ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 12 (2022) No. 6, pp. 1-16
DOI:10.33168/JSMS.2022.0601

Improving the Prediction Resolution Time for Customer Support Ticket System

Su-Cheng Haw [1], Kyle Ong [1], Lit-Jie Chew [1], Kok-Why Ng [1], Palanichamy Naveen [1] and Elham Abdulwahab Anaam [2]

[1] Faculty of Computing and Informatics, Multimedia University,63100 Cyberjaya, Malaysia

[2] Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Malaysia

sucheng@mmu.edu.my

Abstract. Processing customer queries on time will able to engage customer satisfaction, and thus improve the customer retention of a company. Increasing the labour to process these queries is certainly not an ideal solution. Advancing technology such as artificial intelligence and machine learning has led to the goal of automating this process, by predicting the time needed to resolve certain issues based on past similar cases. In this paper, we present the architecture for the Customer Support Ticket System to improve the accuracy of the predicted resolution time. In this research, we first perform the one hot encoding on the categorical variables, followed by feature selection. Next, a combination of classification and regression models is being utilised in our prediction pipeline. Experimental evaluations demonstrated that the Random Forest (RF) regression model has the best performance as compared to Neural Network and ADA boost. In addition, by adding the extremity feature as the attention, a significant performance boost for RF is observed.

Keywords: prediction, predictive analytics, resolution time, customer support, ticket system

1. Introduction

Predictive analytics refers to the use of historical data and statistical models to predict unknown future events. Predictive analytics has become a very popular concept, with its interest steadily rising over the past several years, thanks to the growth of data and technology such as artificial intelligence, machine learning and business intelligence. With predictive analytics, many organizations are now able to utilize past and current data to reliably forecast trends and behaviour, seconds, days or even years into the future (Roy et al., 2022; Zhang et al., 2021). In a business environment, organizations can employ predictive analytics to discover and exploit interesting patterns within data to detect potential market risks and opportunities. Predictive analytics is based on the algorithm, pattern discovery, trend analysis and artificial intelligence to enhance future predictions. Typically, predictive analytics combines powerful analysis technologies with automated discovery algorithms to forecast future events based on the analysis of historical data.

This paper aims to propose predictive analytics to estimate the resolution time needed to solve a particular issue to enable the customer to know tentatively how long duration is needed for their problem to be resolved.

2. Literature Review

The volume of support tickets has grown significantly due to the digitalization efforts across all industries (Amin et al., 2020). Many companies face increasing pressure in automating their STSs to increase customer satisfaction (Stein et al., 2018' Gupta et al., 2018) and to reduce costs (Al-Hawari et al., 2019).

With the emergence of Machine Learning, opens the possibility for automated ticket classification and thus, enables the prediction of the resolution time needed to solve the cases (Chagnon et al., 2017; Han and Sun, 2020). Many ML algorithms existed in the STS system. Among them were Support Vector Machine (SVM), Random Forrest (RF), Decision Tree (DT), K-Nearest Neighbor (KNN), Neural Network (NN), Rule-based, Deep Learning (DL) and so on.

From the review, SVM and its variations yield the best accuracy, which is between 63% (Revina et al., 2020) to 98% (Parmar et al., 2018). It is observed that the accuracy depends heavily on the dataset employed (Parmar et al., 2018; Iscen and Gurbuz, 2019; Yang, 2021).On the other hand, RF outperforms next with its accuracy ranging from 78% to 92% as surveyed by Fuchs et al. (2021). They have summarized the review based on 41 papers from well-established databases such as IEEE, Scopus, Ebsco and Web of Science. Towards recently, DL shows promising results especially to support of large training data (Bejarano et al., 2020).

Several other models existed. Among them are Multiple Linear Regression, Adaboost Regression, Elastic Net Regression and Gradient Boosting Regression (Bajariya et al., 2022; Malikireddy et al., 2021). By summarizing the related literature, it seems that most of the boosting techniques outperformed the others.

Bayesian Network and Logistic Regression had good prediction effects while SVM and LLM were relatively worse to handle large datasets (Jain et al., 2020). For this research, we have chosen the evaluate the non-linear regression models, which are NN, Adaboost and RF.

3. Methodology

Predictive analytics is used to make predictions about unknown future events. One of the most common predictive analytics models is the regression model. For this product, a regression model is used for resolution time prediction (RTP).

Resolution time is the time taken for an issue to be logged into until it is fully resolved. Many problems of current Customer Relationship Management (CRM) systems can be solved with RTP. First and foremost, RTP can improve the support ticketing module, which allows companies to prioritize tickets — customers' support interactions, assign them to reps and track their progress. Next, customers are informed of the predicted resolution time, which allows them to wait for a certain period of time until further updates. Finally, agents are able to utilize RTP to effectively resolve a case, which improves agent performance and time management.

In this research, users can log or register service request cases into the system, and the system will predict its resolution time. Besides, users will be also presented with other information as well include estimated severity level, estimated time of resolution and top 10 similar cases. Furthermore, users are also able to resolve the case and evaluate whether the case has been resolved on time (estimated time of resolution is ahead of the actual time of resolution) or overdue (actual time of resolution is ahead of the estimated time of resolution). The architecture of the product is shown in Fig. 1.

3.1. Offline Modeling

The predictive analytics' offline phase also known as the offline modelling process, consists of 5 different parts — data acquisition, data preprocessing, feature selection, model training and model deployment. Python is the chosen programming language for modelling and Jupyter notebook as its analytics environment. A variety of libraries have been used, mainly pandas and scikit-learn (sklearn). Pandas is used for data analysis and manipulation, while sklearn is used for predictive data analysis. The architecture of the modelling process is shown in Fig. 2 below. The modelling process will be explained along with the code snippets obtained from Jupyter Notebook.

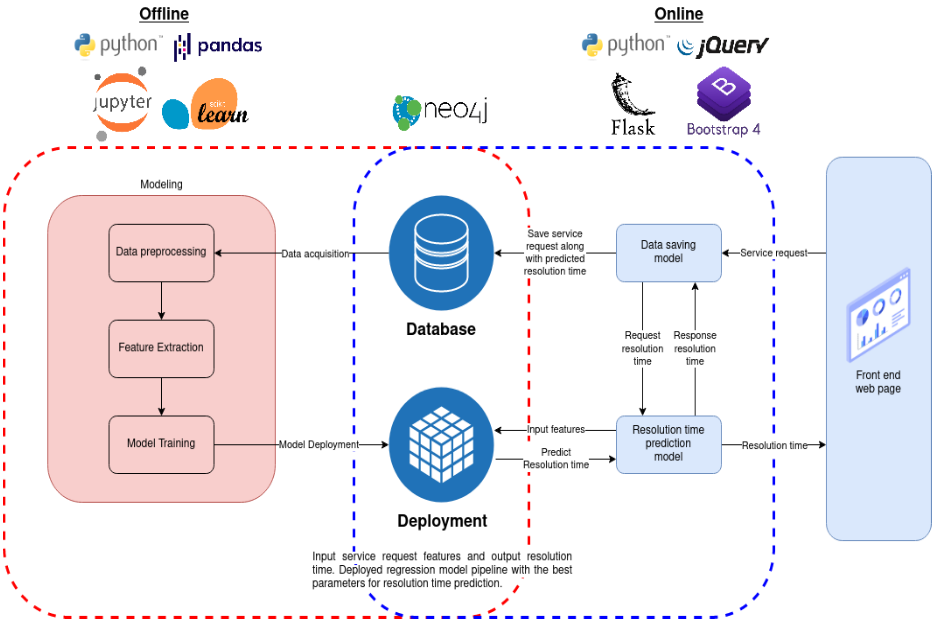


Fig. 1: Predictive analytics architecture

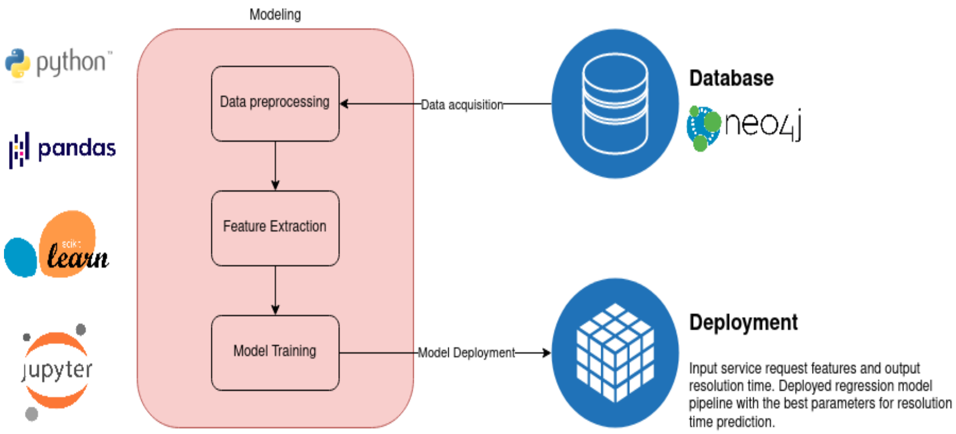


Fig. 2: Offline Predictive analytics architecture

3.1.1. Data Acquisition and Preprocessing

The real dataset is a CRM dataset obtained from a proprietary telecommunication company in Malaysia. The dataset is in Comma-Separated-Value (CSV) format. The dataset consists of 47 attributes and approximately 80,000 entries collected from January 2018 to May 2019 (18 months). The data undergo a series of preprocessing such as raw data transformation, data stripping, spelling correction, upper-lowercase conversion, and data derivation.

3.1.2. Feature Selection

Feature selection or feature extraction is the process of selecting a subset of features manually or automatically in modelling. The goals of feature selection are usually to improve the model's ability to generalize by removing irrelevant features and reducing training time. Here, we performed feature selection automatically through several statistical methods which include entropy, point biserial correlation coefficient and chi square correlation coefficient.

Entropy is a measure of uncertainty or randomness of data. Entropy can be calculated by using the formula below, where k is the number of categories. and $0 \leq H(X) \leq log_2 k$.

$$H(X) = -\sum_{i=1}^{n} p_i \, log_2 \, p_i$$

The closer the value to $log_2 k$, the higher the complexity of the data, and the harder it is to draw any conclusion. Hence, we can utilize the idea of entropy to determine the "diversity" of categorical data. By using the code snippets in Fig. 3, we can identify the entropy for each categorical variable and eliminate variables that have less than a normalized entropy threshold (0.3). Normalized entropy can be calculated by dividing the entropy $H(X)$ from the maximum entropy $log_2 k$.

Besides, point biserial and chi squared correlation coefficient are another two statistical methods in the feature selection process. Both methods result in a coefficient value which varies between -1 (strong negative relationship) and +1 (strong positive relationship), with 0 implying no correlation.

```
describe = df.select_dtypes(["object", "category"]).describe().T
describe["entropy"] =  describe.index.to_series().apply(
        lambda x:stats.entropy(df[x].value_counts(), base = 2)
)
describe["max_entropy"] =  describe.index.to_series().apply(
        lambda x:math.log(df[x].nunique(),2)
)
describe["normalize_entropy"] =  describe.index.to_series().apply(
        lambda x:stats.entropy(df[x].value_counts(), base = df[x].nunique())
)

display(describe.sort_values("normalize_entropy", ascending = False))
```

Fig. 3: Code snippets for entropy

Both point biserial and chi squared are univariate feature selection methods. Point biserial is a correlation coefficient used when one variable is dichotomous (0 or 1) and another variable is continuous, while chi squared is a correlation coefficient used when both variables are dichotomous. Since most of our categorical values are not dichotomous or 2-dimensional, we first encoded all categorical features as a one-hot numeric array. Also, it is encouraged to transform categorical columns into numeric representation, as most machine learning algorithms are not able to work with categorical data in string format directly. One hot encoding separates categorical columns into many columns depending on the number of categories present in that column. For example, if column A has 3 unique categories while column B has 4 unique categories, then one-hot-encoding separated column A into 3 unique columns and column B into 4 unique columns, where each separated column only contains dichotomous values - No (0) or Yes (1).

After performing one hot encoding on the categorical variables, feature selection is performed. We utilized one of the univariate feature selection transformers provided by sklearn — SelectFdr() where features with low false discovery rates are kept. Point biserial and chi squared will also perform null hypothesis tests on each feature to the label (target variable) and return a coefficient value and a p-value. SelectFdr() will select features based on criteria — p-value larger than the alpha-value (0.05) or coefficient-value smaller than + or - 0.1 will not be selected.

3.1.3. Model Training

A combination of classification and regression models have been used in our prediction pipeline. Classification model is used to predict (classify) extremity, which will be later concatenated as an additional feature for resolution time regression. The classified extremity is based on the table guideline shown in Table 1.

Table 1: Calculated entropy for each categorical variable

Attributes	entropy	max_entropy	normalize_entropy	less than 0.3 and eliminated?
insert_by_name	4.436938	5.087463	0.872132	False
group	4.757756	6.108524	0.778871	False
sub_group	6.133096	7.965784	0.769930	False
category	1.827737	2.584963	0.707065	False
problem_location	5.298039	7.658211	0.691812	False
sub_category_3	4.975126	7.475733	0.665503	False
sub_category_2	3.971061	6.539159	0.607274	False
division	2.520255	4.459432	0.565152	False
sub_category_1	3.323913	6.044394	0.549917	False
source	1.168561	2.321928	0.503272	False
status	1.958831	4.321928	0.453231	False
type_call	0.873446	2.000000	0.436723	False
root_cause	2.582387	7.417853	0.348131	False
store_type	1.122447	3.906891	0.287299	True
priority	0.001453	2.000000	0.000727	True
product	0.000305	1.000000	0.000305	True

The experiment was performed with 70 percent training set and 30 percent testing set. At the same time, we utilized sklearn's GridSearchCV for hyperparameter tuning. GridSearchCV performs an exhaustive search over specified parameter values for an estimator.

3.1.3.1. Classification

We first performed classification modelling. The chosen classification model is a decision tree classifier with one-vs-rest multiclass classification strategy. One-vs-rest multiclass classification strategy consists in fitting one classifier per class, where the class is fitted against all the classes. For example, by applying one-vs-rest multiclass strategy on 3 colour classes — red, green and blue, will generate 3 binary classifiers — red or non-red classifier, green or non-green classifier and blue or non-blue classifier. In addition to its computational efficiency, one advantage of this approach is interpretability, since each class is represented by one and only one classifier. Fig. 4 shows one of the classifiers in the one-vs-rest multiclass strategy, while Fig. 5 shows the classification confusion matrix for both the training and testing set.

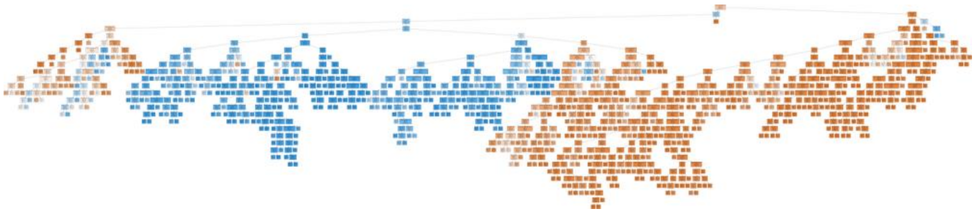
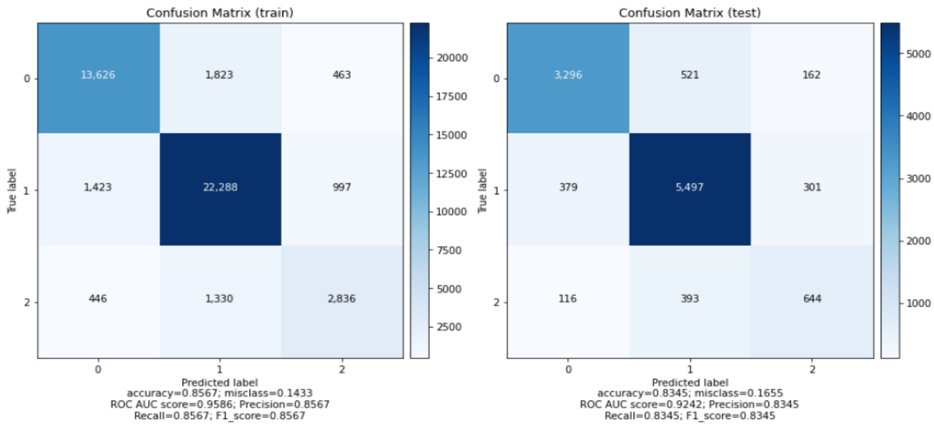


Fig. 4: One of the decision tree in one-vs-rest multiclass strategy



class	precision	recall	fscore	support
0	0.8794	0.8563	0.8677	15912.0
1	0.8761	0.9021	0.8889	24708.0
2	0.6601	0.6149	0.6367	4612.0

class	precision	recall	fscore	support
0	0.8694	0.8283	0.8484	3979.0
1	0.8574	0.8899	0.8734	6177.0
2	0.5818	0.5585	0.5699	1153.0

Fig. 5: Classification confusion matrix

3.1.3.2. Regression

After completing the classification modelling, we proceeded to regression modelling. The classified extremity will be concatenated as an additional feature for resolution time regression. The classified extremity also acts as an "attention" for the regression model. The "attention" effect allows the regression model to learn easier, as it enhances the important parts of the input data — extremity, and fades out the rest. The predicted extremity also has a strong positive relationship with the resolution time. In addition, we also removed any incorrect classified extremity data.

The chosen regression model is a RF regressor. RF regressor has been shown to outperform any other experimented regressor due to its structures and nonlinear nature. RFs are able to aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield better results. Fig. 5 shows the regression experimental plot which includes the true versus prediction plot and residual plot for both the training and testing set.

3.1.4. Model Development

After completing the model training, the trained model pipeline is then deployed into the product (system), which is ready to be used in the online phase. The online phase utilizes the model pipeline trained in the offline phase for resolution time prediction (RTP) (see Figure 7). The system is mainly for admin-use. The backend of the system is coded using Flask, a light-weight microframework built on top of Python programming language. The front-end website is built with bootstrap 4. At the same time, we utilized jquery for API access. The webpage consists of 2 main parts; (1) prediction and (2) evaluation.

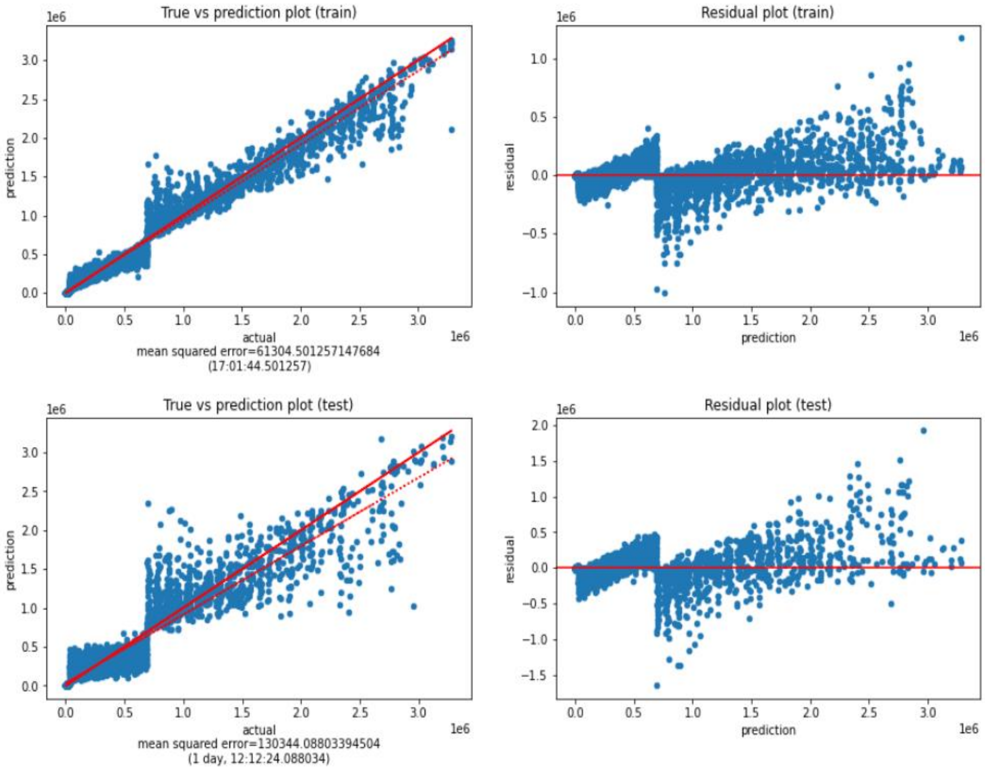


Fig. 6: Regression experimental plots

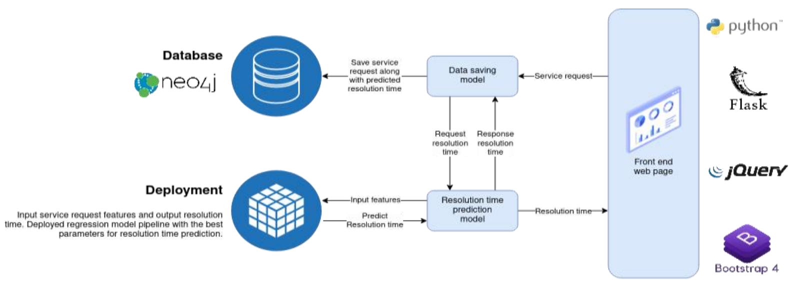


Fig. 7: Offline predictive analytics architecture

In (1), users can register a new service request into the system via a service request form as shown in Fig. 8. Through service request registration, the system can obtain the predicted resolution time from the trained model pipeline. The service request will be saved into the database along with the predicted resolution time. Then, the newly registered service request will be presented along with other unresolved cases in table format as shown in Fig. 9. By selecting one of the service request IDs, users will be presented with service request features as well as other information including estimated severity level, estimated time of resolution and top 10 similar cases shown in Fig. 10. Both predicted resolution time and top 10 similar cases can be obtained through API access.

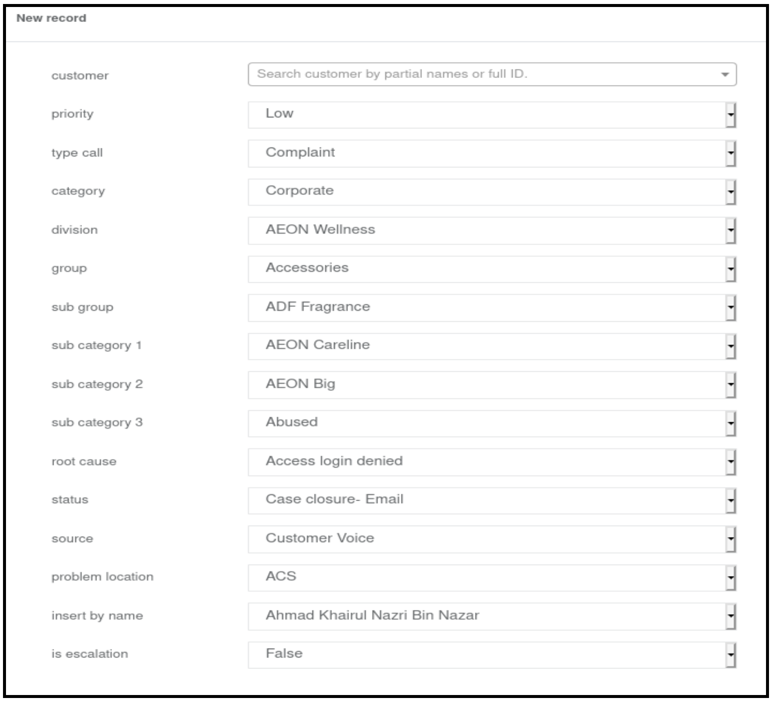


Fig. 8: Service request form

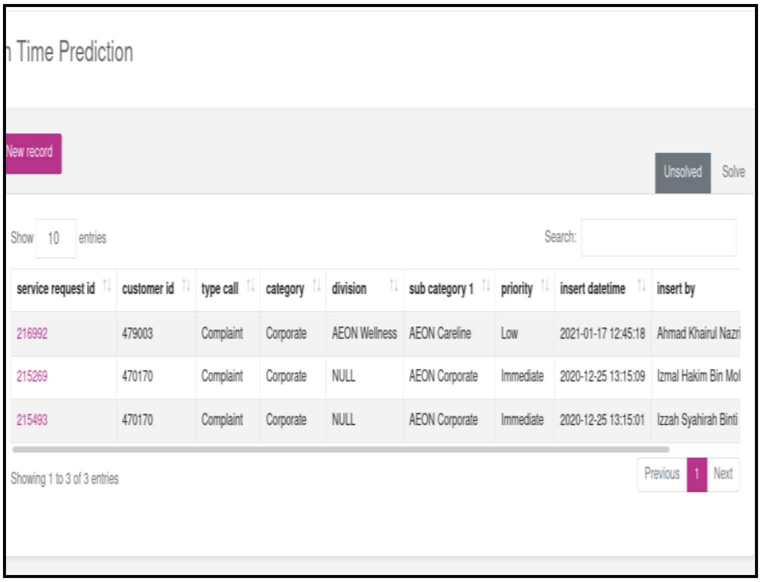


Fig. 9: Unresolved cases

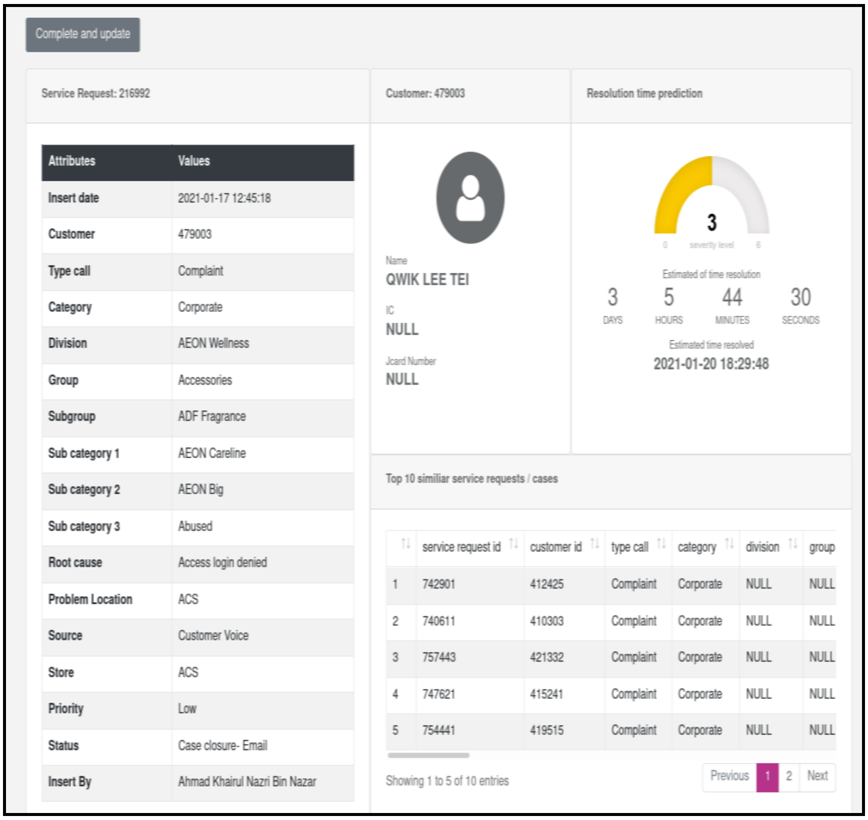


Fig. 10: Service request features, predicted resolution time and top 10 similar cases for unresolve case

In Fig. 10, a "Complete and update" button is shown at the top left corner of the interface. Users are able to resolve the case by clicking on the button. After a case has been resolved, the case will not exist in the unresolved table, whereas it will appear in the resolved table. The resolve table consists of a list of resolved cases as shown in Fig. 11. By selecting one of the service request IDs, users will also be presented with a similar interface as unresolved cases — estimated severity level, estimated time of resolution and top 10 similar cases. However, an additional text has been added in the prediction area where users are able to evaluate whether the case has been resolved on-time (estimated time of resolution is ahead of the actual time of resolution) or overdue (actual time of resolution is ahead of the estimated time of resolution) as shown in Fig. 12.

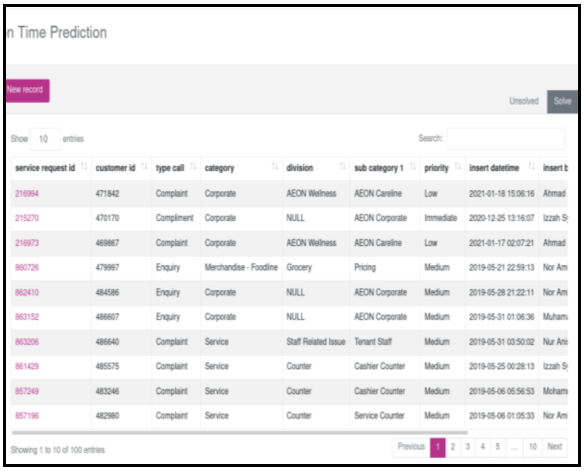


Fig. 11: Service request features

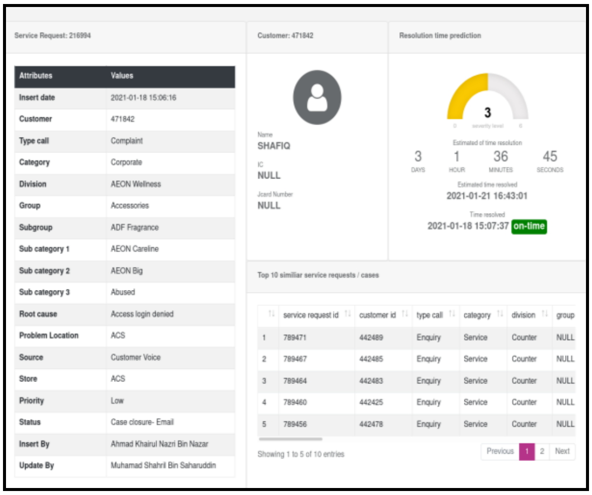


Fig. 12: Service request features, predicted resolution time and top 10 similar cases for resolve case

4. Evaluation and Discussion

We have implemented the three non-linear regression models and evaluated the performance in terms of Root Mean Square Error (RMSE). The train-test split ratio is 80:20 as this is the standard split ratio and through experimental evaluation, we observed that this splitting yields the optimised result.

Fig. 13 shows the actual versus the predicted plot for the train (left) and test (right) dataset across three non-linear regression algorithms: NN, ADA boost (middle) and RF (bottom). The closer the scatter plot touches the red line, the better the result is.

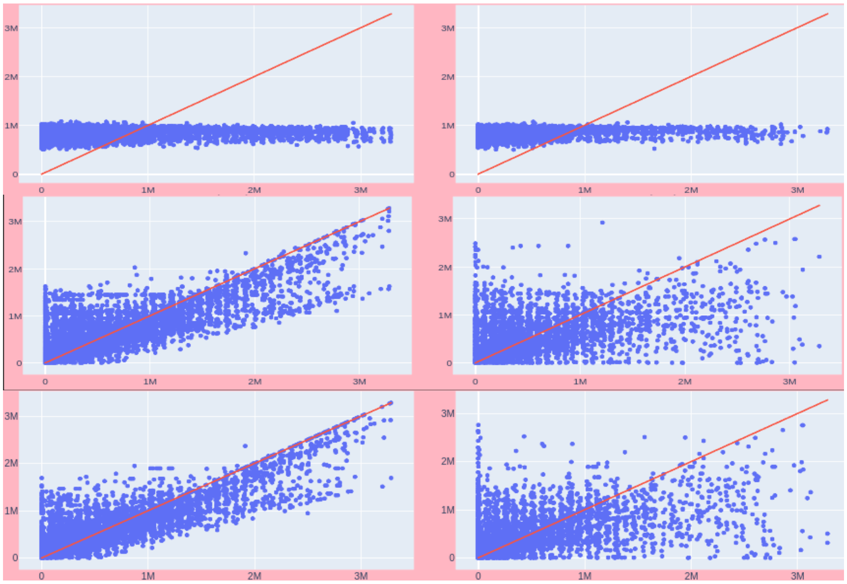


Fig. 13: Actual versus the predicted plot of NN, Ada Boost and RF

Fig. 14 depicts the residual plot for the train (left) and test (right) datasets across NN, ADA boost (middle) and RF (bottom). Similarly, the closer the red line touches the middle horizontal line (y=0), the better the result is.

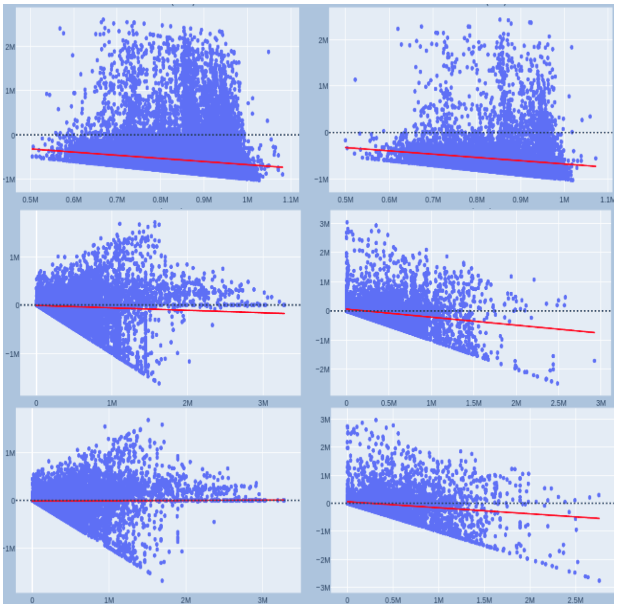


Fig. 14**:** Residual plot for train (left) and test (right) dataset across NN, Ada Boost and RF

Table 2 shows the RMSE evaluation result. From the result, we observed that the NN has the worst performance with 715957.265856 train RMSE in seconds and 717630.768069 test RMSE in seconds. On the other hand, the RF has the best performance with 199414.986081 train RMSE in seconds and 341463.237641 test RMSE in seconds. One possible explanation is that the data is not large enough, due to the "diversity" of the data.

At the same time, we can also see a significant performance boost for RF by adding the extremity feature as the attention. RF regressor with attention has a train RMSE in seconds of 61304.501257 and test RMSE in second of 130344.08803, improving 69.3% in train RMSE and 61.8% in test RMSE over regular RF regressor without attention.

Table 2: Calculated entropy for each categorical variable

Regressor	Train RMSE in seconds (days:hours:minutes:seconds)	Test RMSE in seconds (days:hours:minutes:seconds)
Neural Network	715957.265856 (8:6:52:32)	717630.768069 (8:7:20:31)
ADA Booster	242322.791599 (2:19:18:42)	358863.141741 (4:3:41:3)
Random Forest	199414.986081 (2:7:23:34)	341463.237641 (3:22:51:3)
Random Forest+ attention	61304.501257 (0:17:1:44)	130344.08803 (1:12:12:24)

References

Al-Hawari, F. & Barham, H. (2019). A machine learning based help desk system for IT service management. *Journal of King Saud University - Computer and Information Sciences*

Amin, K., Kapetanakis, S., Althoff, K. D., Dengel, A., & Petridis, M. (2018). Dynamic process workflow routine using deep learning. *SGAI International Conference on Artificial Intelligence*. DOI:10.1007/978-3-030-04191-5

Bajariya, A. & Jaiminee, J. P. (2022). A Review on text supervised learning methods for classification of it ticket and bugs. *Second International Conference on Artificial Intelligence and Smart Energy*, 903-907. DOI:10.1109/ICAIS53314.2022.9742981

Bejarano, G., Kulkarni, A., Luo, X., Seetharam, A., & Ramesh, A. (2020). DeepER: A deep learning based emergency resolution time prediction system. *International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 490-497. DOI:10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00090

Chagnon, J. C., Trapp, A. C., Andrew, C., & Soussan, D. (2017). Creating a decision support system for service classification and assignment through optimization

Fuchs, S., Drieschner, C., & Wittges, H. (2022). improving support ticket systems using machine learning: A literature review. *Hawaii International Conference on System Sciences*, 1893-1902

Gupta, M., Asadullah, A., Padmanabhuni, S., & Serebrenik, A. (2018). Reducing user input requests to improve IT support ticket resolution process. *Empirical Software Engineering*, 23(3), 1664-1703

Han, J. & Sun, A. (2020). DeepRouting: A deep neural network approach for ticket routing in expert network. *IEEE International Conference on Services Computing*

Iscen, E. S. & Gurbuz, M. Z. (2019). A comparison of text classifiers on it incidents using WEKA. *4th International Conference on Computer Science and Engineering*

Jain, H., Khunteta, A., & Srivastava, S. (2020). Churn prediction in telecommunication using logistic regression and logit boost. *Procedia Computer Science*, 167, 101-112. DOI:10.1016/j.procs.2020.03.187

Malikireddy, V. P. & Kasa, M. (2021). Customer churns prediction model based on machine learning techniques: A systematic review, atlantis highlights in computer, 4, 167-174. DOI:10.2991/ahis.k.210913.021

Parmar, P. S., Biju, P. K., Shankar, M., & Kadiresan, N. (2018). Multiclass text classification and analytics for improving customer support response through different classifiers. *International Conference on Advances in Computing, Communications and Informatics*

Roy, D., Srivastava, R., Jat, M., & Karaca, M. S. (2022). A complete overview of analytics techniques: descriptive, predictive, and prescriptive. *EAI/Springer Innovations in Communication and Computing. Springer, Cham*. DOI:10.1007/978-3-030-82763-2_2

Revina, A., Buza, K., & Meister, V.G. (2020), IT ticket classification: The simpler, the better, IEEE Access, 8, 193380-193395

Stein, N., Flath, C., & Bohm, C. (2018). Predictive analytics for application management services. *European Conference on Information Systems*, https://aisel.aisnet.org/ecis2018_rp/186/

Yang, L. (2021). Fuzzy output support vector machine based incident ticket classification. *IEICE Transactions on Information and Systems*, E104.D(1), 146-151

Zhang, J. Z., Srivastava, P. R., Sharma, D., & Eachempati, P. (2021). Big data analytics and machine learning: A retrospective overview and bibliometric analysis. *Expert Systems with Applications*, 184. DOI:10.1016/j.eswa.2021.115561