generally nails this problem, since the actual data can be extended with new projects. This way, SmartSHARK provides an exoskeleton for a natively increasing database, which explains that with every potential experiment, updated data can be used. However, SmartSHARK also demonstrates the limitation regarding the research questions that we have added: which is that the analytical study becomes more difficult as we start to add more data into the tables.

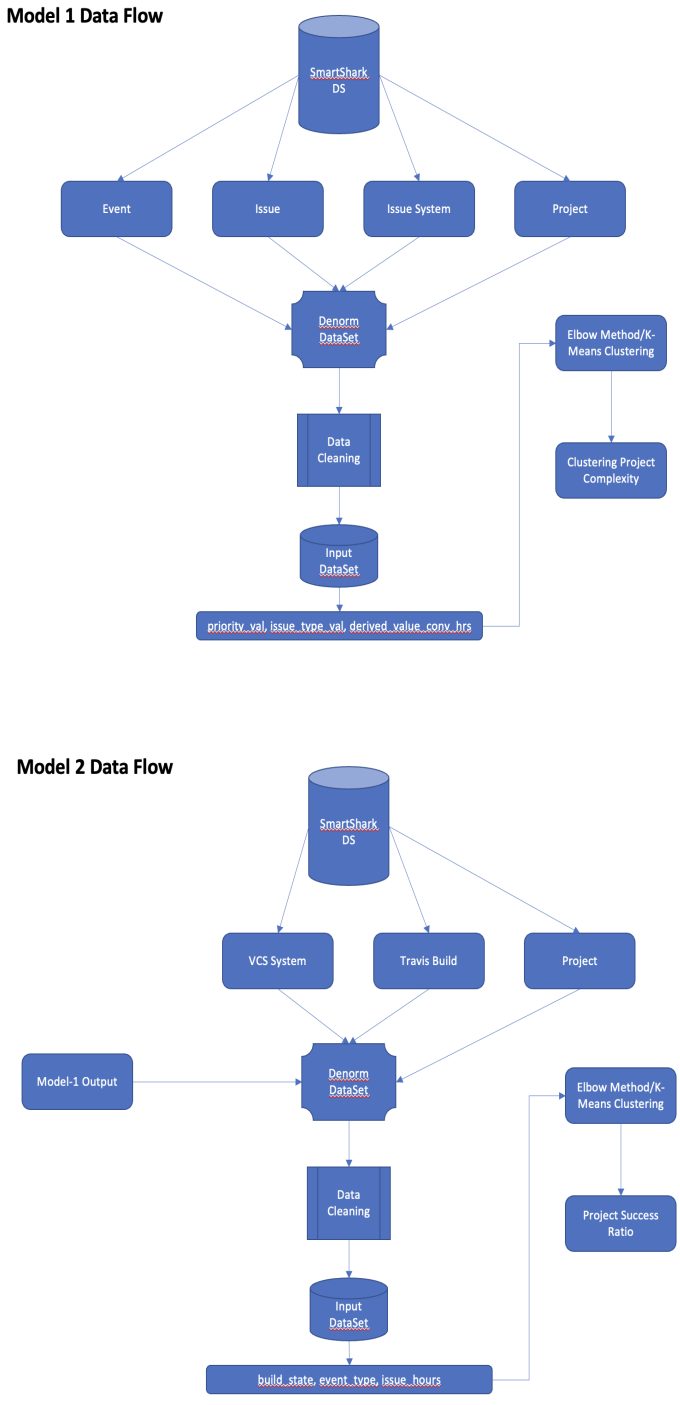
(b) **Unavailability of data.**

SmartSHARK tackles this in two possible ways to handle this issue: the first is a shared cloud deployment, where all have access to the data. The second is creating and sharing the backups of the data hosted over MongoDB server. This also has an inherent disadvantage as the computation time is denormalizing the data is huge.

## Conclusion & Future Work

As this is an unsupervised learning problem, we have evaluated the outputs of Model-1 i.e, clustering projects based on complexity and Model-2 i.e., determining the success ratios of the projects manually looking at the SmartShark dataset values and the results inferred in the bar graph. This can be extended using the large SmartShark dataset and introduce the committer or the developer related details for determining developers efficiency per project level. This can in turn be used as a dependent metric for project complexities which were determined using Model-2 in this work.

## Project End-To-End Flows



## References


[1] "The Developer Data Platform." *MongoDB*, https://www.mongodb.com/

[2] "Architecture." *SmartSHARK*, https://smartshark.github.io/architecture/

[3] Göttingen, Fabian Trautsch Georg-August-Universität, et al. "Adressing Problems with External Validity of Repository Mining Studies through a Smart Data Platform: Proceedings of the 13th International Conference on Mining Software Repositories." *ACM Conferences*, 1 May 2016, https://dl.acm.org/doi/pdf/10.1145/2901739.2901753

[4] University, Fatemeh Khoshnoud Shiraz, et al. "Which Bugs Are Missed in Code Reviews: Proceedings of the 19th International Conference on Mining Software Repositories." *ACM Conferences*, 1 May 2022, https://dl.acm.org/doi/pdf/10.1145/3524842.3527997

[5] A. Frank Ackerman, Lynne S. Buchwald, and Frank H. Lewski. 1989. Software inspections: an effective verification process. IEEE software 6, 3 (1989), 31–36.

[6] A Frank Ackerman, Priscilla J Fowler, and Robert G Ebenau. 1984. Software inspections and the industrial production of software. In Proc. of a symposium on Software validation: inspection-testing-verification-alternatives. 13–40.

[7] Anonymous. [n. d.]. Which bugs are missed in code reviews: An empirical study on SmartSHARK dataset. https://doi.org/10.5281/zenodo.5993800

[8] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. Psychological bulletin 76, 5 (1971), 378.

[9] Xiaofeng Han, Amjed Tahir, Peng Liang, Steve Counsell, and Yajing Luo. 2021. Understanding code smell detection via code review: A study of the openstack community. In 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC). IEEE, 323–334.

[10] Herbold, Steffen, et al. "A Fine-Grained Data Set and Analysis of Tangling in Bug Fixing Commits - Empirical Software Engineering." *SpringerLink*, Springer US, 2 July 2022, https://link.springer.com/article/10.1007/s10664-021-10083-5#Sec1

[11] Goettingen, Alexander Trautsch University of, et al. "The SmartSHARK Ecosystem for Software Repository Mining: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings." *ACM Conferences*, 1 June 2020, https://dl.acm.org/doi/pdf/10.1145/3377812.3382139

[12] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, 10 2018, pp. 908–911.

[13] G. Gousios, "The ghtorent dataset and tool suite," in 2013 10th Working Conference on Mining Software Repositories (MSR). IEEE, 2013, pp. 233–236