Research Project 1
Story Points Estimation

Adham Mohamed
ID:20192945
Promoter: Serge Demeyer
Mentor: Brent van Bladel
(Dated: May 21, 2020)

Abstract: Effort estimation for the issues in big projects is very important to manage the agile methods between the team, our datasets extracted from JIRA are imbalanced. in this approach, we present ways for handling the imbalanced datasets and experiment with a new classifier called XGBoost after it has proven its ability to deal with imbalanced classes [9].

I. INTRODUCTION

In this research project, we will describe what are the story points referring to and how can we use it in the agile software development estimation. also, we will present the issues which extracted from JIRA.

JIRA is known as issue tracking system, this issues may be representing as technical task, software bug or anything that requires to maintain from developers, JIRA contains of some fields to represent the issue, this fields may be string such as (title and Description) or numerical such as (Story points), these fields represent to the developers what the type of the issue, who report it and also a textual information that describes the issue issue[4].

The Story Point is defined as an estimation metric used by the agile teams to estimate what the complexity of the issue is and what is the estimation effort requires to complete the task. each team has its story point criteria such as the number of story points and each number represent a complexity level[10].there are a mental process that each team may go through when they decided to estimate a new issue this method depending on 'planning poker' approach [6]:

1. each developer looks for similar issues that already solved it in the past based on the previous experience
2. each developer gives the number of points based on the similar tasks have been done in the past
3. If the developers can not detect the best number of story points based on the previous tasks, they discuss until to determine the best story points

In this work, we built upon previous research by investigating two ways to improve the existing state-of-the-art classifier. Our First approach applies different techniques to the existing state-of-the-art classifier, such as penalties, to improve it. Our second approach attempts to outperform the classic machine learning classifier by using XGBoost Classifier.

For this reason, we pursue the following research questions:

(RQ1): Can the classic machine learning classifiers be improved by Handling imbalanced data?

(RQ2): What performance can be achieved by using Story Points Estimation Method Based on XGBoost classifier?

II. DATASETS

in this work, we will use the datasets which are collected by Hoa Khanh et al. [2].these datasets have been collected from several open-source projects that use JIRA and have story points estimation such as

- (TISTUD), Appcelerator-studio is IDE.
- (XD) Spring XD is a unified, distributed, and extensible system for data ingestion, real-time analytics, batch processing, and data export.
- (APSTUD) Aptana Studio is a web development IDE.
- (MULE) Mule is a lightweight Java-based enterprise service bus, integration platform and also allows the developers to connect more than one application easily and quickly.
- (MESOS) Apache Mesos is a cluster manager.
- (TIMOB) Titanium SDK/command-line interface(CLI) is an SDK and a Node.js based command-line tool for managing, building, and deploying Titanium projects.

in this research project, we will use (TISTUD) datasets for showing the results after implementing our approach as it is the baggiest datasets. it contains 2919 issues after cleaning the data from nan values we will use 2876 issues for classification.

A. Imbalanced Datasets

We all wish to manipulate with a balanced dataset that our algorithm can learn easily. Nevertheless, different cases may the data be highly imbalanced because of the wrong of the class ratio. this kind of dataset has a minority class that contains a small number of samples and a Majority class that Contains the largest share of samples.

there are several techniques to handle imbalanced Datasets:
our research group has investigated one of these techniques called over-sampling with SMOTE, it is an oversampling algorithm that deals with the minority class for creating its synthetic data. the goal from this method is balancing class distribution by randomly sampling from a group of the minority class observations[8].

in this research project we approach two other techniques for handling imbalanced dataset:

- Penalize Algorithms (Cost-Sensitive Learning)
- XGBoost classifier

talking about cost-sensitive learning technique is penalizing mistakes in the minority classes by defining the miss-classification costs among classes[8]. in this approach we used the $class - wight$ parameter for the Linear Support Vector Machine algorithm(SVM), class wights altering the loss function by supporting more wights for the minority class, or giving fewer wights for the majority class, With that, we will have the balanced classes. for using the $class - wight$. The scikit-learn library for machine learning in Python we can use SVM algorithm and set the value of the $class - wight$=$balanced$.

the second approach we'll consider is using tree ensembles based algorithms such as (XGboos)classifier, it stands for eXtreme Gradient Boosting. we will figure out its capability for handling miss-classification costs between imbalanced classes. The power of XGBoost lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

III. FEATURES

our datasets are consisting of textual data as features ($issuekey$,$title$,$description$) and numerical data as classes ($storypoints$) , after cleaning the data from the missing values in the ($description$) column. when we plotted the story point column for discovering the distribution of the data as shown in the (fig1) we can see that there are eleven classes distributed in our data and we can see also how many items per class as shown in fig 2, now we can see the highly imbalanced dataset.

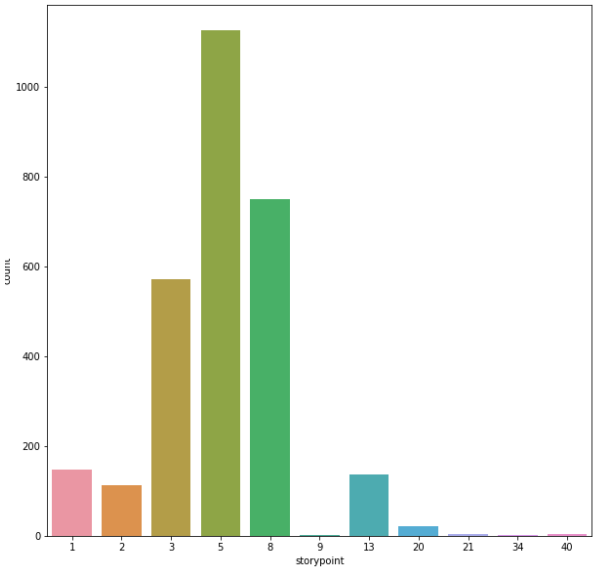

FIG. 1: original classes

.

storypoint	
1	148
2	112
3	571
5	1126
8	751
9	1
13	137
20	22
21	3
34	1
40	4

FIG. 2: the amount of items per point

.

we talked in the previous section about how can we handle the imbalanced datasets. However, we will start by re-clustering those classes into three classes (easy)to prevent highly imbalanced data.

1. Issues with story points less than or equal to 2 grouped as easy level.
2. Issues with story points between more than 2 and less than or equal to 5 groups as a medium level.
3. Issues with story points more than 5 grouped as complex levels.

with this operation, we can't assume that the data now is balanced But it got a little better. we can see the

difference after re-grouping as shown in fig3. as a result, we have 3 lapels (0,1,2) have examples in the order (260,1697,919)

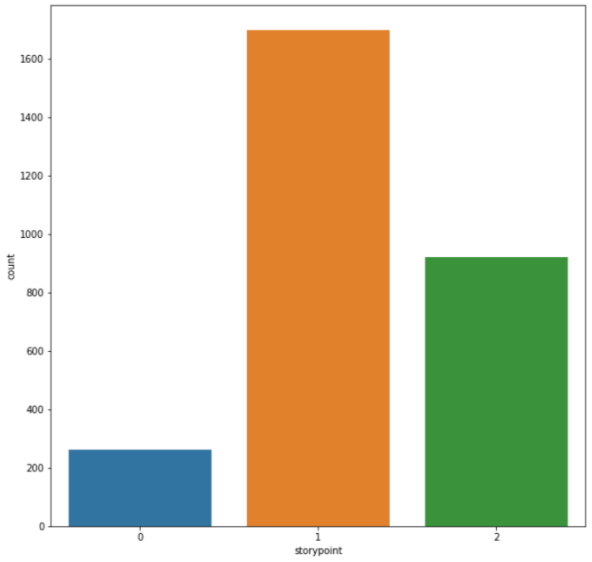

FIG. 3: the new grouping of classes

.

now we can easily select the main features for the classifier. for the (*issuekey*) column we see that it is not important because it has not important values it has the name of the project and the number of issues, so we will drop this column. for the (*description* and *title*) columns they contain important textual data that describe the issues, so we decide to concatenate those two columns under a new column called *titDescription*. and create new column *titDescriptiontLEN* for counting the length of each row in the *titDescription* column to can see what is the relation between the length of textual description and its story points as shown in the fig 4 the Relation between Story Points and *titDescriptiontLEN*

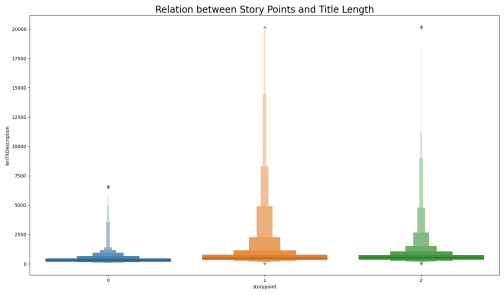

FIG. 4: Relation between Story Points and Title Length

.

Consequently we chose the *titDescription* column as a feature and the *storypoints* column as a lapel for the classifier.

IV. IMPLEMENTATION

After selecting the main features for our classifier, we began to manipulate the textual data to prepare the data for the machine learning Algorithms we have to clean the data from any not important words and characters and also numbers and finally convert the cleaned data to numerical to be fit for the algorithms. we used Natural Language Processing techniques for cleaning the textual data. we used for that the python library $NLTK$:

1. stop word removal for remove all the punctuation, characters and around 500 common English words such as $(the, a, at, as, ..)$
2. stemming the rest texts for removing all the Words extension such as (processing) after stem (process)

after those processes, we have to convert each row to a list of numerical data, we used $TF - IDF$ algorithm is widely known as a $termfrequency - inversedocumentfrequency$ [5] for the term frequency TF of each word by counting how many times the word appears in the single row and then IDF for each word and decrease its weight if the word occurs in many rows.Equation (1) shows the relation between $TF-IDF$

$$tfidf(t, d, D) = tf(t, d).idf(t, D) \quad (1)$$

V. METRICS

After Selecting the features that need this work we should define the metrics that will help us for evaluating our model. we used a python library called $scikit-learn$ is usually uses in machine learning scope.

Before evaluating our model by the following metrics, firstly we used the Cross-Validation technique for splitting all the dataset into k folds and repeat it many times for our problem we will repeat it 10 times to avoid overfitting. and takes one fold randomly for each repeat as a test fold we used $StratifiedKFold$ function for this operation, then at the end of processing the $cross_val_score$ takes the average of each fold together for a final score.

Confusion matrix is one of the most easiest metrics which used for evaluating classification problems. Actually confusion matrix is not used for measuring the performance However, all the performance metrics are based on it. As shown in the (table I)[3]. the confusion matrix consist of four values[7]:

1. (TP) True positive numbers
2. (TN) True Negative numbers
3. (FP) False positive numbers
4. (FN) False Negative numbers

	Actual Positive	Actual Negative
predicted positive	TP	FP
predicted Negative	FN	TN

TABLE I: Confusion Matrix.

Accuracy as shown in Equation 2 is the ratio between the True predicted values to the total values.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

Precision as shown in Equation 3 is the ratio between True predicted positive values to the total predicted positive Values

$$Precision = \frac{\text{TP}}{\text{TP + FP}} \tag{3}$$

Recall as shown in Equation 4 is the ratio of True predicted positive values to the all values in actual class - yes

$$Recall = \frac{\text{TP}}{\text{TP + FN}} \tag{4}$$

F1-Score as shown in Equation 5 is the combination of Precision and Recall to get the weighted average of Precision and Recall.

$$F - score = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}} \tag{5}$$

F1-score takes the false positives and the false negatives into account, in our problem it is usually more useful than accuracy, especially if we have an imbalanced class distribution.

VI. RESULTS

In this section, we will figure out the baseline classifiers Result which we used for this problem and our approaches to improve the result.
Our research group has investigated this problem before and they agree that Linear Support Vector Machine Algorithm(SVM) is the state-of-the-art. but we should take in account that we use in this approach different dataset, we use the datasets which are collected by Hoa Khanh et al [2], for our research group they used the same data sets that are used by Porru et al[10].so maybe we find a little bit different in the comparison of the results with the state-of-the-art.

Before starting to figure out the results for our classifiers We would like to share the result for the default SVM with the original classes to show the difference between our approach for regrouping the classes and the original case. we can see the confusion matrix and the classification report for run the SVM Classifier on the original classes as shown in fig 5 we can see easily the bias between the classes with F1-score=40,12% and Accuracy =41,82%

```
[[ 41   0  23  41  42   0   0   0   0   1   0]
 [  0  13  26  54  17   0   2   0   0   0   0]
 [  6   5 190 287  82   0   1   0   0   0   0]
 [ 11   5 187 650 266   0   7   0   0   0   0]
 [  3   3  76 359 300   0  10   0   0   0   0]
 [  0   0   1   0   0   0   0   0   0   0   0]
 [  4   2   9  45  68   0   9   0   0   0   0]
 [  1   0   0   5  15   0   0   0   0   0   1]
 [  0   0   2   0   0   0   1   0   0   0   0]
 [  1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   1   1   0   2   0   0   0   0]]
              precision    recall  f1-score   support

           1       0.61      0.28      0.38       148
           2       0.46      0.12      0.19       112
           3       0.37      0.33      0.35       571
           5       0.45      0.58      0.51      1126
           8       0.38      0.40      0.39       751
           9       0.00      0.00      0.00         1
          13       0.28      0.07      0.11       137
          20       0.00      0.00      0.00        22
          21       0.00      0.00      0.00         3
          34       0.00      0.00      0.00         1
          40       0.00      0.00      0.00         4

    accuracy                           0.42      2876
   macro avg       0.23      0.16      0.17      2876
weighted avg       0.41      0.42      0.40      2876
```

FIG. 5: Confusion matrix and Classification report for constant classifier

.

A. Support Vector Machines(SVM)

The SVM classifier performs the classification process by finding the hyperplane that maximizes the margin Among the classes.
After applying our approach for regrouping the classes we figured out the improvement of the scores on the SVM classifier without adding a $class - wight$ parameter to the classifier, the accuracy becomes 62.83% and the F1-score becomes 60.99% as shown in table III. Given the confusion matrix as shown in table II the class 1 has the largest share of examples with Precision 67% and recall 97%. for those results, we can approve that our step for regrouping the classes it was a good step for handling the imbalanced dataset a little bit.

	class 0	class 1	class 2	sum
class 0	**54**	150	56	260
class 1	31	**1349**	317	1697
class 2	12	563	**355**	919
sum	97	2062	728	2876

TABLE II: Confusion Matrix for default SVM.

Accuracy	0.6283
Precision	0.6140
Recall	0.6283
F1-score	0.6099

TABLE III: Evaluation metrics for default SVM algorithm

After adding our approach for cost-sensitive learning ($class - wight =' balanced'$). we figured out a little bit

improvement with the TP values in class 0 and class 1 with total accuracy 65.23% and F1-score 62.26%, almost the score has increased with 2% than the default SVM.

	class 0	class 1	class 2	sum
class 0	**57**	162	41	260
class 1	21	**1473**	203	1697
class 2	17	556	**346**	919
sum	95	2191	590	2876

TABLE IV: Confusion Matrix for weighted SVM.

Accuracy	0.6523
Precision	0.6383
Recall	0.6522
F1-score	0.6226

TABLE V: Evaluation metrics for weighted SVM algorithm

refer to our first research question:
(RQ1): Can the classic machine learning classifiers be improved by Handling imbalanced data?
An SVM classifier with a penalized algorithm for story point estimation can achieve a little bit improvement with increasing accuracy with 2% than the default SVM and the F1-score with 1%.

B. K Nearest Neighbors

k-nearest neighbors algorithm is a simple algorithm used for classification problems, it depends on the similarity measures such as distance function for selecting the nearest K.
after applying the K-Nearest Neighbors classifier we got the results as shown in tables(VI,VII), the accuracy is 60.74% and the F1-score is 57.74% which is better than the constant classifier, the weighted SVM results increasing with almost 5% accuracy compared to The KNN algorithm, till now the best result for weighted SVM classifier.

	class 0	class 1	class 2	sum
class 0	**51**	169	40	260
class 1	46	**1344**	307	1697
class 2	18	568	**333**	919
sum	115	2081	680	2876

TABLE VI: confusion matrix for KNN algorithm

Accuracy	0.6074
Precision	0.5776
Recall	0.6008
F1-score	0.5774

TABLE VII: Evaluation metrics for KNN algorithm

C. Logistic Regression

the next classifier we have used is the logistic regression algorithm, TablesVIII,IX show the results for this algorithm, the accuracy is 64% and the F1-score is 60.67%, we can see that the accuracy is better than the KNN classifier with A difference of almost 3% and the F1-score increasing with a difference of 2%.

	class 0	class 1	class 2	sum
class 0	**23**	185	52	260
class 1	2	**1480**	215	1697
class 2	1	563	**355**	919
sum	26	2228	622	2876

TABLE VIII: Confusion Matrix for LR.

Accuracy	0.6460
Precision	0.6543
Recall	0.6460
F1-score	0.6067

TABLE IX: Evaluation metrics for LR algorithm

D. XGBoost Classifier

Extreme Gradient Boosting (**XGBoost**) is an implementation of gradient boosted decision trees based on ensemble Machine Learning algorithm designed for speed and performance, it was developed as a research project at the University of Washington[1] Tables X,XI show the result of the confusion matrix and the Evaluation metrics, the accuracy is 65.68% and the F1-score is 63.29% which there are slightly increase of the performance with almost 1% in accuracy and F1-score compared to the weighted SVM.

so we can approve that the best result for the accuracy and the F1-score for our problem is the XGboosted Classifier.
According to our second Research Question 2:
(RQ2):What performance can be achieved by using Story Points Estimation Method Based on

	class 0	class 1	class 2	sum
class 0	**47**	153	60	260
class 1	17	**1426**	254	1697
class 2	4	499	**416**	919
sum	68	2078	730	2876

TABLE X: Confusion Matrix for XGBOOST

Accuracy	0.6568
Precision	0.6494
Recall	0.6568
F1-score	0.6329

TABLE XI: Evaluation metrics for XGBOOST algorithm

XGBoost classifier? We Have Investigated the XG-boosted classifier and we figured out that it has the best result and accieved slightly increase of performance compared to Weighted SVM Classifier.

E. SMOTE

in this section, we will present the SMOTE function based oversampling technique results which our research group has investigated before. we applied this technique in addition to our approaches. we would like to do A fair comparison. our research group proved that the SVM classifier is state-of-the-art. knowing that we used different datasets so the results will not the same but we just need to clarify the concept and the ability of each classifier for our approach.

Table XII shows the results for the previous classifier after using the SMOTE function. we can see that there are no difference between the results in default SVM and weighted SVM.
also, we can see in this dataset and after applying oversampling which both achieved an accuracy of 81.81% and F1-score of 81.32%. The results are also shown that the K-Nearest Neighbors classifier has the least score of the accuracy of 66% and F1-score 60%.
finally, we can prove the capability of the XGBoost classifier for achieves the best results at all for any comparison in this research project which we reach accuracy and F1-score of 83.59%.

VII. CONCLUSION

in this research project, we tried to investigate the story points estimation from JIRA with the classic machine learning classifiers by handling the imbalanced dataset.

we reached out to the result after regrouping the story points to just 3 story points (0 easy,1 medium, 2 complex), In this way, it allowed more samples to each class and it managed to reduce the imbalance to some extent.

the results show us that our approach for manipulating the miss-classification costs(penalize technique) works better in SVM algorithm by setting the $class - weight =' balanced$ Parameter that resulted in accuracy 65.23% and F1-score 62.26% compared to The Default SVM classifier which resulted in accuracy 62.83% and F1-score 60.99%.
on the other hand, our second research question shows us the ability of one of the ensemble Algorithms $XGboost$ classifier for manipulating with the imbalanced datasets and it also can achieve the best score for $TISTUDE$ dataset compared to the weighted SVM classifier with accuracy 65.68% and F1-score 63.29%.

After applying the Smote function for oversampling we figured out :

1. the performance of all classifiers are increased Significantly.
2. There are no effects on the results when we used the miss-classification costs (penalize technique) with the oversampling using Smote function, which they increased with the same score.
3. The XGBoost classifier has achieved the best score.

TABLE XII: SMOTE Results for classic machine learning classifiers.

Classifiers	Accuracy	Precision	Recall	F1-score
Default(SVM)	0.8181	0.8153	0.8181	0.8132
Wighted (SVM)	0.8181	0.8153	0.8181	0.8132
KNN	0.6611	0.7005	0.6611	0.6021
Logistic Regression	0.7762	0.7723	0.7762	0.7732
XGBoost	0.8359	0.8361	0.8359	0.8359

VIII. REFRENCES

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[2] M Choetkiertikul, H K Dam, T Tran, T T M Pham, A Ghose, and T Menzies. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, PP(99):1, 2018.

[3] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[4] John Fisher, D Koning, AP Ludwigsen, et al. Utilizing atlassian jira for large-scale software development management. In *14th International Conference on Accelerator & Large Experimental Physics Control Systems (ICALEPCS)*, 2013.

[5] Binyam Gebrekidan Gebre, Marcos Zampieri, Peter Wittenburg, and Tom Heskes. Improving native language identification with tf-idf weighting. In *the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA8)*, pages 216–223, 2013.

[6] James Grenning. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3:22–23, 2002.

[7] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[8] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[9] Satwik Mishra. Handling imbalanced data: Smote vs. random undersampling. *International Research Journal of Engineering and Technology (IRJET)*, 2017.

[10] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. Estimating story points from issue reports. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 1–10, 2016.