Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 437–446
DOI: 10.15439/2022F297
ISSN 2300-5963 ACSIS, Vol. 30

Application of Diversified Ensemble Learning in Real-life Business Problems: The Case of Predicting Costs of Forwarding Contracts

Milena Trajanoska*, Pavel Gjorgovski*, Eftim Zdravevski
Faculty of Computer Science and Engineering
Ss.Cyril and Methodius University, Skopje, Macedonia
ORCIDs: {0000-0003-0105-7693, 0000-0002-6859-4402, 0000-0001-7664-0168}

***Abstract*—Finding an optimal machine learning model that can be applied to a business problem is a complex challenge that needs to provide a balance between multiple requirements, including a high predictive performance of the model, continuous learning and deployment, and explainability of the predictions. The topic of the FedCSIS 2022 Challenge: 'Predicting the Costs of Forwarding Contracts' is related to the challenges logistics and transportation companies are facing. To tackle these challenges, we established an entire Machine Learning framework which includes domain-specific feature engineering and enrichment, generic feature transformation and extraction, model hyper-parameter tuning, and creating ensembles of traditional and deep learning models. Our contributions additionally include an analysis of the types of models which are suitable for the case of predicting a multi-modal continuous target variable, as well as an explainable analysis of the features which have the largest impact on predicting the value of these costs. We further show that ensembles created by combining multiple different models trained with different algorithms can improve the performance on unseen data. In this particular dataset, the experiments showed that such a combination improves the score by 3% compared to the best performing individual model.**

***Index Terms*—Costs of Forwarding Contract, explainability, prediction ensembles, Diversified Ensemble Learning**

I. Introduction

TO BE competitive in the market, companies need to be able to utilize all available data and perform analytics to identify hidden patterns [1]. This can allow them to improve their processes, better understand their customers, and make predictions (e.g. churn prediction, service-outage prediction, fraud detection, etc.). To achieve such goals, companies are facing a variety of challenges, ranging from data integration from a variety of sources [1] and finding suitable machine learning models that are both performant but also practical and explainable [2], to maintaining the corresponding infrastructure. To perform such analytical data processing and machine learning on a large scale, companies require a complex computing infrastructure and methods that will minimize their total cost of ownership [3], and yet scale the computation to multiple nodes [4].

In this paper, we focus on the problem of finding an optimal machine learning algorithm that can be easily applied in a real-life business domain, meaning it should achieve high predictive performance, continuous learning and deployment, and explainability of the models and their predictions. The topic of the FedCSIS 2022 Challenge, hosted on the KnowledgePit portal is 'Predicting the Costs of Forwarding Contracts' [5]. The competition addresses the challenges of transportation, shipping, and logistics companies related to their digital transformation. Particularly, the benefits of the research boosted by this competition for such companies can be multi-fold:

- Identify reasons and circumstances that lead to increased transportation costs.
- Improve companies' planning to lower the costs, and generally, improve their investment strategy.
- Help companies in selecting contracts that maximize their profits by predicting the forwarding (i.e., delivery) contract cost.

Similar real-world challenges were addressed at previous competitions on the KnowledgePit platform, such as predicting escalations in customer support [6], network device workload prediction [7], suspicious network event recognition [8], and predicting victories in video games [9], to name a few. These papers also demonstrate how predictions from individual solutions could be integrated into diversified ensembles to create more powerful and more robust models.

For the case study on which we focus in this paper, we choose to use the XGBoost model with a grid search with 5-fold cross-validation [10], due to its extensive use in retail sale predictions [11]. We also use Random Forest models with grid search, as well as deep learning models that are commonly used in demand forecasting in multi-channel retail [12]. Finally, the Linear Regression models are one of the most commonly used simple models for price prediction in industry

*These authors contributed equally to this work.

settings [13]. All hyper-parameter tuning is done via exhaustively searching a specified subset of the hyper-parameter space of the given model. The validation is performed using a 5-fold cross-validation, which balances validation speed and metric accuracy of the test data.

The rest of the paper is structured as follows. Section II reviews the most important related works. Section III describes the experimental setup and multiple validation procedures we used to evaluate each model we have developed in this study. Section IV describes the preprocessing of the data, including data transformations and aggregations. Subsection IV-A contains information on the preprocessing implemented over the main table training and testing data, and subsection IV-B includes the preprocessing information for the routes table training and testing data. The section V describes the implemented feature selection methods. The experiments, including model hyper-parameter tuning, training, and evaluation techniques are described in section VI, along with an overview of the final scores of the implemented models. The paper concludes with section VIII where we give a brief overview of the entire Machine Learning workflow, limitations to the study, and opportunities for further work and improvement of the methods.

II. Literature Review

Even though similar challenges have been extensively studied in other industries, this problem is fairly new in the logistics sector. Authors of [14] analyze the shipping cost differences between various carriers, and attempt to identify opportunities for reducing transportation costs.

Similarly, in [15], authors utilize neural networks to forecast shipping freight rates and compare them with traditional time series analysis models. The key objective of their work is to improve the forecasting accuracy of traditional time series analysis. In relation to the competition task, this article also highlights the importance of the information contained in forwarding freight agreements in relation to predictive accuracy.

Another interesting approach is presented in [16], where the impact of the demand and cargo capacity on the shipping price is identified. Forecasting of the long-term cost of logistics contracts is particularly important in long-term agreements with upfront-defined prices, such as in various types of tenders and auctions. On one hand, the bids should be attractive so that the contract can be won, while still being profitable for the logistic company. This challenge was researched in [17], which utilized historic data to train the models.

On a related topic, Men et al. [18] use an ensemble of mixture density neural networks for the purpose of short-term wind speed and power forecasting. They show that this methodology works well for multi-step ahead prediction. Additionally, [19] illustrates the use-case of multi-observation and multi-dimensional data cleaning methods for applying machine learning algorithms. In this study [19], the authors use transactions from the Lending club data set for training tree-based models to predict peer-to-peer (P2P) loan default and observe that the LightGBM algorithm, using multiple observational data, has the best performance. In many cases, it has been shown that decision tree-based methods significantly outperform linear models for predicting complex response variables, such as the example of predicting accrual expenses in a balance sheet by utilizing the unused vacation time of employees [20].

The scientific community has placed a massive effort into studying individual algorithms (e.g. ensemble algorithms, various deep learning architectures, etc.). Additionally, some studies also focus on finding ways to utilize the diverse algorithms and integrate their predictions. This process is often referred to as diversified ensemble learning and aims to find the best classification algorithms (out of many heterogeneous classification algorithms) and an optimal method to combine them [2]. Note that the individual algorithms used in a diversified ensemble could be ensembles on their own (e.g., XGBoost [21] or Random Forest [22]), so the term diversified ensemble learning refers to another layer of integration. Some methods train another classifier whose inputs are the predictions of the individual classifiers [23] or use other ways of voting. In this paper, algorithms perform weighted voting based on empirically identified weights.

III. Validation procedure

As in all practical machine learning problems, the experimental setup concerning the training/validation/test split should resemble the natural chronological and logical process as closely as possible, so that the models built are valid and robust over time. In that regard, we attempted to split the training dataset into two subsets, one for training and one for validation, in a way that we thought would most resemble the natural setting in which the data was collected. Considering that this is a very practical problem coming from the industry, any results of the transformation and validation methods should be applicable in a production setting.

That being said, we considered the id_payer column, the client identifier, as special because it gave us the ability to use it primarily for splitting the original training set into our training and validation subsets. For this purpose, we first analyzed the frequency of rows in the main table per id_payer, dubbing it $number\_of\_contracts$. We noticed the huge discrepancy in the frequency of contracts, ranging from just a few to upwards of thousands. Therefore, we tried several approaches in how we considered this fact:

- **Split by alternating frequency of records per** id_payer. In this approach, we ordered the id_payer records by the $number\_of\_contracts$, and we assigned them to our training or validation split in alternating order. The idea was that roughly 50% of the records will be our training set, and the other 50% will be the validation set. One additional benefit of this approach was that it made sure that the id_payer column would not have an effect on the prediction. With this approach we are very conservative to overfitting, trying to train the models on one subset of the data, and applying the models to a completely new set of data. Indeed, our

first submissions showed that our own validation results were considerably worse than the leaderboard results, but were still consistent when comparing different algorithms or feature subsets (the better models per our internal evaluation were also better on the leaderboard).

- **Time-sensitive split**. We also tested splitting the data in such a way that the older contracts (records with an earlier start date) were in the training set, while newer records were in the validation set. This approach mitigates the previous conservativeness, by allowing the same clients to be in the training and validation set, while also allowing some new clients to appear in the validation set.

After the initial testing of the previous approaches, we noticed that the hyper-parameter tuning procedures performed on our hold-out training set (a subset of the competition training set) were not fully applicable when we used the whole training dataset provided in the competition. Namely we used our hold-out training set and the remaining of the training set to learn the hyper-parameters. Then, we compared two models trained with the same hyper-parameters – one using the hold-out training set, and another trained on the full training dataset. The former performed significantly better on the leaderboard result, even though it was trained on smaller data set. With this counter-intuitive finding that contradicts the common principle that more training data is better, and having a very limited time for this competition, we decided to use 5-fold cross-validation in the remaining experiments so that we can use the full training dataset for making the final test predictions. Despite that, we strongly believe that further experiments in the validation procedure are needed to properly tackle the problem.

IV. Data Preprocessing

After the initial data exploration phase, we decided to primarily focus on the main table and extract whichever knowledge we can from it, before proceeding with utilizing the detailed table of expected routes.

A. Main Table

Firstly, the $prim\_train\_line$ and $prim\_ferry\_line$ features were not used, due to the high missing data ratio (between 80% and 90% missing from the total number of observations). Additionally, these columns had unstandardized data (e.g., temperature ranges or temperature and unit combined strings in the same column as a descriptive field, etc.) For the remaining columns which had missing data, we applied mean (for continuous columns) or median filling (for nominal data).

The transformations done on the Main Table were split into two major types:

- One-hot encoding of categorical (nominal) data. We considered utilizing the Weight of Evidence [24] approach, but considering that the categorical features had a relatively small number of different values, the one-hot encoding technique was considered sufficient.
- Combinations of two or more features to create a new meaningful feature. Such features were a result of calculations based on the columns that contain date or timestamp information.

1) Nominal to numeric features with one-hot encoding: The one-hot encoding was done to maximize data balance while minimizing the loss of information. Binary features or features with a few different values were transformed with classic one-hot encoding. The features with over 15 values were split into 3 major categories: low-frequency categories (those that had appeared under 1000 times in the data set), high-frequency categories (those that had appeared over 1000 times in the data set), and the highest frequency category of the feature was separated as an individual category.

2) New domain-specific features based on other features:

a) Date-time related features: A combining of two or more features was done for the $route\_start\_numeric$ and $time\_taken\_minutes$ features. The $route\_start\_numeric$ is the difference in days between the minimum date found in the dataset (i.e., $1/1/2016$), and the start date of the specific route. This was done by using the $route\_start\_datetime$ feature, and finding the number of days between it and $1/1/2016$. Similarly, the $time\_taken\_minutes$ is the time the complete route is estimated to take in minutes. This is calculated by finding the difference in minutes between the $route\_start\_datetime$ and $route\_end\_datetime$ features.

b) Geo-spatial features: To enhance the geo-spatial information about the routes, we created a new feature, using the Euclidean distance [25] between the $route\_starting\_point$ and $route\_ending\_point$. This calculation uses the latitude and longitude values of the original points. Additionally, we used the geo-spatial (Haversine) distance [26], given in the competition dataset.

B. Routes table

The initial experiments were conducted using only data from the Main Table. To further improve model performance in later experiments, we enriched the dataset with aggregate features extracted from the Routes Table.

The columns which had missing data were very sparse in the general case. Moreover, the lack of entries seemed correlated in most cases. For this reason, we decided to ignore such columns and do not create features based on them, especially considering the limited time we had for experiments. Still, we believe that more sophisticated data imputation methods could be explored in the future, or at least to prepare some bins of values in cases when such data is available. The ignored columns for this reason were: $ferry\_line$, $train\_line$, and another 17 columns whose names started with $vehicle\_$ or $id\_vehicle\_$.

We have extracted the aggregate features by grouping the dataset based on the column $id\_contract$. Before the aggregation, one-hot encoding was performed on the $step\_type$ feature with the goal of extracting the number of steps of each type that were taken in one route. The following features were extracted for each route:

- $num\_steps\_with\_vehicle$ - the number of rows having $id\_vehicle$ equal to 1
- $num\_steps\_with\_trailer$ - the number of rows having $id\_trailer$ equal to 1
- num_steps - the maximum from the $step$ column values within one $id\_contract$ partition
- num_steps_A - the number of steps of type A
- num_steps_B - the number of steps of type B
- num_steps_D - the number of steps of type D
- num_steps_F - the number of steps of type F
- num_steps_K - the number of steps of type K
- num_steps_N - the number of steps of type N
- num_steps_O - the number of steps of type O
- num_steps_P - the number of steps of type P
- num_steps_R - the number of steps of type R
- num_steps_S - the number of steps of type S
- num_steps_W - the number of steps of type W
- num_steps_Z - the number of steps of type Z
- num_steps_A - the number of steps of type A
- num_steps_empty - number of the steps in which the if_empty flag was equal to 1
- $num\_external\_steps$ - the number of the steps in which the flag $external\_fleet$ was 1
- max_loaded_kg - the maximum kg_load_unload
- $max\_unloaded\_kg$ - the minimum of kg_load_unload
- $average\_load\_step$ - the mean of the feature $kg\_current$
- max_load_step - the maximum of the feature $kg\_current$
- min_load_step - the minimum of the feature $kg\_current$
- num_steps_ferry - the number of steps in which the flag $ferry$ was equal to 1
- num_steps_train - the number of steps in which the flag $train$ was equal to 1
- $total\_km\_train$ - the sum of $train\_km$ for each step
- max_time - the maximum of the feature $estimated\_time$ in each step
- min_time - the minimum of the feature $estimated\_time$ in each step
- km_per_step - the mean value of the feature km in each step
- $km\_nonempty\_max$ - the maximum value of the feature $km\_nonempty$
- $km\_nonempty\_total$ - the sum of $km\_nonempty$ in each step
- $average\_time\_per\_step\_minutes$ - the average of the time difference in minutes for each step

V. Feature Selection Methods

A. Manual Filtering of Correlated Features

This method was used for training the Linear Regression models. Since the Maximum likelihood (MLE) estimations [27] can be highly disturbed by correlated features, we decided to manually remove features with correlations greater than 0.6 in absolute value. For this purpose, we calculated the Pearson Correlation [28] between each pair of continuous variables in the main dataset. We only calculated the correlations between the continuous features from the main dataset in order to exclude the features that have a high correlation from the feature engineering step for training the linear regression models.

The linear regression models were only trained using the uncorrelated features from the main dataset, plus higher degrees of some of the most important features chosen by applying domain knowledge. These features include the total kilometers, the time taken and the maximum weight. This was done in order to use the results of the most basic linear regression models as an internal evaluation baseline for all the other trained models.

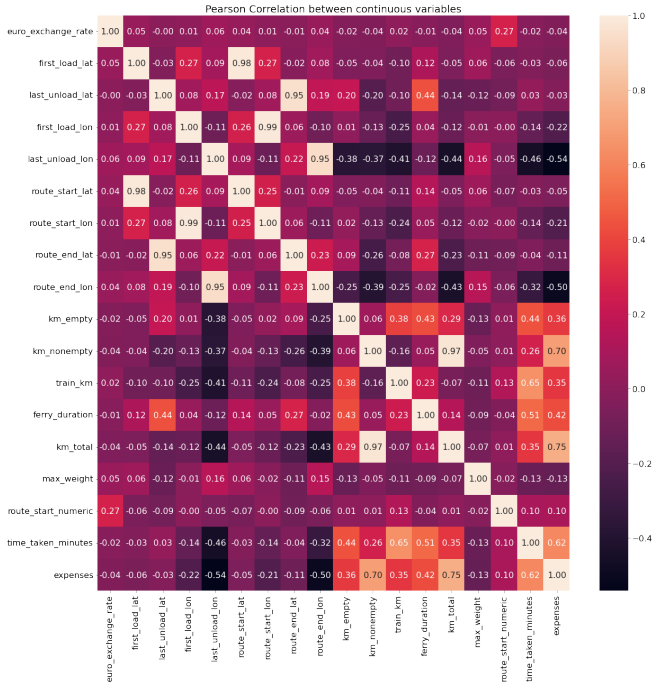

Fig. 1. **Main table extracted continuous features correlation.** This figure presents the Pearson Correlation coefficients calculated between each pair of continuous features from the main table.

The experiments for the Linear Regression models were conducted using only the Main Table data. The correlation map of the continuous features is displayed in Figure 1. As it is evident from the figure, a lot of features have high correlations (positive and negative), so removing these dependencies was one of the feature selection methods we implemented.

B. XGBoost Feature Importance

Another method for feature selection was using the built-in feature importance metric from the Extreme Gradient Boosting model (XGBoost) for regression. The process of optimizing this model is detailed in section VI.

Figure 2 represents the feature importance obtained from XGBoost on the full dataset (containing the main table and routes table data). As we can see from the figure, only one

feature, namely $direction\_d$ (a binary feature stating whether the direction is d or not) dominates the rest of the features in the dataset.

Sometimes, these built-in feature importance estimates can be inaccurate, as we suspected in this case. The reason behind this is that XGBoost weights the features based on the frequency of splitting, gain and coverage metrics. In the case of categorical variables, the frequency of splitting can be very low, since there are few possible split points, contributing to an overall lower importance than the actual one for categorical variables. Additionally the gain for continuous features can be lower than that of the categorical ones since more split points are possible to be made which in turn can rule out fewer examples in each split compared to the categorical features.

In our case we are creating shallow individual decision trees as weak learners. This means that we have fewer levels in each tree, thus splitting by a binary (or categorical) feature, which is correlated with the target variable, can have a higher value for the gain compared to a continuous feature which is correlated with the target variable because the continuous feature might rule out fewer examples in each split. In turn, this can result in inaccurate calculations of the overall feature importance when using a mix of categorical and continuous features.

For this reason, we further try to extract the important features using wrapper methods [29] over the XGBoost algorithm, which is explained in the following subsection.

C. Boruta search with Shapley values

Boruta search [30] is a wrapper algorithm originally built over the Random Forest classification model, but further extended for all types of decision tree-based models and regression. The method implements feature selection by creating copies of the original features and shuffling them to remove any correlation with the target variable. These features are called shadow features. The algorithm then compares the shadow features' Z-scores [31] to the original features' Z-scores. Each feature that fails a two-sided test for significant difference of importance with the shadow feature with maximum importance is removed from the dataset. The feature importance, in this case, was measured using Shapley values [32]. The Shapley values are often used as a method for explainable AI (XAI) because they reveal the average marginal contribution of a feature value across all possible coalitions.

The results of implementing this feature selection method over the XGBoost algorithm are shown in Figure 3. A total of 48 features were identified as important, and their importance compared to the shadow features is displayed in the figure.

From the figure, we can again see that the features $euclidean\_distance$, $direction\_d$, and km_total dominate in their importance for the algorithm compared to all of the other features in the dataset, which means that their impact is most significant in determining the $expenses$ variable's values.

VI. Results

A. Linear Regression

The Linear Regression models were first experimented with by using the features in the main table and standardizing them according to the needs of the algorithm. The $ferry\_intervals$, $train\_intervals$, and $id\_service\_type$ features were scaled using the Min-Max scaling, while all other features that were continuous were scaled using the Standard scaling [33]. The root mean squared error (RMSE) of this primary model, using the train/validation split for validation, was 0.6703. The RMSE of this model on the leaderboard was 0.4598.

The same model was later modified to include only the features that are not correlated, according to the OLS [34] statistical test for feature relationships and the correlations represented in Figure 1. Using only those features, the model had a RMSE of 0.6942 on the validation data set, and a RMSE of 0.5027 on the leaderboard.

Finally, the squared values of the features km_total, max_weight and $time\_taken\_minutes$ were added to the model. This improved the model's RMSE on the validation set to 0.5713, and the RMSE of the test set to 0.4309. The coefficients and their significance are shown in Figure 4.

We can see that all of the features have significant coefficients according to the reported p-value. In this case, the total number of features for training the model was 15. Since linear regression might not estimate the coefficients right in the case of a large number of features, we decided to further go with other non-linear models that better handle a large number of features.

Moreover, we examined that the target variable is multimodal, meaning that any model which expects a Gaussian distribution of the target variable will not be suited well in this scenario. For this reason, we mainly focus on tree-based models and ensembles. We further try Gaussian Mixture distributions with neural networks, but due to the time limit, we did not have the resources to optimize these types of models.

B. Extreme Gradient Boost Regression

The first XGB Regressor model that was built only on the main table dataset included $alpha\_booster$, eta, $lambda$, and max_depth as hyper-parameters in the tuning job, using Bayesian Search [35] to find the optimal values. This resulted in an average of 0.34 RMSE on the 5-fold cross-validation, and a 0.1735 RMSE on the leaderboard.

The model was later improved with the addition of the routes table, as well as the selected features from the Boruta Shap search, which resulted in improving the RMSE of the CV to approximately 0.18 and 0.15, respectively on the 5-fold cross-validation and improving the test RMSE on the leaderboard to 0.1649 and 0.1622 respectively.

C. Random Forest

The first Random Forest model was built on the transformed features of the main table. Hyperparameter optimisation was done on the max_depth, $min\_samples\_leaf$ and

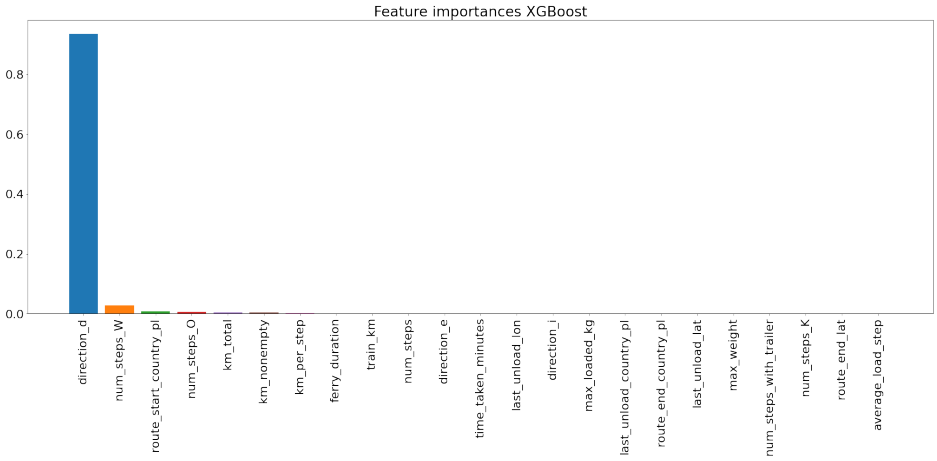

Fig. 2. **XGBoost built-in feature importance for the merged dataset.** This figure represents the feature importance of the main table and the routes table. The horizontal axis represents the features, while the vertical axis represents their corresponding impact on the prediction of the target variable. Only features having an importance greater than 10^{-5} are shown.

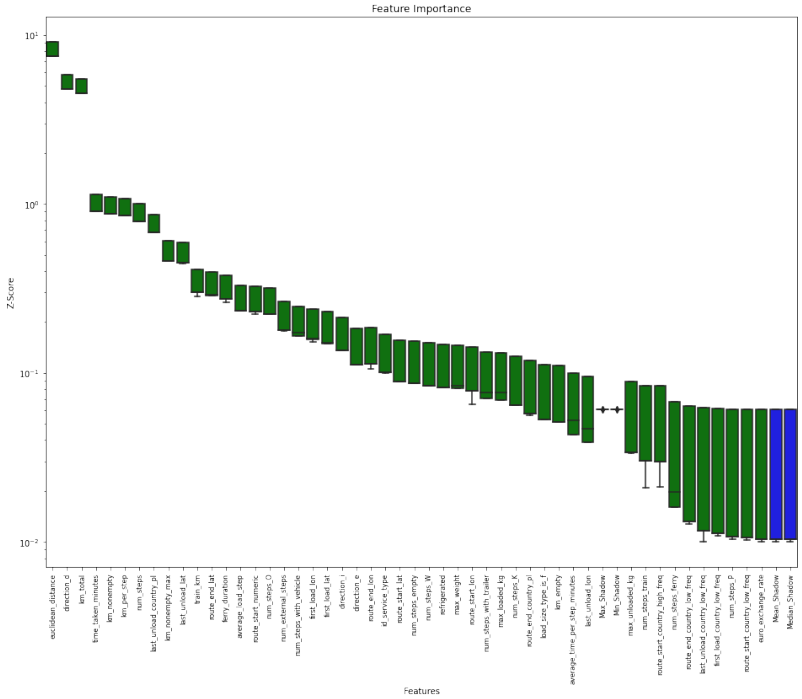

Fig. 3. **Extracted features using Boruta search with Shapley values as an evaluation metric.** The green rectangles represent the original features, while the blue rectangles represent the min, max, mean, and median shadow features. The vertical axis represents the Z-score for each feature.

$max\_features$ hyper-parameters. All Random Forest models in this competition had n_estimators set to 200. This resulted in a RMSE of 0.0476 on the 5-fold cross-validation, and a RMSE of 0.1726 on the test data.

The model was later improved by using a deeper grid search on all features from the merged main and route tables. This time, max_depth, $max\_features$, $min\_samples\_leaf$ and $min\_samples\_split$ were optimized using 5-fold cross-validation, which resulted in the validation RMSE being 0.0234. The test results had a RMSE of 0.1625.

Both Random Forest models were clearly over-fitted, however, we tackled that issue with the different Ensembles of models later on.

D. Deep Learning Models

A few feed-forward neural networks were implemented using different configurations. The neural networks were trained on the full dataset. The networks only included dense layers and the main activation function used in the hidden layers was ReLU [36]. We experimented with a few regularization

OLS Regression Results

Dep. Variable:	expenses	R-squared:	0.870
Model:	OLS	Adj. R-squared:	0.870
Method:	Least Squares	F-statistic:	8.659e+04
Date:	Sat, 14 May 2022	Prob (F-statistic):	0.00
Time:	16:54:51	Log-Likelihood:	-52431.
No. Observations:	181108	AIC:	1.049e+05
Df Residuals:	181093	BIC:	1.050e+05
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
euro_exchange_rate	0.6569	0.005	123.488	0.000	0.646	0.667
first_load_lat	0.0252	0.000	74.723	0.000	0.025	0.026
first_load_lon	-0.0048	0.000	-26.095	0.000	-0.005	-0.004
route_end_lat	0.0245	0.000	78.022	0.000	0.024	0.025
last_unload_lon	-0.0122	0.000	-77.225	0.000	-0.013	-0.012
km_empty	-0.0008	8.15e-06	-95.747	0.000	-0.001	-0.001
km_total	0.0024	4.34e-06	561.506	0.000	0.002	0.002
train_km	0.0008	3.69e-06	220.801	0.000	0.001	0.001
ferry_duration	0.0004	2.39e-06	174.749	0.000	0.000	0.000
max_weight	1.491e-06	1.36e-07	10.976	0.000	1.22e-06	1.76e-06
route_start_numeric	5.525e-05	1.57e-06	35.153	0.000	5.22e-05	5.83e-05
time_taken_minutes	8.148e-07	3.81e-07	2.136	0.033	6.71e-08	1.56e-06
km_total_2	-5.385e-07	1.73e-09	-311.493	0.000	-5.42e-07	-5.35e-07
max_weight_2	-8.619e-12	2.57e-12	-3.358	0.001	-1.36e-11	-3.59e-12
time_taken_minutes_2	-6.906e-11	4.76e-12	-14.522	0.000	-7.84e-11	-5.97e-11

Fig. 4. **Coefficients and significance of manually selected features for the linear regression algorithms including squared features.** This figure represents the coefficients of the variables included in the linear regression estimation, along with confidence intervals and p-values. All features have significant coefficients.

techniques including dropout, batch normalization, and kernel regularization with L2 [37].

From the conducted experiments, we concluded that adding regularization caused the model to underfit the training data. Moreover, batch normalization caused a significant performance degradation in this case.

For the output layer, we tried two activation functions, namely softplus [38] and linear. In this case, the same network configuration had better performance using the linear activation instead of softplus.

The model which obtained the best results had the following configuration:

- Dense(128, activation=relu)
- Dense(64, activation=relu)
- Dense(64, activation=relu)
- Dense(1, activation=linear)

The results of this model were RMSE of 0.1683 on the random validation split of 20% training data and RMSE of 0.1775 on the leaderboard, respectively.

Adding more layers results in better model performance, however, it also causes the model to overfit the data in the early stages of training.

Since the neural networks approximate a Gaussian-like distribution, they are not quite suitable for the multimodal target in this case. We additionally tried Mixture Distribution networks but did not have the resources to optimize these models. A basic model with the following configuration of two dense layers with 100 neurons and ReLU activation each for approximating the parameters of the distribution, resulted in a RMSE of 0.1868 on the leaderboard.

E. Diversified Ensemble Models

Ensemble methods [39] were used in order to compensate for models which might be overfitted or underfitted in the data, and further use the errors the models make in a way to further tune the final predictions.

One of the ensemble methods used was model stacking. For this type of ensemble, we used the best performing XGBoost model using the features extracted with the Boruta search method and additionally re-trained a Bagging model of 100 linear regressions with the same features.

The outputs from these models were then fed to another linear regression model, which learned the weights to assign to each individual model, thus creating a weighted ensemble. This approach resulted in a 0.1631 RMSE on the leaderboard and a RMSE of approximately 0.06 on the validation data. This means that the stack resulted in obvious overfitting.

We further experimented with the same approach using a decision tree in the last layer instead of a Linear regression, for learning the weights in the ensemble, however, this resulted in a more overfit model, with a RMSE of 0.1805 on the leaderboard.

Since this approach resulted in fast overfitting, we decided to abandon it.

The other method of ensembling the models that we attempted was a simple weighted Ensemble. Using a combination of the Linear Regression models, the XGBoost models, the Random Forest models, and the feed-forward neural network models, we attempted to manually adjust the weights that these models had on the final outcome. The best Ensemble was found to be an equal weights Ensemble between the highest-scoring Random Forest model, and highest scoring XGBoost model, which had a validation RMSE of 0.1318, and a test RMSE of 0.1586.

We then tried another ensemble, which used the underfitted feed-forward neural network with a weight of 0.2, the highest-scoring XGBoost model with a weight of 0.4, and the highest-scoring Random Forest model with a weight of 0.4, all trained on the features chosen with the Boruta search method. We expected that the feed-forward neural network would generally make mistakes in the opposite direction of the Random Forest and XGBoost models, and therefore contribute to the reduction of the average mistake. The weight of the feed-forward neural network model is low, however, due to its larger average errors. This resulted in an Ensemble with a validation RMSE of 0.1856, and a test RMSE of 0.1567, our best score in this competition.

F. Model evaluation

In this subsection, we present the results of the individual models which were optimized and chosen for creating ensembles in the final stage of experimentation. Table I shows the

models, their training configurations, the features they used, and their validation and leaderboard RMSE scores.

The final 3 models which were chosen for the competition include:

- Ensemble using the best performing feed-forward neural network, Random Forest, and XGBoost models
- Ensemble using only the best performing Random Forest and XGBoost models (excluding models which expect Gaussian distributions of the target variable)
- The best performing Random Forest which uses the features chosen with the Boruta search method to avoid over-smoothing or overfitting the ensemble methods

VII. Discussion and Future Work

The top-performing models were the XGBoost and Random Forest models, strongly outperforming the Linear Regression models and slightly outperforming the feed-forward neural network models. This was expected, due to the multi-modal nature of the target variable, which is hard to estimate using models that expect a Gaussian distribution of the target.

Moreover, the ensembles of diverse models performed the best out of all the predictive options. They used weights that were calculated using the inverse of the RMSE scores of the cross-validation of the models they were composed of. The singular Linear Regression models were not used in the Ensembles due to their massive underperformance compared to the other three model types. However, they were used with the bagging regressors, but this approach also underperformed compared to the non-linear approaches.

Hyper-parameter optimization on all models was performed using the grid search algorithm, with 5-fold cross-validation, and RMSE as a metric to evaluate performance. Grid search was used because it is one of the most thorough hyper-parameter tuning algorithms. Given more time, we would have expanded the search space of the grid search of all models.

According to the Boruta search for feature importance, the $euclidean\_distance$, $direction\_id$, and km_total columns were considered the most important for determining the $expenses$ value, with starting and ending locations of low-frequency destinations being some of the least important features. This further implies that the distance of the route is the most important deciding factor in the final expenses of forwarding contracts.

The main challenge in working with this dataset was the limited information we had on the meaning of some of the given features. With better information on the features, the data engineering process, as well as the model building process, would have been more specific and exhaustive.

While experimenting with the aforementioned validation procedures in section III, we noticed that some additional features could be extracted from the id_payer column, considering that it, in its original form, is not applicable as a feature. Such derived features could be:

- $num\_previous\_contracts$ - the number of previous contracts (before this contract date) for the same client (id_payer)
- $average\_cost\_previous\_contracts$ - the average cost of previous contracts (before this contract date) for the same client (id_payer)
- $average\_duration\_previous\_contracts$ - the average duration of previous contracts (before this contract date) for the same client (id_payer)
- $average\_length\_previous\_contracts$ - the average length of previous contracts (before this contract date) for the same client (id_payer)
- $ratio\_length\_previous\_contracts$ - the ratio of current length divided by the average of previous contracts (before this contract date) for the same client
- $cost\_most\_similar\_contract$ - the cost of the previous contract with the most similar length, adjusted by the difference in exchange ratios

Considering that computation of such features should be properly handled and should be closely integrated with the training-validation split process, we did not utilize them. Despite that, we believe that there is merit in further experimenting with them.

Although it was considered, fuel prices were ultimately not used in the prediction of the target variable. This was due to the uncertainty of the availability of current fuel prices, making using them a potential data leak.

Finally, the usage of external public datasets could have vastly improved the predictions of all models. Unfortunately, due to time restrictions, we were unable to properly search for, test, and use any relevant public dataset.

VIII. Conclusion

The original goal of the challenge was to use preprocessing methodologies, Machine Learning algorithms and feature selection methods, in order to most accurately predict the costs related to the execution of forwarding contracts in a transporting company. Using feature engineering techniques, as well as a weighted diversified ensemble of XGBoost, Random Forest, and deep learning models (a feed-forward neural network), we were able to predict the expenses of the forwarding contracts with a RMSE of 0.1573.

In this paper, all missing data was imputed using mean filling and median filling. However, in the future, more sophisticated methods for data imputation can be utilized, such as Multiple Imputation by Chained Equations [40] or Regression Imputation [41].

In a broader context, we can conclude that in real-life business problems, domain knowledge and information are essential. With manual feature extraction that reflects the domain knowledge, valuable features could be created that improve the model performance. Likewise, without the domain knowledge, the model validation from a practicality and explainability perspective could be limited.

Acknowledgement

This work was partially financed by the Faculty of Computer Science and Engineering at the Ss.Cyril and Methodius University, Skopje, Macedonia.

REFERENCES

[1] E. Zdravevski, P. Lameski, C. Apanowicz, D. Slezak, From big data to business analytics: The case study of churn prediction, Applied Soft Computing 90 (2020) 106164. doi:https://doi.org/10.1016/j.asoc.2020.106164.

[2] J. Bi, C. Zhang, An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme, Knowledge-Based Systems 158 (2018) 81–93. doi:https://doi.org/10.1016/j.knosys.2018.05.037.

[3] M. Grzegorowski, E. Zdravevski, A. Janusz, P. Lameski, C. Apanowicz, D. Slezak, Cost optimization for big data workloads based on dynamic scheduling and cluster-size tuning, Big Data Research 25 (2021) 100203. doi:https://doi.org/10.1016/j.bdr.2021.100203.

[4] E. Zdravevski, P. Lameski, A. Kulakov, S. Filiposka, D. Trajanov, B. Jakimovski, Parallel computation of information gain using hadoop and mapreduce, in: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2015, pp. 181–192.

[5] A. Janusz, A. Jamiołkowski, M. Okulewicz, Predicting the costs of forwarding contracts: Analysis of data mining competition results, in: Proceedings of the 17th Conference on Computer Science and Intelligence Systems, FedCSIS 2022, Sofia, Bulgaria, September 4-7, 2022, IEEE, 2022.

[6] A. Janusz, G. Hao, D. Kaluza, T. Li, R. Wojciechowski, D. Slezak, Predicting escalations in customer support: Analysis of data mining challenge results, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 5519–5526.

[7] A. Janusz, M. Przyborowski, P. Biczyk, D. Slezak, Network device workload prediction: A data mining challenge at knowledge pit, in: 2020 15th Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2020, pp. 77–80.

[8] A. Janusz, D. Kaluza, A. Chkadzynska-Krasowska, B. Konarski, J. Holland, D. Slezak, Ieee bigdata 2019 cup: suspicious network event recognition, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 5881–5887.

[9] M. Matraszek, A. Janusz, M. Swiechowski, D. Slezak, Predicting victories in video games-ieee bigdata 2021 cup report, in: 2021 IEEE International Conference on Big Data (Big Data), IEEE, 2021, pp. 5664–5671.

[10] P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation., Encyclopedia of database systems 5 (2009) 532–538.

[11] G. Behera, N. Nain, Grid search optimization (gso) based future sales prediction for big mart, in: 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), IEEE, 2019, pp. 172–178.

[12] S. Punia, K. Nikolopoulos, S. P. Singh, J. K. Madaan, K. Litsiou, Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail, International journal of production research 58 (16) (2020) 4964–4979.

[13] A. Dutta, A. Dureja, S. Abrol, A. Dureja, et al., Prediction of ticket prices for public transport using linear regression and random forest regression methods: A practical approach using machine learning, in: International Conference on Recent Developments in Science, Engineering and Technology, Springer, 2019, pp. 140–150.

[14] S.-J. Joo, H. Min, C. Smith, Benchmarking freight rates and procuring cost-attractive transportation services, The International Journal of Logistics Management (2017).

[15] Z. Yang, E. E. Mehmed, Artificial neural networks in freight rate forecasting, Maritime Economics & Logistics 21 (3) (2019) 390–414.

[16] A. Ubaid, F. Hussain, J. Charles, Modeling shipment spot pricing in the australian container shipping industry: case of asia-oceania trade lane, Knowledge-based systems 210 (2020) 106483.

[17] S. Nataraj, C. Alvarez, L. Sada, A. Juan, J. Panadero, C. Bayliss, Applying statistical learning methods for forecasting prices and enhancing the probability of success in logistics tenders, Transportation Research Procedia 47 (2020) 529–536.

[18] Z. Men, E. Yee, F.-S. Lien, D. Wen, Y. Chen, Short-term wind speed and power forecasting using an ensemble of mixture density neural networks, Renewable Energy 87 (2016) 203–211.

[19] X. Ma, J. Sha, D. Wang, Y. Yu, Q. Yang, X. Niu, Study on a prediction of p2p network loan default based on the machine learning lightgbm and xgboost algorithms according to different high dimensional data cleaning, Electronic Commerce Research and Applications 31 (2018) 24–39.

[20] C.-Y. Wang, M.-Y. Lin, Prediction of accrual expenses in balance sheet using decision trees and linear regression, in: 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI), IEEE, 2016, pp. 73–77.

[21] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, et al., Xgboost: extreme gradient boosting, R package version 0.4-2 1 (4) (2015) 1–4.

[22] S. J. Rigatti, Random forest, Journal of Insurance Medicine 47 (1) (2017) 31–39.

[23] E. Zdravevski, P. Lameski, R. Mingov, A. Kulakov, D. Gjorgjevikj, Robust histogram-based feature engineering of time series data, in: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), 2015, pp. 381–388. doi:10.15439/2015F420.

[24] E. Zdravevski, P. Lameski, A. Kulakov, S. Kalajdziski, Transformation of nominal features into numeric in supervised multi-class problems based on the weight of evidence parameter, in: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), 2015, pp. 169–179. doi:10.15439/2015F90.

[25] P.-E. Danielsson, Euclidean distance mapping, Computer Graphics and image processing 14 (3) (1980) 227–248.

[26] N. R. Chopde, M. Nichat, Landmark based shortest path detection by using a* and haversine formula, International Journal of Innovative Research in Computer and Communication Engineering 1 (2) (2013) 298–302.

[27] I. J. Myung, Tutorial on maximum likelihood estimation, Journal of mathematical Psychology 47 (1) (2003) 90–100.

[28] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise reduction in speech processing, Springer, 2009, pp. 1–4.

[29] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[30] M. B. Kursa, W. R. Rudnicki, Feature selection with the boruta package, Journal of statistical software 36 (2010) 1–13.

[31] P. Crewson, Applied statistics handbook, AcaStat Software 1 (2006) 103–123.

[32] E. Winter, The shapley value, Handbook of game theory with economic applications 3 (2002) 2025–2054.

[33] T. Jayalakshmi, A. Santhakumaran, Statistical normalization and back propagation for classification, International Journal of Computer Theory and Engineering 3 (1) (2011) 1793–8201.

[34] G. D. Hutcheson, Ordinary least-squares regression, L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research (2011) 224–228.

[35] H. A. Chipman, E. I. George, R. E. McCulloch, Bayesian cart model search, Journal of the American Statistical Association 93 (443) (1998) 935–948.

[36] A. F. Agarap, Deep learning using rectified linear units (relu), arXiv preprint arXiv:1803.08375 (2018).

[37] I. Nusrat, S.-B. Jang, A comparison of regularization techniques in deep neural networks, Symmetry 10 (11) (2018) 648.

[38] H. Zheng, Z. Yang, W. Liu, J. Liang, Y. Li, Improving deep neural networks using softplus units, in: 2015 International joint conference on neural networks (IJCNN), IEEE, 2015, pp. 1–4.

[39] T. G. Dietterich, Ensemble methods in machine learning, in: International workshop on multiple classifier systems, Springer, 2000, pp. 1–15.

[40] I. R. White, P. Royston, A. M. Wood, Multiple imputation using chained equations: issues and guidance for practice, Statistics in medicine 30 (4) (2011) 377–399.

[41] Z. Zhang, Missing data imputation: focusing on single imputation, Annals of translational medicine 4 (1) (2016).

APPENDIX A
APPENDIX 1: CONFIGURATION FOR THE INDIVIDUAL MODELS

TABLE I
RESULTS AND CONFIGURATION FOR THE INDIVIDUAL MODELS.

Id	**Algorithm**	**Params**	**Features / preprocessing**	Total Features	**Train config**	RMSE val	RMSE leaderboard
1	LinearRegression	/	[$euro\_exchange\_rate$, $first\_load\_lat$, $last\_unload\_lat$, $first\_load\_lon$, $last\_unload\_lon$, $route\_start\_lat$, $route\_start\_lon$, $route\_end\_lat$, $route\_end\_lon$, km_empty, $km\_nonempty$, $train\_km$, $ferry\_duration$, km_total, max_weight, $route\_start\_numeric$, $time\_taken\_minutes$] - STANDARDIZED [$ferry\_intervals$, $train\_intervals$, $id\_service\_type$] - MIN_MAX_SCALED select from above preprocessed features	43	validation split based on id_payer	0.6703012807	0.4598
2	LinearRegression	/	[$euro\_exchange\_rate$, $first\_load\_lat$, $first\_load\_lon$, $route\_end\_lat$,$last\_unload\_lon$, km_empty, km_total, $train\_km$, $ferry\_duration$, max_weight, $route\_start\_numeric$, $time\_taken\_minutes$]	12	validation split based on id_payer	0.6942584157	0.5027
3	LinearRegression	/	add km_total_sq, max_weight_sq and $time\_taken\_minutes\_sq$ to above features	15	validation split based on id_payer	0.5713369216	0.4309
4	LinearRegression	/	add km_total_cube, max_weight_cube and $time\_taken\_minutes\_cube$ to above features	18	validation split based on id_payer	0.5092784726	1.0127
6	XGBoost		Predicted using XGBRegressor with hyper-parameter tuning on alpha,booster, eta, lambda and max_depth.	43	validation split based on id_payer	0.34	0.1735
7	XGBoost	max_depth = 10, other default	Extracted aggregate features from routes table	73	5-fold cv RMSE averaged	0.18	0.1649
8	XGBoost	max_depth = 10, n-estimators = 1000	Feature selection with Boruta shap	48	validation split based on id_payer	0.15	0.1622
9	Feed Forward Network	Dense(64, activation=relu) Dropout(0.2) BatchNormalization() Dense(128, activation=relu) Dropout(0.2) BatchNormalization() Dense(1, activation=softplus)	Min/Max scaling all features except binary	48	BATCH_SIZE = 32 EPOCHS = 500 LEARNING_RATE = 10e-3 OPTIMIZER = Adam EARLY_STOPPING on validation RMSE patience 3 VALIDATION_SPLIT = 0.2	0.2911	0.2895
10	Feed Forward Network	Dense(128, activation=relu) Dropout(0.2) BatchNormalization() Dense(64, activation=relu) Dropout(0.2) BatchNormalization() Dense(64, activation=relu) Dropout(0.2) BatchNormalization() Dense(1, activation=softplus)	Min/Max scaling all features except binary	48	BATCH_SIZE = 16 EPOCHS = 50 LEARNING_RATE = 10e-4 OPTIMIZER = Adam EARLY_STOPPING on validation RMSE patience 3 VALIDATION_SPLIT = 02.	0.183	0.193
11	Feed Forward Network	Dense(128, activation=relu) Dense(64, activation=relu) Dense(64, activation=relu) Dense(1, activation=linear)	Min/Max scaling all features except binary	48	BATCH_SIZE = 16 EPOCHS = 50 LEARNING_RATE = 10e-4 OPTIMIZER = Adam EARLY_STOPPING on validation RMSE patience 3 VALIDATION_SPLIT = 0.2	0.1683	0.1775
12	Random Forest	max_depth = 10, min_samples_leaf = 5, max_features = 40	All features used	40	5-fold cv RMSE averaged	0.1662	0.1726
13	Random Forest	max_depth = 10, min_samples_leaf = 5, max_features = 40, n_samples = 200, min_samples_leaf = 1	Feature selection with Boruta shap	40	5-fold cv RMSE averaged	0.1594	0.1625