**A**

**PROJECT REPORT**

**ON**

# PROJECT RECOMMENDATION SYSTEM

**For subject Big Data Application Development.**

***Submitted By***

**MONIL SHAH (16012121027)**
**NINAD THAKER (16012121034)**
**DEV TRIVEDI (15012121035)**
**PRINCE PATEL (15012121020)**



NOVEMBER 2019

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING U.V.PATEL COLLEGE OF ENGINEERING GANPAT UNIVERSITY**



# CERTIFICATE

## TO WHOM SO EVER IT MAY CONCERN

This is to certify that students of B.Tech. Semester (VII) (COMPUTER SCIENCE & ENGINEERING) has completed his one full semester on project work titled "**PROJECT RECOMMENDATION SYSTEM"** satisfactorily in subject **"Big Data Application Development"** of **Computer Science & Engineering**, Ganpat University in the year 2019.

MONIL SHAH (16012121027)
NINAD THAKER (16012121034)
DEV TRIVEDI (16012121035)
PRINCE PATEL (15012121030)

**Internal Guide**
Prof. Aniket Patel

**Date:**

**HOD**
**CSE Department**
Dharmesh Darji

**Date:**

# ACKNOWLEDGEMENT

We have been constantly putting our efforts to make this project possible. However, it would not have been possible without the kind support and help of many individuals.
We would like to extend our sincere thanks to all of them.

We are highly indebted to **Prof. Aniket Patel**, **Prof. Parth Parekh** and **Prof. Arun Paswan,** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude members of **Institute of Computer Technology (ICT)** for their kind co-operation and encouragement which help us in completion of this project.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

# ABSTRACT

Based on the analysis of the recommendation system, it is now known that the recommendation for a particular thing tends to work when the recommendation comes from ideally known and familiar faces or someone who honestly describes pros and cons of the thing. Here, we have tried to use the review from different students who have selected some projects in their career and by using machine learning, we created a model which then based on one's interest, provides best project out of available ones in the same field.

# CONTENTS



**List of Tables:**

# INTRODUCTION

## 1.1. Analysis of the Recommendation System:

Recommendation system works by taking data which is classified and has all kind of necessary fields in it. The system is dependent upon the data provided to it as a feedback or historical data. Unless there exists a set of data for the given field, the recommendation engine will fail to recognize the required parameters and thus will not be able to provide required review as a suggestion to the user.

Based on a few opinions the engine will provide biased review, since, there might exists such cases which machine never encountered during its training which can result in bad recommendation. As for the number of the data points for one particular thing, the more points provided, the more accurate recommendation will be provided.

There are 2 basic types of recommendation engine which can be used in the project. First one is Content Based Filtering and Second is Collaborative Filtering. The one which is used here is Collaborative filtering algorithm.

## 1.2. Project definition:

In the Project Recommendation System, we have randomly generated data. Firstly, our dataset contained a few columns which were not required for the project thus, we dropped all such columns before doing any further process. Then, we plotted the distribution of project rating plotly library's offline plotting function to provide an overview of the dataset we also created another graph for distribution to display ratings per project and as a challenge since majority reviews were neutral, the graph showed biasedness in it. This would allow us to visualize the actual status of recommendable content from the dataset.

# PROJECT SCOPE

The field of this project is very specific about recommending project to student, though we are implementing it for the university students, there is always room for this project at various levels such as if a company is using it, then they can easily assign a project to an employee based on the area of interest and availability of the project in that region. This can aid the project manager to understand which employee with required skill set will be best. Recommendation system has been proven to be a valuable technique as it aims to predict the preference to an item of a target user. Machine learning algorithms include Singular Value Decompositions (SVD), Naive Bayes Line only, KNN Baseline, KNN Basic, Normal Predictor and Co-Clustering which is measured under the RMSE – Root Mean Square Error of machine learning errors to find accuracy or the least error.

# SOFTWARE AND HARDWARE REQUIREMENTS

## 3.1. SOFTWARE REQUIREMENTS:

PYTHON 3.6

SURPRISE

SCIKITSURPRISE

PANDAS

MATPLOTLIB

JUPYTER NOTEBOOK

ANY OS WHICH SUPPORTS ABOVE MENTIONED REQUIREMENTS

## 3.2. HARDWARE REQUIREMENTS:

A standalone computer having at least 8 GB RAM, 2 Core Processor with 2.40 GHz Clock Speed

Recommended Hardware: 16 GB RAM and 4 Cores Processor with 3.8 GHz Clock Speed

# PROJECT PLAN

| Time | Work done |
|---|---|
| July 1st 3 weeks | Learning and analyzing how to do |
| July 4th week | Preprocessing for data and Finding way to implement all algorithm at once |
| August 1st week | Applying Half of listed algorithm |
| August 2nd week | Applying remaining Algorithm with graphs |
| September 2nd week | Learning of Model Error |
| September 3rd and 4th week | Modifying available parameters to reduce error for some of the models |
| October 1st and 2nd week | Finalizing and Implementation of model engine |

*Table 1: PROJECT PLAN.*

# IMPLEMENTATION DETAILS

## 5.1. Algorithms Preprocessing Part (For Data):

a. Removing all such columns which were irrelevant of the project requirement parameters such as Age and Location etc.
b. Converting all such Floating-point data into integer in order to avoid more training time.

## General Overview of all implemented Models:

### Singular Value Decomposition (SVD):

In linear algebra SVD is measures that is used for real or complex matrices as it is the generalization of Eigen decomposition of positive normal matrices which then can help identify the density of sparced data.

### SVDpp:

Just like the SVD, SVDpp also does the similar calculative projection decomposition but this is rather used as a parameter evaluator or specifically for evaluation of recommendation engine.

### SlopeOne:

As this is not just Content Based Project recommendation, as it also uses the help of collaborative filtering, this SlopeOne algorithm tries to find at least one related, relevant item from the dataset.

### Non-Negative Matrix Factorization:

For some math problem where it is not solvable completely in this case not all data point may be accurate while taking in as datapoint, this makes sure that the resulting matrices from such big dataset stays inspectable.

### Normal Predictor:

Algorithm predicting a random rating based on the distribution of the training set, which is assumed to be normal. Algorithm predicting the baseline estimate for given user and item.

### KNNBaseline:

A basic collaborative filtering algorithm taking into account a **baseline** rating.

### KNNBasic:

The regular K – Nearest Neighbor to find out relatively close datapoints from the data which would allow us to find the most accurate class of prediction of the project.

### KNNwithMeans:

Number of clusters identified as the basic class of the rating to represent the mass – the trend of rating to a particular rating.

### KNNWithZScore:

**Z-values** and k-nearest neighbors with a good one-dimensional hashing function the **KNN** search becomes simple: One hashes the training set and sorts the one-dimensional representation of the training set.

### BaseLineOnly:

A baseline provides a point of comparison for the more advanced methods that you evaluate later and in this case comparison with other projects and storing all the best project from individual fields and later providing one of them as best for the result.

### CoClustering:

Set of clustering techniques which allows us to find bimodality of real world data as it is often bimodal, that is to say created by a joint interaction between two types of entities.

## 5.2. Code Snapshots

Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
```

Reading CSV files and removing columns which are not required for the model training.

```
user = pd.read_csv('D:/GNU_Submission/SEMESTER-7/BDAD/Users.csv', sep=';', error_bad_lines=False, encoding="latin-1")
user.columns = ['UserID', 'Location', 'Age']
rating = pd.read_csv('D:/GNU_Submission/SEMESTER-7/BDAD/Ratings.csv', sep=';', error_bad_lines=False, encoding="latin-1")
rating.columns = ['UserID', 'ProjectID', 'ProjectRating']
df = pd.merge(user, rating, on='UserID', how='inner')
df.drop(['Location', 'Age'], axis=1, inplace=True)
df.head()
```

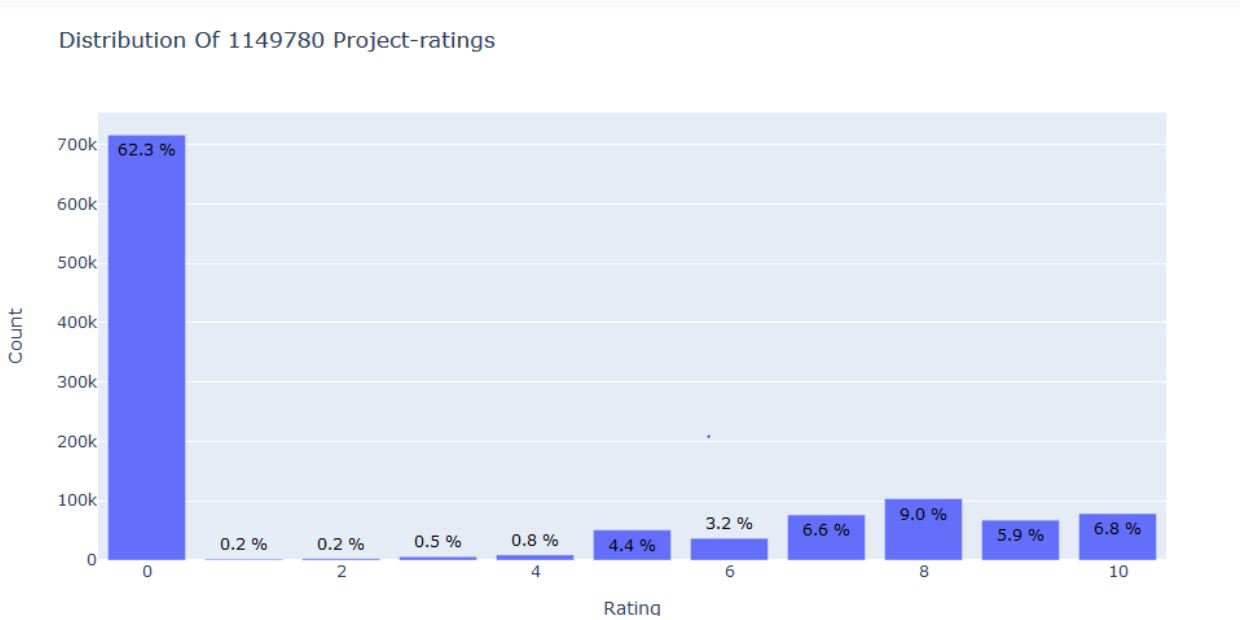
Output of the top 5 (head) data from the modified dataset.

| | UserID | ProjectID | ProjectRating |
|---|---|---|---|
| 0 | 2 | 0195153448 | 0 |
| 1 | 7 | 034542252 | 0 |
| 2 | 8 | 0002005018 | 5 |
| 3 | 8 | 0060973129 | 0 |
| 4 | 8 | 0374157065 | 0 |

Creating a bar graph from the rating values and number (count) of ratings of entire dataset and plotting it in 2D bar graph.

```
from plotly.offline import init_notebook_mode, plot, iplot
import plotly.graph_objs as go
init_notebook_mode(connected=False)

data = df['ProjectRating'].value_counts().sort_index(ascending=False)
trace = go.Bar(x = data.index,
               text = ['{:.1f} %'.format(val) for val in (data.values / df.shape[0] * 100)],
               textposition = 'auto',
               textfont = dict(color = '#000000'),
               y = data.values,
               )
# Create layout
layout = dict(title = 'Distribution Of {} Project-ratings'.format(df.shape[0]),
              xaxis = dict(title = 'Rating'),
              yaxis = dict(title = 'Count'))
# Create plot
fig = go.Figure(data=[trace], layout=layout)
iplot(fig)
```

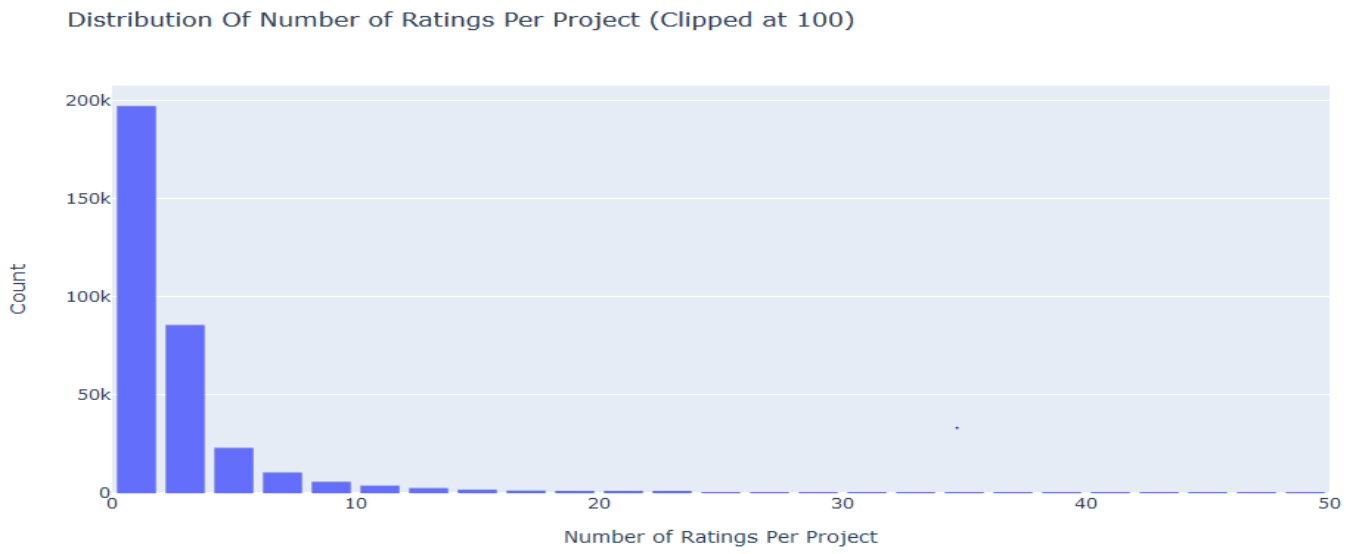
Output of the bar graph generated.




Creating a histogram for distribution of number of ratings per project.

```
# Number of ratings per Project
data = df.groupby('ProjectID')['ProjectRating'].count().clip(upper=50)

# Create trace
trace = go.Histogram(x = data.values,
                     name = 'Ratings',
                     xbins = dict(start = 0,
                                  end = 50,
                                  size = 2))
# Create layout
layout = go.Layout(title = 'Distribution Of Number of Ratings Per Project (Clipped at 100)',
                   xaxis = dict(title = 'Number of Ratings Per Project'),
                   yaxis = dict(title = 'Count'),
                   bargap = 0.2)

# Create plot
fig = go.Figure(data=[trace], layout=layout)
iplot(fig)
```

Histogram showing the number of ratings per projects.




Listing the projects with the highest ratings observed grouped by "ProjectID" and "ProjectRating".

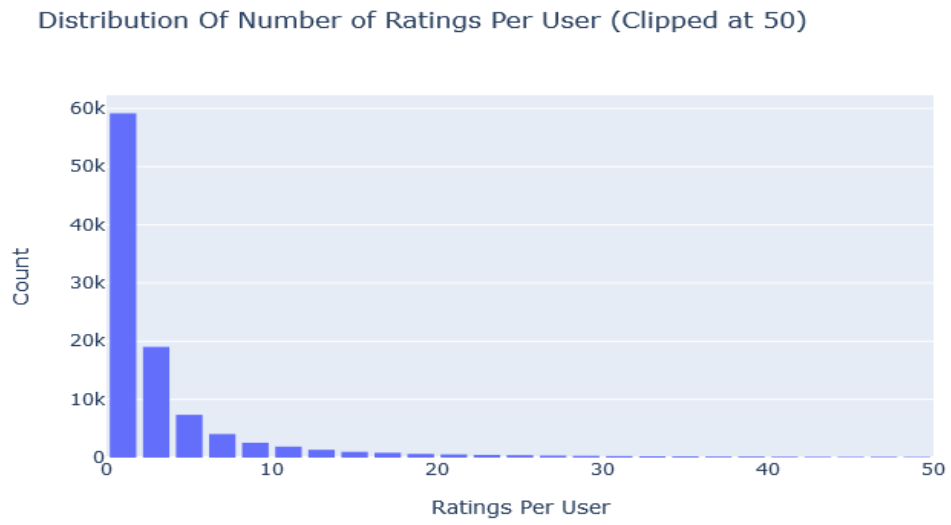
```
df.groupby('ProjectID')['ProjectRating'].count().reset_index().sort_values('ProjectRating', ascending=False)[:10]
```

| | ProjectID | ProjectRating |
|---|---|---|
| **247408** | 0971880107 | 2502 |
| **47371** | 0316666343 | 1295 |
| **83359** | 0385504209 | 883 |
| **9637** | 0060928336 | 732 |
| **41007** | 0312195516 | 723 |
| **101670** | 044023722X | 647 |
| **166705** | 0679781587 | 639 |
| **28153** | 0142001740 | 615 |
| **166434** | 067976402X | 614 |
| **153620** | 0671027360 | 586 |

Creating a histogram showing the number of ratings per user grouped by “UserID” and “ProjectRating” columns.

```
# Number of ratings per user
data = df.groupby('UserID')['ProjectRating'].count().clip(upper=50)

# Create trace
trace = go.Histogram(x = data.values,
                     name = 'Ratings',
                     xbins = dict(start = 0,
                                  end = 50,
                                  size = 2))
# Create layout
layout = go.Layout(title = 'Distribution Of Number of Ratings Per User (Clipped at 50)',
                   xaxis = dict(title = 'Ratings Per User'),
                   yaxis = dict(title = 'Count'),
                   bargap = 0.2)

# Create plot
fig = go.Figure(data=[trace], layout=layout)
iplot(fig)
```

Histogram showing output of the distribution of number of ratings per user.



Listing the users which gave the highest number of project ratings.

```
df.groupby('UserID')['ProjectRating'].count().reset_index().sort_values('ProjectRating', ascending=False)[:10]
```

| | UserID | ProjectRating |
|---|---|---|
| **4213** | 11676 | 13602 |
| **74815** | 198711 | 7550 |
| **58113** | 153662 | 6109 |
| **37356** | 98391 | 5891 |
| **13576** | 35859 | 5850 |
| **80185** | 212898 | 4785 |
| **105111** | 278418 | 4533 |
| **28884** | 76352 | 3367 |
| **42037** | 110973 | 3100 |
| **88584** | 235105 | 3067 |

Filtering the data based on min_project_ratings and min_users_ratings based on some minimum criteria.

```
min_project_ratings = 50
filter_projects = df['ProjectID'].value_counts() > min_project_ratings
filter_projects = filter_projects[filter_projects].index.tolist()

min_user_ratings = 50
filter_users = df['UserID'].value_counts() > min_user_ratings
filter_users = filter_users[filter_users].index.tolist()

df_new = df[(df['ProjectID'].isin(filter_projects)) & (df['UserID'].isin(filter_users))]
print('The original data frame shape:\t{}'.format(df.shape))
print('The new data frame shape:\t{}'.format(df_new.shape))
```

```
The original data frame shape:  (1149780, 3)
The new data frame shape:       (140516, 3)
```

Loading the necessary modules from surprise library.

```
from surprise import accuracy,Reader,Dataset,SVD,SVDpp,SlopeOne,NMF,NormalPredictor
from surprise import KNNBaseline,KNNBasic,KNNWithMeans,KNNWithZScore,BaselineOnly,CoClustering
from surprise.model_selection import train_test_split,cross_validate
reader = Reader(rating_scale=(0, 9))
data = Dataset.load_from_df(df_new[['UserID', 'ProjectID', 'ProjectRating']], reader)
```

Iterating through multiple algorithms to obtained the “Root Mean Square Error” as a measurement for the performance.

```
benchmark = []
# Iterate over all algorithms
#for algorithm in [SVD(), SVDpp(), SlopeOne(), NMF(), NormalPredictor(), KNNBaseline(), KNNBasic(), KNNWithMeans(),
#KNNWithZScore(), BaselineOnly(), CoClustering()]:
for algorithm in [SVD(),SVDpp(),CoClustering(),BaselineOnly(),SlopeOne(),NormalPredictor(),KNNBaseline(),KNNBasic(),
                  KNNWithMeans(),KNNWithZScore()]:
    # Perform cross validation
    results = cross_validate(algorithm, data, measures=['RMSE'], cv=3, verbose=False)

    # Get results & append algorithm name
    tmp = pd.DataFrame.from_dict(results).mean(axis=0)
    tmp = tmp.append(pd.Series([str(algorithm).split(' ')[0].split('.')[-1]], index=['Algorithm']))
    benchmark.append(tmp)

pd.DataFrame(benchmark).set_index('Algorithm').sort_values('test_rmse')
```

Output:

```
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
```

| Algorithm | test_rmse | fit_time | test_time |
|---|---|---|---|
| BaselineOnly | 3.379444 | 0.314476 | 0.312300 |
| CoClustering | 3.472776 | 2.980235 | 0.351952 |
| SlopeOne | 3.473571 | 0.838347 | 4.561420 |
| KNNWithMeans | 3.486691 | 0.874760 | 6.606373 |
| KNNBaseline | 3.494022 | 1.152879 | 7.815291 |
| KNNWithZScore | 3.508245 | 1.124273 | 7.290777 |
| SVD | 3.544947 | 7.325822 | 0.482119 |
| KNNBasic | 3.724929 | 0.830834 | 5.705274 |
| SVDpp | 3.785350 | 148.169044 | 5.359945 |
| NormalPredictor | 4.671221 | 0.187428 | 0.380129 |

Using Alternating Least Square to measure “rmse” and choosing the model with the least error, that is the “BaseLineOnly” algorithm.

```
print('Using ALS')
bsl_options = {'method': 'als',
               'n_epochs': 5,
               'reg_u': 12,
               'reg_i': 5
               }
algo = BaselineOnly(bsl_options=bsl_options)
cross_validate(algo, data, measures=['RMSE'], cv=3, verbose=False)
trainset, testset = train_test_split(data, test_size=0.25)
algo = BaselineOnly(bsl_options=bsl_options)
predictions = algo.fit(trainset).test(testset)
accuracy.rmse(predictions)
```

Output:

```
Using ALS
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
RMSE: 3.3655

3.3655317980106347
```

Enlisting best and worst 10 predictions done by the algorithm.

```
def get_Iu(uid):
    try:
        return len(trainset.ur[trainset.to_inner_uid(uid)])
    except ValueError: # user was not part of the trainset
        return 0

def get_Ui(iid):
    try:
        return len(trainset.ir[trainset.to_inner_iid(iid)])
    except ValueError:
        return 0

df = pd.DataFrame(predictions, columns=['uid', 'iid', 'rui', 'est', 'details'])
df['Iu'] = df.uid.apply(get_Iu)
df['Ui'] = df.iid.apply(get_Ui)
df['err'] = abs(df.est - df.rui)
best_predictions = df.sort_values(by='err')[:10]
worst_predictions = df.sort_values(by='err')[-10:]

print(best_predictions)
print(worst_predictions)
```

Output:

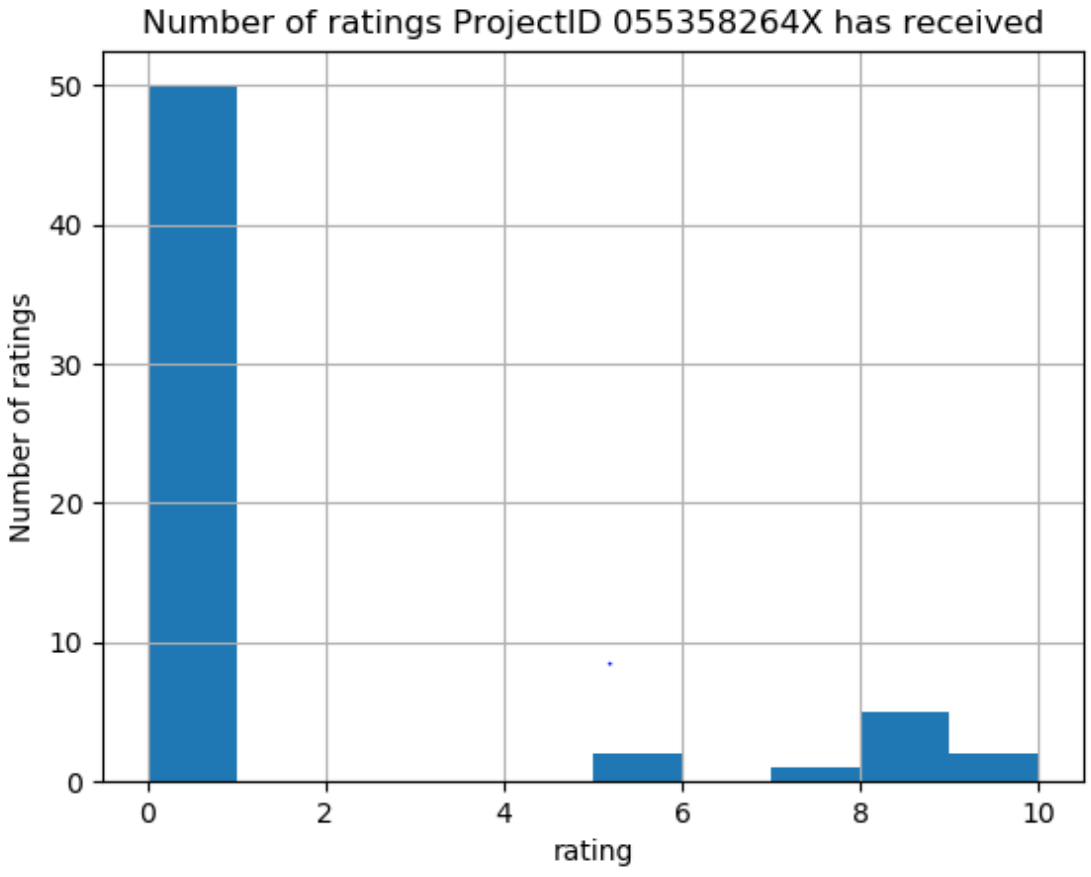
```
         uid        iid  rui  est                    details   Iu  Ui  err
24262  124942  0451160533  0.0  0.0  {'was_impossible': False}  173  25  0.0
26083   76352  0553566032  0.0  0.0  {'was_impossible': False}  510  31  0.0
22233   24921  0451141083  0.0  0.0  {'was_impossible': False}   92  34  0.0
26061   55492  0671741195  0.0  0.0  {'was_impossible': False}  367  54  0.0
10878  203968  0440211891  0.0  0.0  {'was_impossible': False}   82  43  0.0
1929   142093  055357227X  0.0  0.0  {'was_impossible': False}   58  35  0.0
16055  198711  042516098X  0.0  0.0  {'was_impossible': False}  357  84  0.0
31898  224435  0345409329  0.0  0.0  {'was_impossible': False}   72  30  0.0
31900  131046  0425147363  0.0  0.0  {'was_impossible': False}  134  67  0.0
31907  234623  0440237262  0.0  0.0  {'was_impossible': False}  232  32  0.0
         uid        iid   rui       est                    details   Iu  Ui
1946   14521  0553269631  10.0  0.109868  {'was_impossible': False}  171  31
29274  26544  055358264X  10.0  0.102565  {'was_impossible': False}  205  38
28083  20115  0679412956  10.0  0.055568  {'was_impossible': False}   50  31
14705  251422  0684801469  10.0  0.037907  {'was_impossible': False}  114  22
19307  161752  1551668912  10.0  0.000000  {'was_impossible': False}   58  36
21694   26544  0515128600  10.0  0.000000  {'was_impossible': False}  205  32
7140    81045  0515120278  10.0  0.000000  {'was_impossible': False}   47  27
10593  205735  0373825013  10.0  0.000000  {'was_impossible': False}   76  66
4088   238120  0385413041  10.0  0.000000  {'was_impossible': False}  332  30
18528  200674  0451160533  10.0  0.000000  {'was_impossible': False}  154  25

           err
1946    9.890132
29274   9.897435
28083   9.944432
14705   9.962093
19307  10.000000
21694  10.000000
7140   10.000000
10593  10.000000
4088   10.000000
18528  10.000000
```

# TESTING

Plotting the graph of Number of Ratings and Number of ratings for the given ProjectID.

```
import matplotlib.pyplot as plt
%matplotlib notebook
df_new.loc[df_new['ProjectID'] == '055358264X']['ProjectRating'].hist()
plt.xlabel('rating')
plt.ylabel('Number of ratings')
plt.title('Number of ratings ProjectID 055358264X has received')
plt.show()
```

Output:



# CONCLUSION AND FUTURE WORK

From the models we applied above for dataset, which contained the data from different users and provided that the data was almost ready to use yet with minor adjustments for fitting it to the model, it resulted in almost expected outcome of predication of the project which should be recommended to the particular user.

For future scope, there can be a GUI based app which takes interest of user as an input and then applying it as parameter to the model to find the outcome. Also, by applying sentimental analysis as quick review, the classification of the project definition allotment can be manipulated up to certain point, thus, it can be one of the way to modify the current model.

# REFERENCES


1) https://en.wikipedia.org
2) https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1
3) https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
4) http://surpriselib.com/