IT254: Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking

Sohanraj R - 191IT149
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: sohanrajr.191it149@nitk.edu.in

Harini Thirunavukkarasan - 191IT221
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: harinithiru.191it221@nitk.edu.in

Gayathri Nisha - 191IT116
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: gayathrinisha.191it116@nitk.edu.in

Abstract—**Ranking web services is very helpful in choosing the best web service.In this paper, throughput and response time values are used to evaluate trust predictio and ultimately rank five different web services. AdaBoostM1 is the binary classifier used on a benchmark web services dataset. The trust score (TS) measuring method is adopted where the confusion matrix is used to determine the trust scores of all the web services. Trust prediction is calculated using the predicted values from the model stored in an array. A graphical user interface is also created using html,css and flask which allows the user to input five .xlsx files containing the throughput and response time values of 100 users for 5 web services.Once the upload is complete, the ranking is displayed to the user.Accurate prediction of trusted and untrusted users in web services invocation has helped select the best web service from a pool of similar web services.**

keywords-AdaBoostM1, Web services, Fuzzy rules, Trust prediction, Cross-validation, Confusion matrix, Flask

I. INTRODUCTION

Trustworthiness of a web service plays a huge role in its likelihood to be chosen over other web services. In our proposed fuzzy-based users' trust prediction approach, we are taking the end-users values of quality attributes (such as response time and throughput) and then ranking web services by calculating their trust scores.A trusted web service is one that is reliable as well as provides high throughput and low response time.The main objective of trust prediction is to calculate the users' trust in the invoked web services. Then, the calculated users' trust is used to rank web services from a pool of web services accessed by the same users of various regions.

Web services consumers will rate the same web service differently in terms of the QoS properties. For example, users 'p' and 'q' might rate WS1 as having high throughput and low response time but user 'r' might rate the same service as having low performance (throughput) and high response time. This is because of the subjective perception. Users 'p', and 'q' would want their invoked service to respond within 1 or 2 seconds. However, user 'r' may not have a high standard and would be okay with a web service that responds within 30 seconds. We can specify that different users have provided their trust values by differently rating web services.

AdaBoostM1 is the binary classifier used.It is used to boost the performance of decision trees on binary classification problems. It is a model that achieves accuracy just above random chance on a classification problem. Users' reputation is determined by clustering the information obtained from similar users to identify the clusters of users and invoked web services. Web service trustworthiness is dependent on users, and it may be maintained in the inappropriate clusters. As a result, this inappropriate clustering affects the trustworthiness of certain web services. To tackle this issue, we have proposed an approach which uses the confusion matrix measures.A confusion matrix is used to describe the performance of a classifier on a set of test data whose true values are already known.It allows visualization of an algorithm's performance. For the binary classification, a confusion matrix is represented as 2*2 matrix. For a confusion matrix,four measures namely 'true positive' (TP),'true negative' (TN), 'false positive' (FP), and 'false negative'(FN), have been reported.

Flask is a popular python web framework used for developing web applications.It is an excellent framework for processing requests and it follows a minimalistic approach.A graphical user interface build with flask in the backend and html and css in the front end will help the users to easily upload files and get the ranking of the web services. The ultimate goal of ranking is to identify the web services with the high trust score of users and prioritize them for better future selection of web services.

II. LITERATURE SURVEY

Our base paper for the implementation of this project was Evaluating trust prediction and Confusion Matrix measures for

Web Services Ranking by Muhammed Hasnain and team. This study basically deals with developing a model to predict the trust percentage of a particular web service based on their Quality of Services, namely Throughput and Response time. The paper involves training of two models AdaboostM1 and J48 which includes the comparison of the performance of the two models for predicting the trust percentage of a website. They use three models each of AdaboostM1 and J48 tested using 5- fold,10-fold and 15-fold cross validation respectively.

Authors	Methodology	Advantages	Limitations
Mohammed Husnain, Muhammad Fermi Pasha, Imran Ghani,	Trust prediction and ranking of web services using Confusion matrix measures on models like AdaboostM1 nad J48	Trust score calculated is analogous to the accurate prediction of the trust instances.	It requires the true accurate values of the QoS pair provided by the fuzzy rules to compare it with the predicted result through the confusion matrix to find the trust score.
Dr.Ding	combined QoS prediction and estimation of customer satisfaction in their proposed approach known as CSTrust, which is used to release the customer satisfaction information on web services.	It is directed to find the trust score of mainly cloud based web services.	It cannot find trust scores for web services based on open standards such as XML, WSDL and SOAP.
Vivek Singh Bhadauria	Uploading, Processing and Downloading Files in Flask	It is used to learn how to handle files from server in Flask	-

III. OUTCOME OF LITERATURE SURVEY

The survey begins with the research in the background of connection between the Trust prediction using binary classification and Fuzzy rules which constitutes the basic part of this project. The paper involves training of two models AdaboostM1 and J48 which includes the comparison of the performance of the two models for predicting the trust percentage of a website. They use three models each of AdaboostM1 and J48 tested using 5- fold, 10-fold and 15-fold cross validation respectively. The values predicted by these models are analyzed with the accurate results using a confusion matrix measure. The different aspects of the confusion matrix is used to determine the percentage of trust.
We learnt how to implement AdaboostM1 using python and implement the UI using flask, Implementation of binary classification using fuzzy logic and using flask on how to provide interface to user for uploading a file and process file on server side

IV. MOTIVATION

Ranking of the web services helps users to choose the best services which have a high throughput value and less response time. This prevents confusion between similar web services and allows the user to choose the most trusted web service.This paper can also be very helpful for software architects and managers. Software architects can use the proposed approach to build better web services by using the trust features of the consumers. Web services managers can use the ranking of the web services based on their users' trust to improve the quality of web services

V. PROBLEM STATEMENT

Ranking the Web Services by their Trust Scores which are calculated using Response Time and Throughput feedback values given by different Users.

VI. OBJECTIVES

- To propose fuzzy rules to provide ground truth for training and evaluation of binary classifiers.
- To propose a trust prediction binary classification approach by using QoS attributes of web services.
- Analysis using confusion matrix.
- To develop an UI, Frontend and Backend based on HTML form, CSS, Python and Flask.
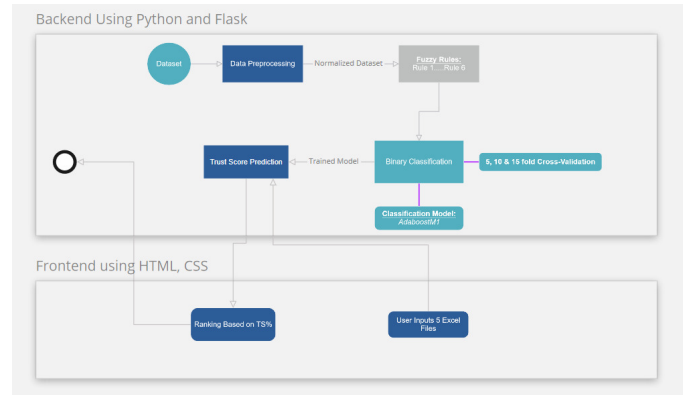
VII. PROPOSED WORK

A. *Proposed model*

Fig. 1: Flowchart

B. *Methodology*

1) Dataset: The quality WS-dream dataset was used to integrate the model. This dataset contains matrix measures of two Quality of Services namely Throughput and Response time of the web services. In addition to metric values each web service has web service Id, WSDL address, 'internet protocol' IP address, country, 'autonomous system'(AS), latitude, and longitude properties. The dataset consists of data from 339 users across the world.

2) Data Pre-processing: We have implemented a normalizing technique on the above dataset to improve the accuracy of the classifier on the numerical dataset. To normalize the data we used the classic min- maxim approach based on the formula

$$Zi = (Xi - Xmin)/(Xmax - Xmin)$$

where Xi is one element of the data metrics of one web service and Xmin and Xmax are minimum and maximum values in the particular data metrics. These normalized data were stored in excel files which were then used to train the binary classifier for the classification of web service instances. The min-max normalization method helps to preserve the association among the ordinal input data.

3) Fuzzy Rules: Fuzzy Rules are defined as a specific logic mapping from the process state to the relevant consequent state. We use the "IF...ELSE" rules along with connectors such as AND and OR to define the membership functions of the fuzzy set used in the rules. The format of the rule is such that

R: IF x is A and y is B THEN Z = f(x,y)

where A,B are the premise or fuzzy set and Z is the consequence and x,y are the crisp inputs. If the trust of a user in web services is represented by a linguistic variable, then a term set Ta can be of the following form:

Ta = Very high, high, medium, low, very low

Each linguistic term above given is associated with the fuzzy set defined on the domain [1, 0]. Very high can be associated with near to 1, and very low can be linked near to 0; high can be linked to 0.08; medium can be linked to 0.06, and low can be linked to 0.04. The fuzzy rule is constructed from various sources, such as the opinion of domain experts, knowledge engineering, and historical data analysis. We have constructed the Fuzzy rules with the help of logical operators to address the binary classification problem and the throughput(TP) and response time(RT) being the crisp inputs.

1) Rule 1: ((If TP > 0.06) OR (RT > 0 and ≤ 0.02)) then a user is trusted.
2) Rule 2: ((If TP > 0.03 and ≤ 0.06) OR (RT > 0.02 and ≤ 0.04)) then a user is trusted.
3) Rule 3: ((If TP > 0.02 and ≤ 0.03) and (RT > 0.04 and ≤ 0.06)) then a user is not trusted.
4) Rule 4: ((If TP > 0.01 and ≤ 0.02) and (RT > 0.06 and ≤ 0.08)) then a user is not trusted.
5) Rule 5: ((If TP ≤ 0.01) and (RT > 0.08)) then a user is not trusted.
6) Rule 6: ((If TP > 0.02 and ≤ 0.03) and (RT > 0.06 and ≤ 0.08)) then a user is not trusted.

4) Binary Classification: In this phase of our approach the binary classification of the web services is done. Basically we will be training a model on the web services dataset for classification.We have chosen AdaboostM1 as our model for classification. This model is trained on the web services dataset. Adaboost classifier is a boosting algorithm and is chosen for its production of high accuracy results. It creates a robust classifying model by focusing on the mis-classified records present in the past models.This technique gives value for every instances or records. Subsequently all the weights are set 1/n and this is referred in each cycle of the technique.

Rule No.	Translation	Decision
1	Very High OR Very Low	Trusted
2	High OR low	Trusted
3	Medium AND Medium	Untrusted
4	Low AND High	Untrusted
5	Very Low AND Very High	Untrusted
6	Medium AND High	Untrusted

Fig. 2: Linguistic terms

The combination of two distinct sorts boosting and decision tree is used to reduce the changes in the robust model

1) **AdaboostM1**: AdaboostM1 is a part of the most known boosting family which was implemented in WEKA. The classifier basically works on sequential traing of the models and each of its round has trained model. The misclassified records are identified and are used in a new training set which is used in training of the new model. In this project we have a need for a binary classification so we have implemented a c1 vs c0 classification.

Fig. 3: Algorithm

2) **Train-test split for training and testing**: The above AdaboostM1 model is trained on the web services dataset. The dataset in divide into two subsets in the ratio of 70:30, where 70 percent of the dataset is assigned to the training part whereas 30% for the testing part. This distribution was considered based on observation of the accuracy given by the model. The aim is to maximize the accuracy of the model and also make sure the model is not overfitted. Overfitting of the model causes a false increase in accuracy which will be fatal for the prediction. The testing part gave us an accuracy of 95.436% .

3) **Cross k-validation**: We used the cross k-validation technique to evaluate our proposed model. Three k-folds 5,10 and 15 were used to train the models. Here we will be basically creating three models where the first one is evaluated on a 5-fold CV, second one on a 10-fold Cv and the third one on a 15-fold CV. One of the advantages of using many k-folds in our implementation is to remove any biases or overfitting issues while training and testing. This minimizes the generalization of errors. For a 5-Fold CV the data is divided into K number of subsets, where K-1 subsets

are used for training and rest for testing. This process reports until all the data is used for testing and training. The similar process occurs for 10-fold and 15-fold as well while evaluating the model. This technique gave us the accuracy of 95.467%, 95.288% and 95.164% respectively.

4) **Trust Prediction**: To find the trust prediction of a given web service, we have basically passed the response time and throughputs of the particular service which was made available in the form of an excel sheet to the trained model for prediction. Each individual pair of Throughputs and Response time passed gives us a prediction that is either a zero or a one. This result is appended on to a dictionary and stored in it. This data is passed on to all the three models and the predicted results is stored on to separate dictionaries. The truth score is calculated by the equation

$$Truth - Score = (No.\_of\_prediction\_showing\_1) * 100/(No.\_of\_prediction\_showing\_0)$$

The trust score predictions computed from the three models are averaged to get an average trust score for the web service. This average trust score is used for ranking the web services. The higher the average trust score the higher the ranking.

5) **Flask, HTML, CSS**: The Front-end development or the client-side development is designed using the HTML and CSS. HTML forms were used to collect 5 Files from the user and store it in an "Uploads" folder, and an array of the predicted output is sent to the server side using Flask. Flask is a lightweight web app framework designed to make complex applications using python. In this project Flask has been used to provide an user interface for uploading the excel files, processing the excel files on the server side and to display the final ranking of the websites to the Users. Backend processing includes running the 5 excel sheets which are stored in the "Uploads folder" through the 3 k-fold binary classification models and getting the final average trust percentage score for each of the 5 files, which was resent to the 'Results' html page which was rendered. Once the process ends, the 5 excel sheets in the "Uploads" folder are discarded for user information security and to avoid any possible embedded threats.
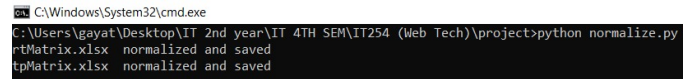
C. Proposed Modification

The base paper we referred to does not have a user interface so we have implemented a graphical user interface in our paper to ensure that the user-system interaction is as simple, fluid, intuitive and efficient as possible. HTML and CSS have been used for the frontend while python and flask have been used to do the backend work. The website is hosted at http://localhost:5000/user. This allows the users to enter response time and throughput values for the web service they desire to test for trust prediction. This UI can help the user navigate with ease without having to worry about the algorithm behind the ranking.
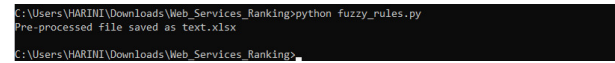
D. Work Done

1) Data Pre-processing: The dataset rtmatrix.xlsx and tp-matrix.xlsx collected from the WS-Dream is run through the normalize.py to pre-process the data. There are 2500 websites and 339 user feedback collected for each website in the response time and throughput sheets.
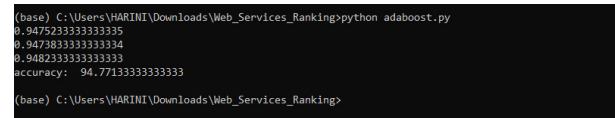

Fig. 4: Normalizing, Terminal Output

Following the data pre-processing, the sheets are run through fuzzy.py where each response time and throughput value is sent through the fuzzy rules and the classified as ["trusted":1, "untrusted":0] which is stored in the third column. The fused sheet is saved as text.xlsx.
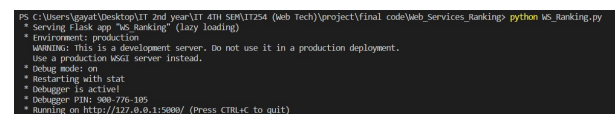

Fig. 5: Fuzzification, Terminal Output

2) Binary classification model: The final "text.xlsx" sheet which is the preprocessed dataset is used to create and train the adaboostM1 model. Following which the model is evaluated using the repeated stratified k-fold cross-validation methods where k in [5,10,15].

Fig. 6: AdaboostM1 Model training and k-fold Cross-Validation

3) Trust Score Prediction: A simple HTML page "index.html" is designed which includes a form asking the user for 5 inputs of type "files" and a submit button is created and the template is rendered once the server is initiated.
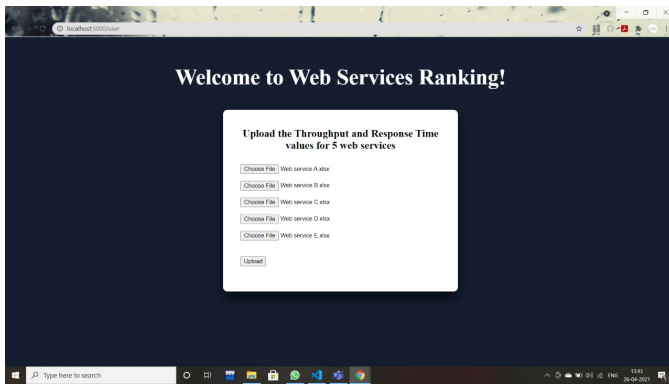

Fig. 7: Server Initialized

Fig. 8: User Input page

Fig. 11: Results page

Once the Submit button is clicked, the files are uploaded and files details are sent to the server side using the POST method. The file details are retrieved from the request url and then from the upload folder, each file is sent to the 3 classifier models and the trust score percentage is returned for each classifier model whose average is computed.

Terminal Output of predicted array from the 3 classifiers and the Ts percent of the website is shown below:
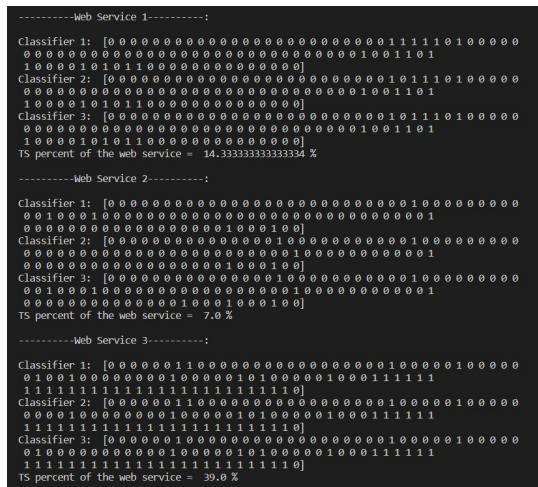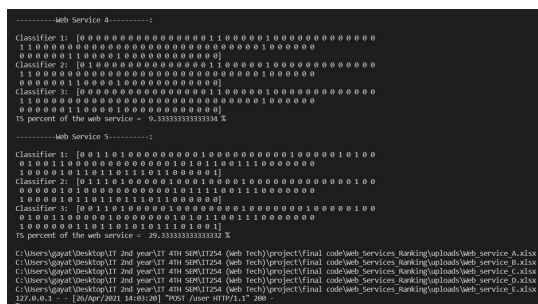

Fig. 9: Terminal Output1

Fig. 10: Terminal Output2

The final trust score percentages are sent to the "results.html" which is rendered by the server.
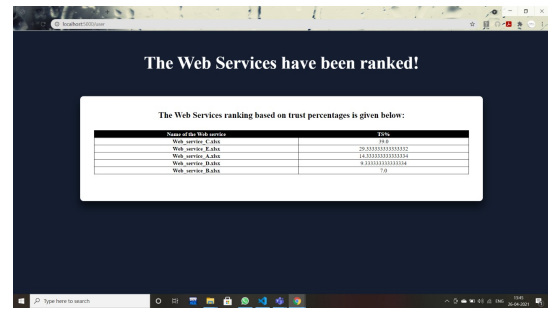
VIII. RESULTS AND ANALYSIS

All of the computation and model building was done on an i5 processor and the result obtained are as follows. The different metrics that we have used to evaluate the model are:

$$Precision = TP/(TP + FP)$$
$$Recall = TP/(TP + FN)$$
$$F1-Score = 2*(Precision*Recall)/(Precision+Recall)$$

(where TP-True Positives, FP-False Positives,FN-False Negatives)

The figure shows that the 0 being non-trusted has a precision of 0.91 , recall of 0.80 and 0.85 f1-score of 0.85. The 1 being trusted has a precision of 0.96, recall 0f 0.98 and f1-score of 0.97. It is clearly seen that the non-trusted part has a less f1-score and increasing it is definitely a potential improvement area of the model.

Fig. 12: Classification report

We also have the confusion matrix analysis of the model. The model is tested using three k-fold cross validation 5-fold,10-fold and 15-fold respectively. The matrix below show the results of the testing.

TABLE I: Matrix terminology

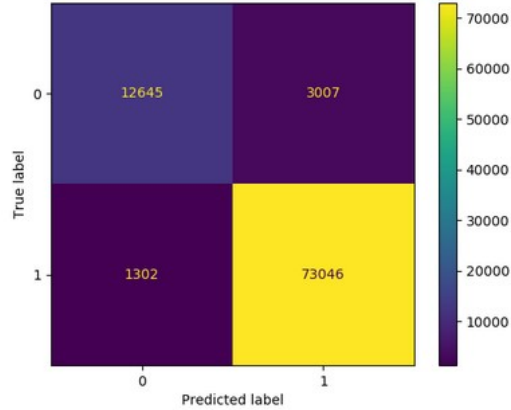| (0,0)	True Negative
(0,1)	False Positive
(1,0)	False Negative
(1,1)	True Positive

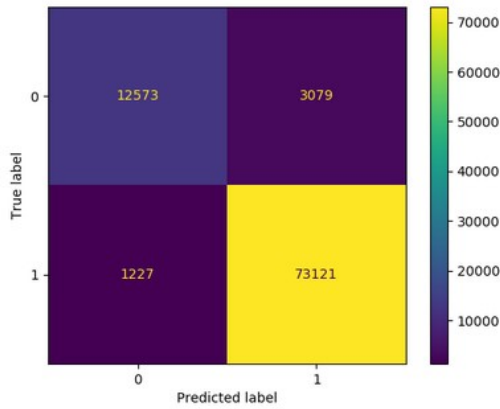Fig. 13: Confusion Matrix for 5-Fold cross validation

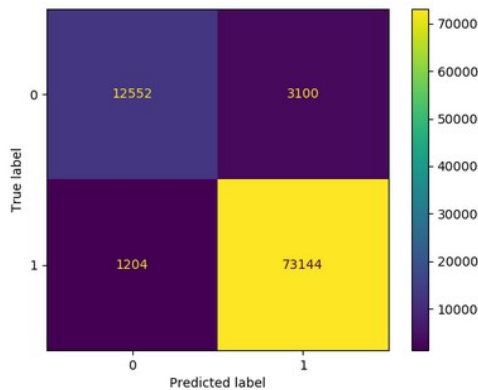Fig. 14: Confusion Matrix for 10-Fold cross validation

Fig. 15: Confusion Matrix for 15-Fold cross validation

IX. CONCLUSION

We have developed a web ranking system based on the user feedback in terms of response time and throughput. We have proposed fuzzy rules for determining if the following feedback indicates if the web service is trusted or not. Next we have established a trust prediction formula based on the prediction array we are using. We have used AdaBoostM1 to predict the trusted and untrusted values in the given responses from the user. We have trained these models using three different k-fold cross validation. This paper shows that the software architects can build better web services based on the trust features of customers. The web service architects can use the web service ranking to build better services by improving the quality of web service based on the user's trust.

X. TIMELINE
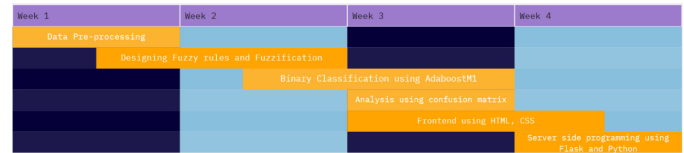
XI. INDIVIDUAL CONTRIBUTION

1) Sohan - 191IT149:
- Analysed the dataset and performed preprocessing on it through steps like normalization.(normalise.py)
- Trained the AdaboostM1 model on the web services dataset. Used the cross k-validation technique to evaluate the proposed model. Three k-folds 5,10 and 15 were used to train the models.(Adaboost.py)

2) Harini - 191IT221:
- Worked on normalizing the dataset as part of the preprocessing step.This was done to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. (normalise.py)

- Used the python framework flask to take the .xlsx files input from the users as part of the graphical user interface that is created.(WS_Ranking.py in Web_Services_Ranking.zip)

3) Gayathri - 191IT116:
- Decided on and implemented the fuzzy rules. The rules are used to classify the web services as either trusted(1) or not trusted(0) based on the throughput and response time values by different users.(fuzzy_rules.py)
- Created a graphical user interface using html and css in the frontend and flask in the backend.User can input five .xlsx files containing the throughput and response time values of 100 users for 5 web services. Once the upload is complete, the ranking of the web services are predicted using the adaboost classifier and the results are displayed to the user.(Web_Services_Ranking.zip which includes WS_Ranking.py,html and css files)

REFERENCES

[1] The github repository link which consists of the code: https://github.com/harini0-0/Web_Services_Ranking

[2] Dataset from WS-Dream. Available at: https://github.com/wsdream

[3] Base paper - M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani and R. Budiarto, "Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking," in IEEE Access, vol. 8, pp. 90847-90861, 2020, doi: 10.1109/ACCESS.2020.2994222.

[4] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for Web service recommendation," Knowl.-Based Syst., vol. 115, pp. 5565, Jan. 2017.

[5] Y. Ma, S.Wang, P. C. K. Hung, C.-H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown Web service QoS values," IEEE Trans. Services Comput., vol. 9, no. 4, pp. 511523, Jul. 2016.
. Almulla, H. Yahyaoui, and K. Al-Matori, "A new fuzzy hybrid technique for ranking real world Web services," Knowl.-Based Syst., vol. 77, pp. 115, Mar. 2015.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world Web services," IEEE Trans. Services Comput., vol. 7, no. 1, pp. 3239, Jan./Mar. 2014.

[7] A. Bawazir, W. Alhalabi, M. Mohamed, A. Sarirete, and A. Alsaig, "A formal approach for matching and ranking trustworthy context-dependent services," Appl. Soft Comput., vol. 73, pp. 306315, Dec. 2018.

[8] A. Ben-David and E. Frank, "Accuracy of machine learning models versus 'hand crafted' expert systemsA credit scoring case study," Expert Syst. Appl., vol. 36, no. 3, pp. 52645271, 2009.