Model Report on

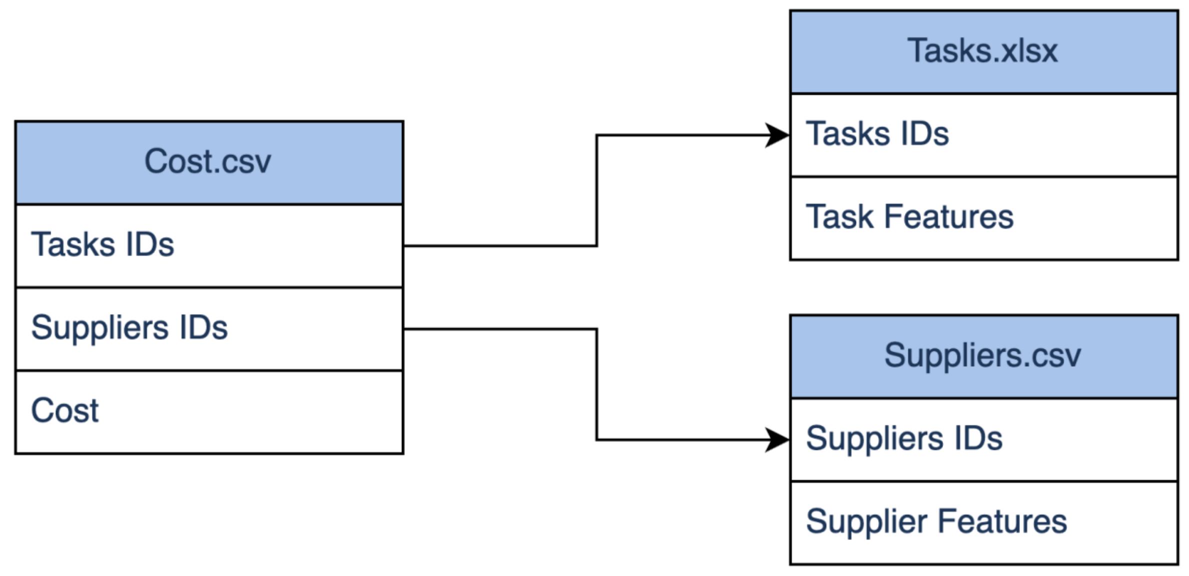
ACME’s Supplier Cost

15th December, 2022

Introduction

The provided dataset contains three files (Figure-1), the goal of this project is to identify a suitable supplier for a given TaskID. The machine learning problem is a regression problem with the cost being the response variable. The supplier with a minimal predicted cost for a given TaskID would be recommended for the task.

Figure-1. Dataset Diagram



This report presents the process and result of two machine learning models, Ridge Regression and Gradient Boosting Regressor, that tackle the Company’s problem.

Session 1: Data Preprocessing

Figure-2. Data Preprocessing Pipeline

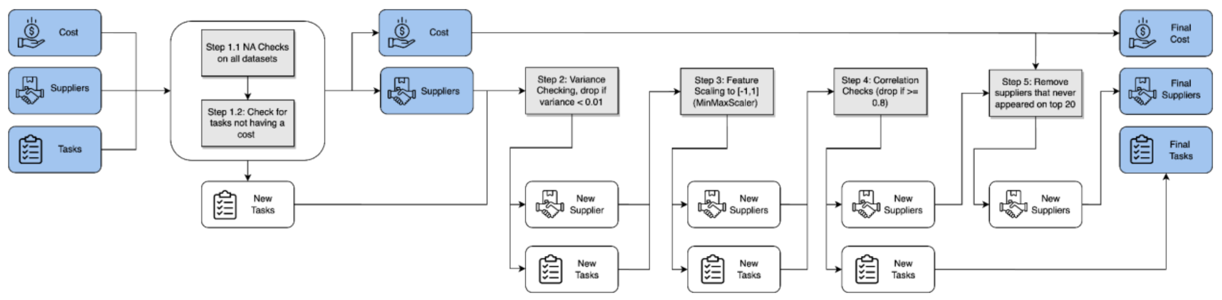


Figure-2 summarises the process for data preprocessing. Firstly, missing-data check was performed individually across three datasets. While there were no missing values identified in three datasets, 10 TaskIDs in the cost dataset were not in the tasks dataset, Table-1 provides the ten removed TaskIDs and Table-2 illustrates the remaining number of TaskIDs.

Table-1. The Removed TaskIDs

“2020 01 28”	“2020 03 09”
“2020 10 09”	“2020 10 13”
“2020 10 16”	“2020 10 19”
“2020 10 21”	“2021 03 31”
“2021 08 13”	“2021 09 16”

Table-2. Number of Task Features after removed

	Original	After
Unique TaskIDs - Tasks.xlsx	130	120
Unique TaskIDs - Cost.csv	120	120

Secondly, 33 features with variances less than 0.01 were removed because these variables were unlikely to be predictive. The dropped TFs are summarised in Table-3, and Table-4 illustrates the changing number of TaskIDs.

Table-3. The Removed Task Features (variance < 0.01)					
Task Features	Variance	Task Features	Variance	Task Features	Variance
TF5	0.0062659643	TF46	0.0006285714	TF79	0.0000000000
TF7	0.0050673319	TF47	0.0004716807	TF84	0.0000000000
TF9	0.0022830020	TF50	0.0075873319	TF88	0.0000000000
TF11	0.0022038585	TF51	0.0091962955	TF92	0.0000000000
TF13	0.0007396090	TF53	0.0067292367	TF96	0.0000000000
TF15	0.0013105812	TF57	0.0067292367	TF100	0.0000000000
TF31	0.0000000000	TF61	0.0027389286	TF104	0.0000000000
TF35	0.0083333333	TF63	0.0025653711	TF108	0.0000000000
TF38	0.0002278711	TF64	0.0027389286	TF112	0.0000000000
TF39	0.0002840266	TF66	0.0025653711	TF114	0.0039612106
TF42	0.0070843417	TF75	0.0000000000	TF116	0.0005774510

Table-4. Number of Task Features after removed based on based on variance <0.01		
	Original	After
Task Features (TF)	116	83

Thirdly, all features in Tasks and Suppliers dataset were scaled to the range of [-1,1] using MinMaxScaler() reducing the effect of scale difference between variables on the ML models.

Scaling is particularly important for Ridge Regression, as the same penalty coefficient will be applied across every feature; difference in scale could limit its effect.

Fourthly, leveraging the process in Figure-3 features pairs with high correlations (≥ 0.8) were removed from the dataset as highly correlated features could lead to multicollinearity and hinder model performance. Figure-4 and Figure-5 are the correlation heatmap before and after correlation analysis, with the specific features dropped shown in Table-5, the change in shape in TFs were shown in Table-6.

Figure-3. Process of Dropping High Absolute Correlation Features.

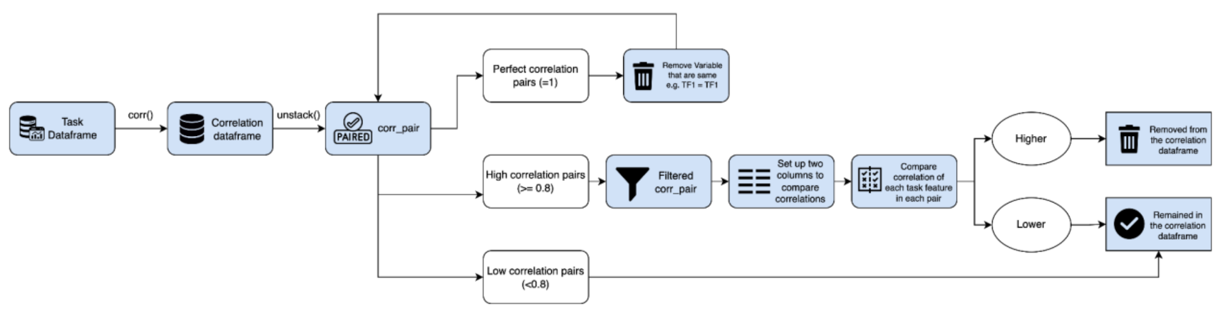


Table-5. Removed Task Features (High Absolute Correlation)								
“TF2”	“TF3”	“TF4”	“TF6”	“TF8”	“TF10”	“TF12”	“TF14”	“TF18”
“TF22”	“TF24”	“TF25”	“TF28”	“TF29”	“TF34”	“TF40”	“TF41”	“TF43”
“TF44”	“TF48”	“TF49”	“TF54”	“TF55”	“TF56”	“TF58”	“TF59”	“TF60”
“TF62”	“TF65”	“TF68”	“TF69”	“TF70”	“TF72”	“TF73”	“TF74”	“TF76”
“TF78”	“TF80”	“TF81”	“TF82”	“TF83”	“TF86”	“TF87”	“TF89”	“TF90”
“TF93”	“TF94”	“TF95”	“TF97”	“TF98”	“TF99”	“TF101”	“TF102”	“TF106”
“TF107”	“TF110”	“TF113”						

Table-6. Number of Task Features after removed based on high absolute correlation		
	Original	After

Task Features (TFs)	83	26

Figure-4. Absolute Correlation of All Pairs of Features

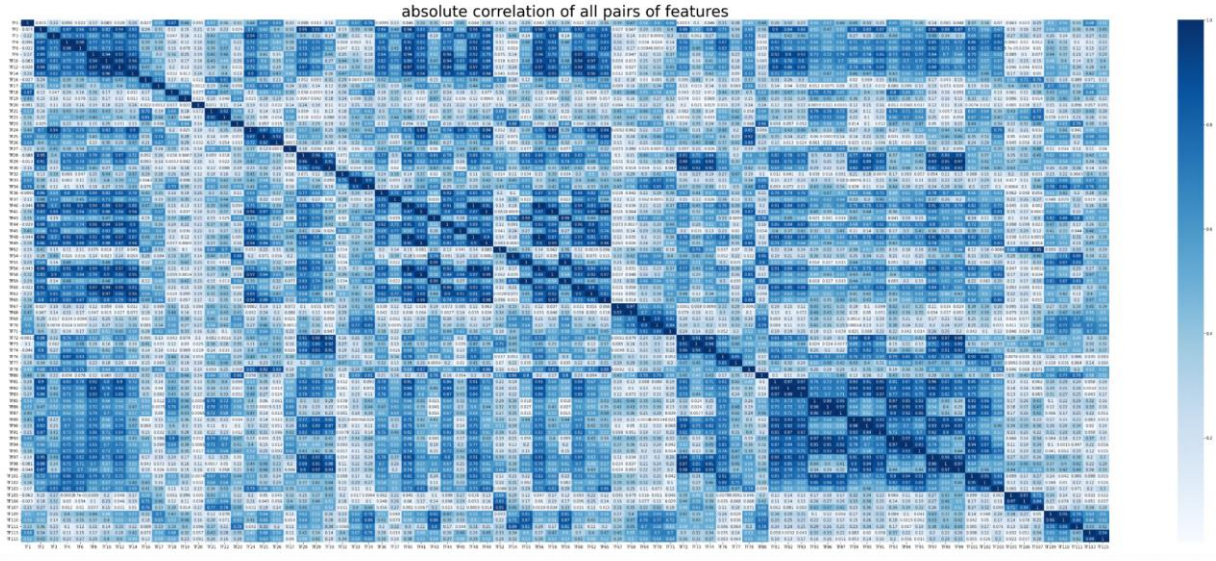
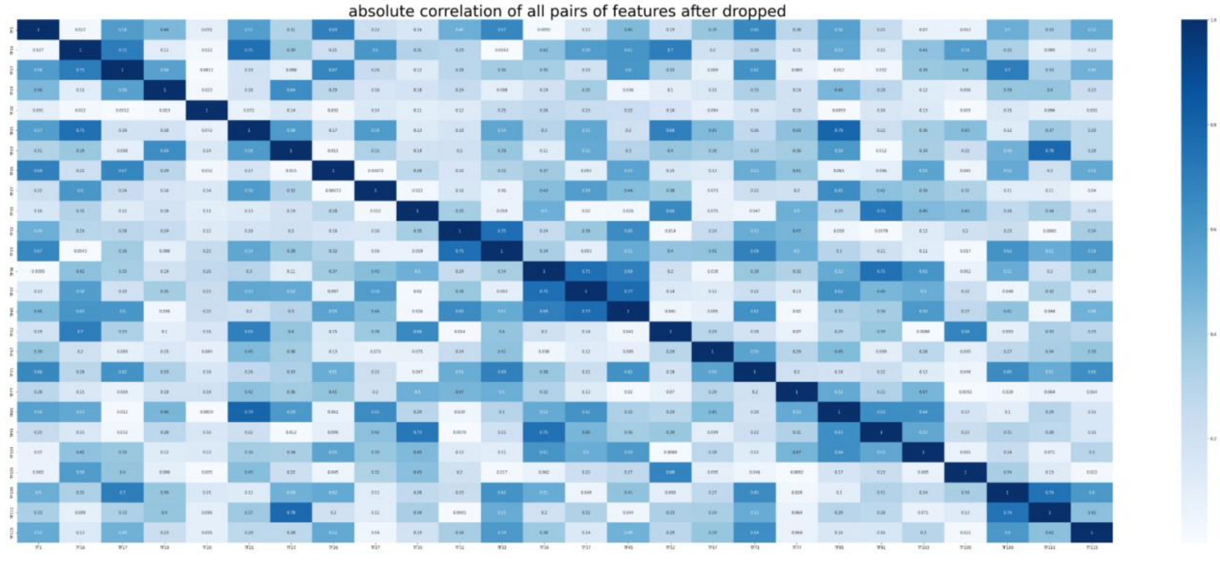
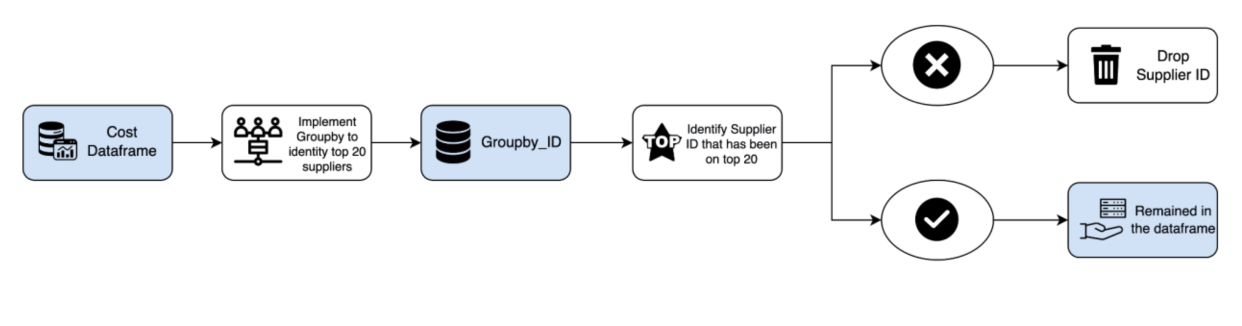


Figure-5. Absolute Correlation of All Pairs of Features after Dropped



Finally, the top twenty suppliers with the lowest costs were identified for each task. Suppliers that have never been in the top twenty would be removed because it is likely it will never be selected. Among 64 suppliers, Supplier 2 (S2) was the only supplier that has never been in the top twenty and was removed. The dropping process is shown in Figure-6.

Figure-6. Process of Dropping Suppliers



Section 2: Exploratory Data Analysis (EDA)

In the EDA section, we created an array of plots to explore the relations and trends among suppliers, cost values, and features in the given data.

Figure-7. The Distribution of Feature Values for Each TaskID

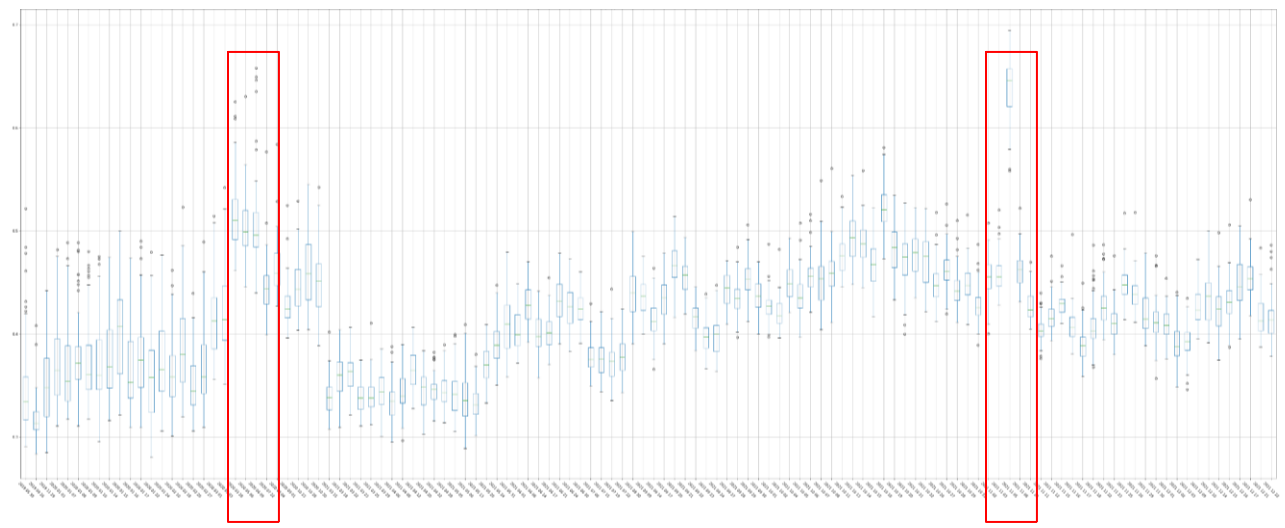
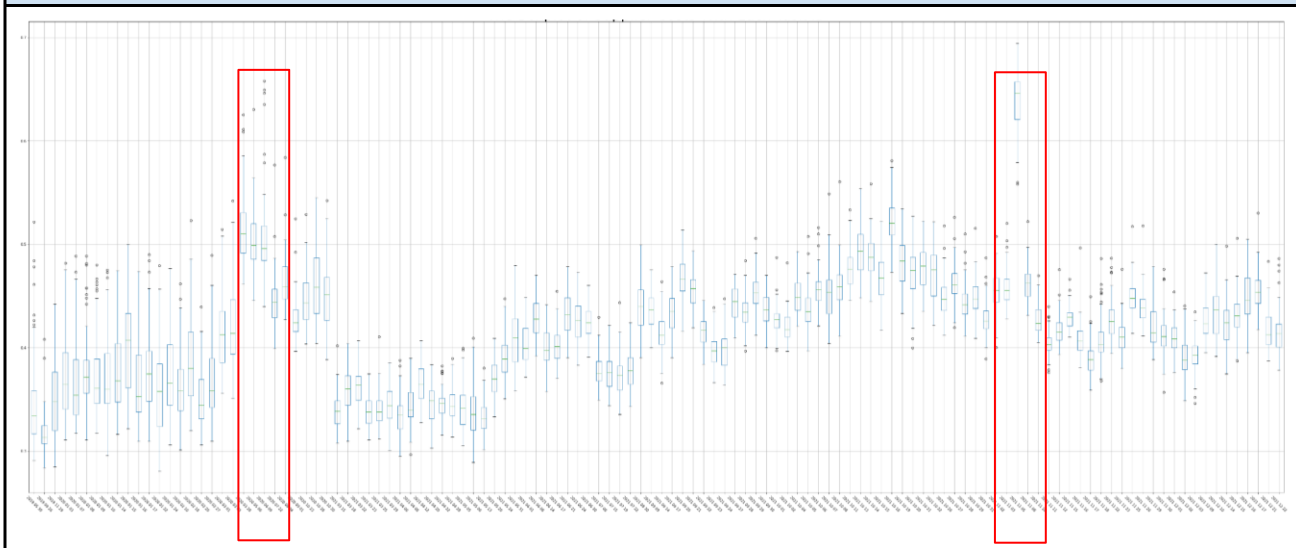


Figure-8. The Distribution of Costs of Each TaskID



The comparison was performed between two boxplots that illustrate the distribution of TFs and the cost information of suppliers performing each TaskID; the two peak points in Figure-8 correspond to the two lowest points in the TF distribution boxplot (Figure-7).

Figure-9. The Cost Values as A Matrix of Tasks and Suppliers

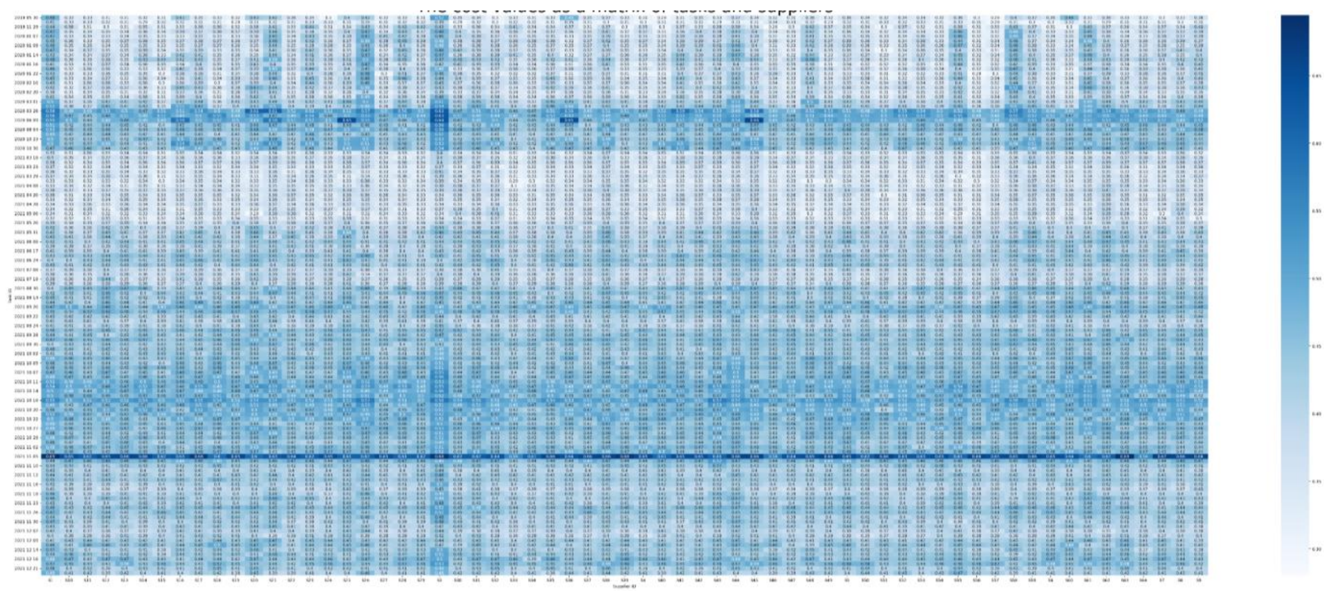


Figure-10. Suppliers’ Costs for each TaskID

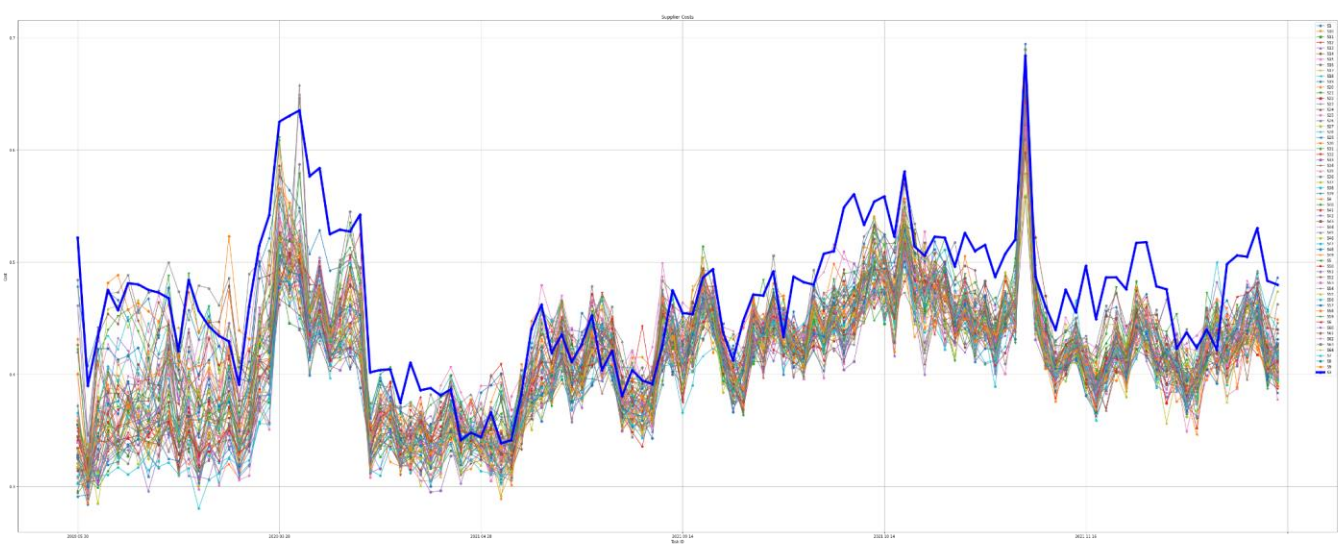
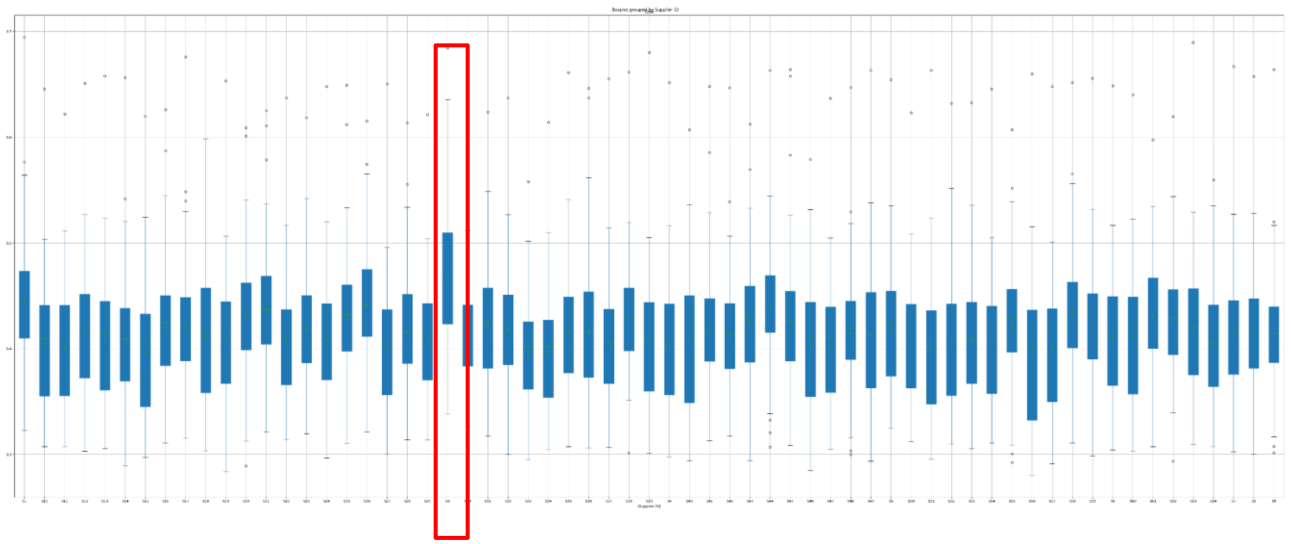


Figure-11. The Distribution of Suppliers' Costs for each TaskID



Further analysis was performed to identify the difference of suppliers' costs across different tasks leveraging heatmap (Figure-9), line plot (Figure-10) and boxplot (Figure-11). From those three graphs, most suppliers have similar costs across all tasks, with Supplier 3 being the exception and having significantly higher cost than other suppliers.

Figure-12. The Distribution of Error Term and RMSE for Each Supplier

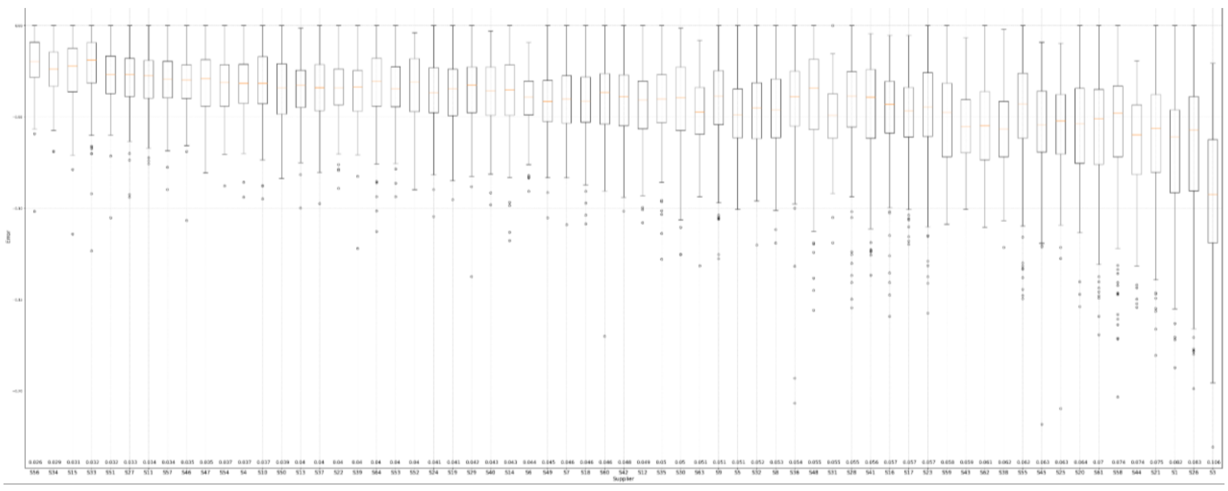
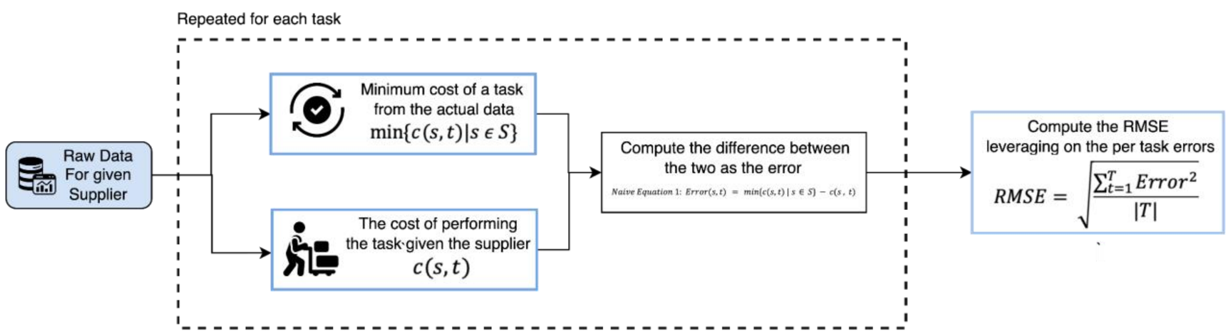


Figure-13. Process of Computing per Supplier RMSE (Naive Model)



A Naive model that assumes each supplier selected to perform each task was computed. According to Figure-13, leveraging Naive Error Equation-1, each column in cost (c(s, t)) was replaced with c(s't , t). With error for each observation calculated as the difference between cost (c(s, t)) and the minimum cost of each task. A dataframe that contains an error column for each TaskID performed by each supplier can be retrieved and was further processed by Equation-2 to calculate the RMSE for each SupplierID (Error(s, t)). Since the prediction of Naive method is based on the minimum cost of different tasks, the number of RMSE is the same as that of SupplierIDs. The Naive model's RMSE acts as a benchmark to justify whether the machine learning models are effective enough in predicting minimum costs to select the right supplier.

We created a boxplot (Figure-12) showing the distribution of Errors for each supplier and marked the RMSE of each supplier for all tasks under each boxplot to check where the RMSE of the following models were located. Table-7 is a brief summary of the RMSE values.

Equation-1: Equation-for the Naive Error

$$Naive\ Error(s,t) = min\{c(s,t) \mid s \in S\} - c(s\ ,t)$$

Equation-2: Equation-for the Score/RMSE

$$Score = \sqrt{\frac{\sum_{t=1}^{T} Error(t)^2}{|T|}}$$

Table-7. RMSE of the Naive model

The mean of RMSE	0.0499221455
The median of RMSE	0.0481400710

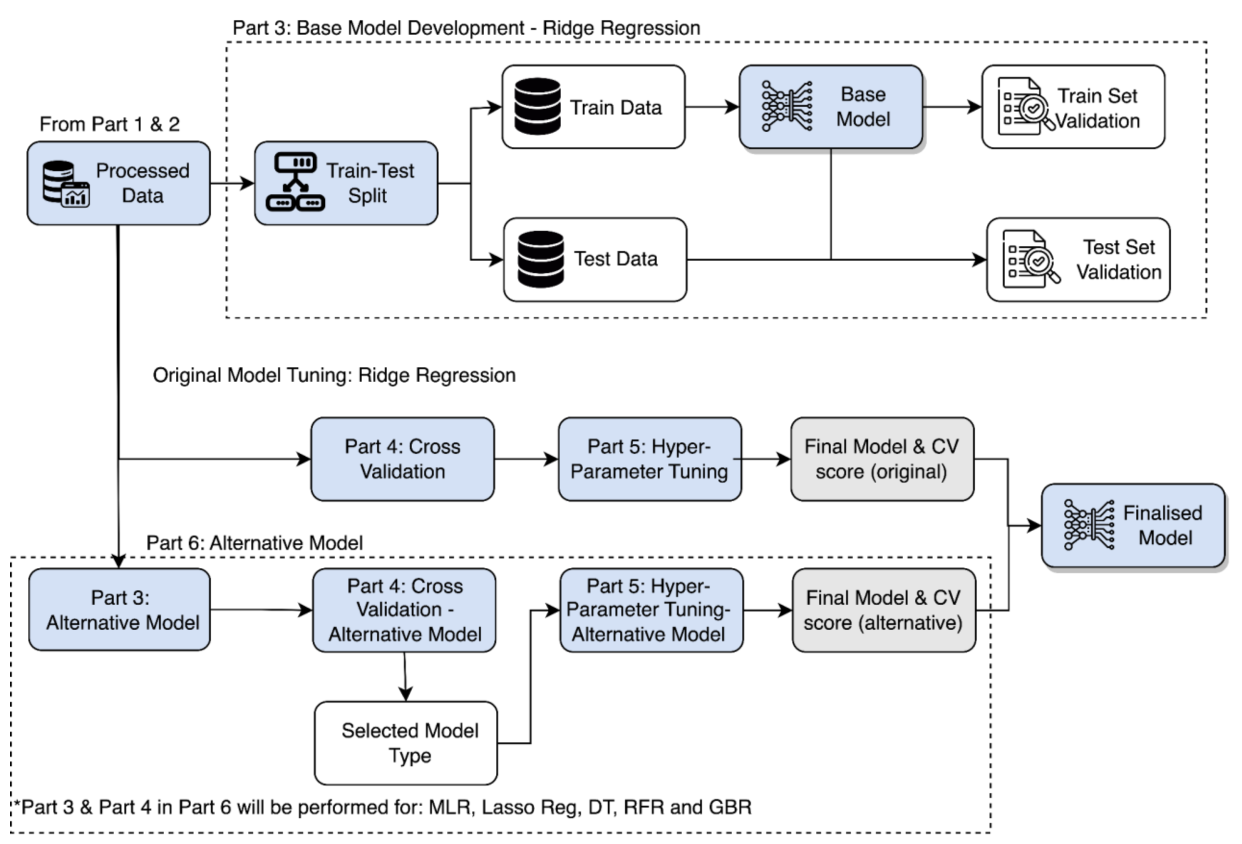
The lowest RMSE	0.0255943038
The highest RMSE	0.1062954520

Section 3: General Methodology and Justification

There are two approaches to train the data for conducting cross validation and grid search.

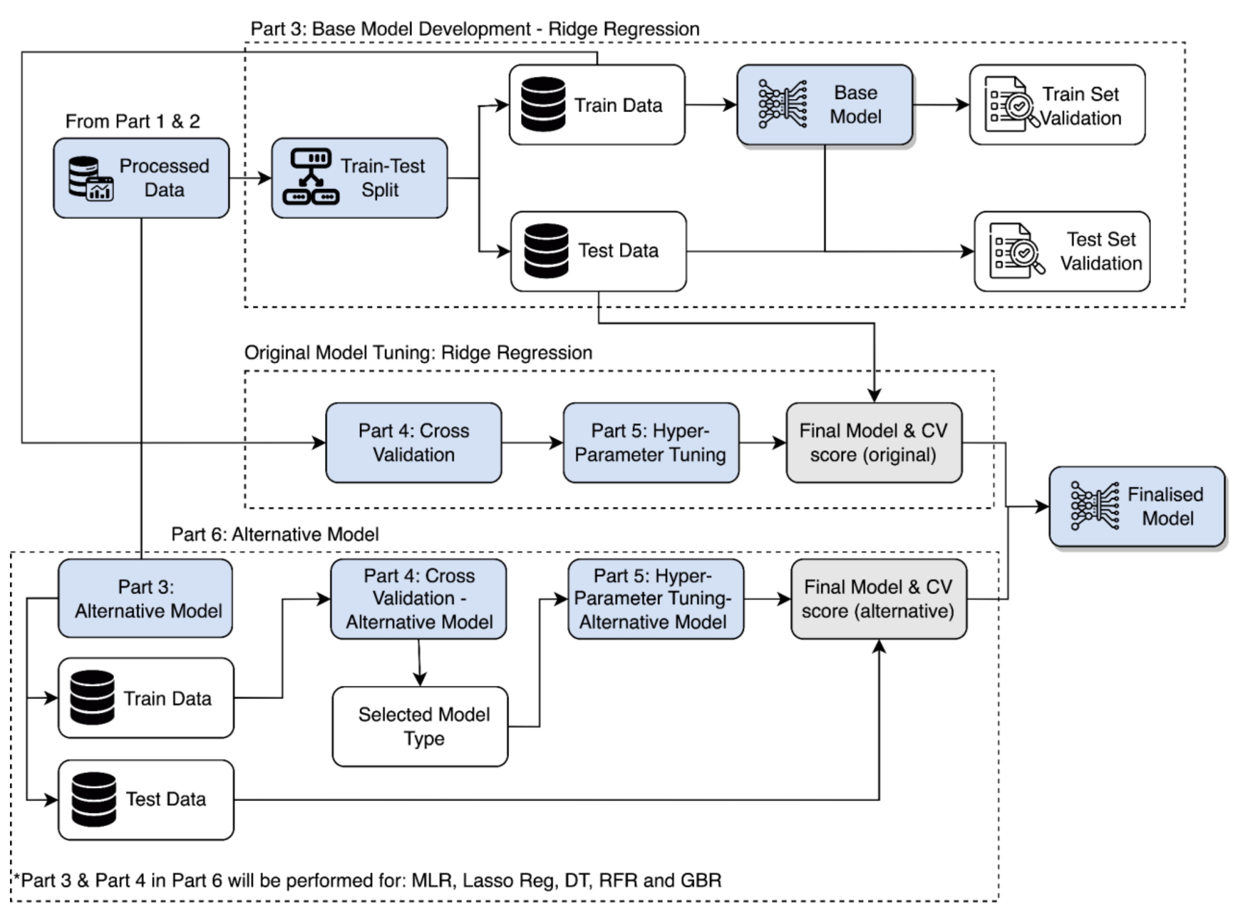
By default, the instruction recommends the implementation of approach 1 as summarised in the diagram below. Approach 1 leveraged the entire dataset (both Train and Test) during the cross-validation and hyper-parameter tuning stages instead of relying only on the Train Set (Figure-14). This approach is beneficial as it is less complex to implement and takes advantage in developing a more robust model with more data observations.

Figure-14. Modelling Approach 1



In the alternative approach, instead of passing the entire dataframe for cross validation and hyper-parameter tuning, only the Train Set was leveraged and the Test Set would be leveraged to validate out-of-sample model performance before and after tuning (Figure-15). Compared to approach 1, this approach is more robust and allows for a more comprehensive assessment of the model's performance.

Figure-15. Modelling Approach 2



In reality, approach 1 could be deemed more suitable if we expect to leverage on incoming future data as the Test Set for model validation; however, owing to the limitation of gathering further data for the current project, we stuck to approach 2.

Furthermore, instead of leveraging a single exhaustive grid search or a single randomised search for hyper-parameter tuning, we performed multiple rounds of grid search in a wide-to-deep manner. The range of each round of the hyper-parameters we tuned was shrunk, and the exhaustive space became more granular. Such an approach was leveraged because it was impossible to do a granular exhaustive grid search owing to the time constraints, computation constraints of the project and the possibilities of missing out on ideal parameters from the randomised search.

Section 4: Base Model - Ridge Regression

A base model was developed with the Train Set using Ridge Regression with default hyper-parameters. The R-squared of this model is summarised in Table-8, the R-squared value was significantly closer to 1 in the Train Set when compared to the Test Set, suggesting that the model was overfitted.

Table-8. R-squared value for the Base Ridge Regression Model	
Train Set R-squared	0.6929898348
Test Set R-squared	0.1539158259

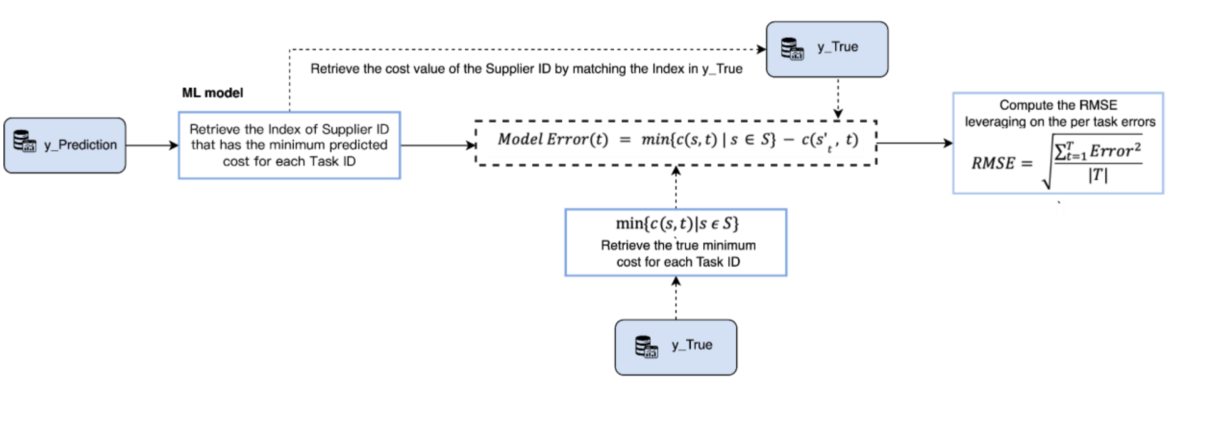
To calculate the RMSE of the Test Set to evaluate the model performance, Model Error for each task was calculated through Equation-3, and the process is shown in Figure-16.

1. The Index (SupplierIDs) of the minimum cost for each task in the predicted set was retrieved, which was then used to retrieve the corresponding true cost in the true set.
2. The true minimum cost for each task was retrieved from the true set.
3. The model Error was then calculated by Equation-3, which subtracts step 1 from step 2.
4. Model RMSE was calculated with Equation-2.

Equation-3: Equation-for the Model Error

$$Model\ Error(t) \ = \ min\{c(s,t) \mid s \in S\} - c({s'}_t \, , t)$$

Figure-16. Model Error and Model RMSE



From Table-9, the Train Set RMSE of the Base model (0.0392203752) is lower than the mean RMSE of the Naive model (0.0499221455), suggesting that the Base model has a better performance than randomly choosing suppliers for each task. However, it is still higher than the lowest RMSE of the Naive model (0.0255943038). As stated in Table-10 and Table-11, the predictions of the Train and Test Set return Supplier 37 as the chosen supplier across all TaskIDs. Hence, it shows that the prediction of the Base model is not adequate when compared to the Naive model.

Table-9. Comparison of RMSE for the Base Ridge Regression Model and Naive Model			
Train Set RMSE	0.0392203752	Naive Mean RMSE	0.0499221455
Test Set RMSE	0.0443700216	Naive Lowest RMSE	0.0255943038

Table-10. Train Set Prediction of Base Ridge Regression Model (Full Table-in Appendix 3)				
TaskID	‘2019 05 30’	‘2019 09 26’	…….	‘2021 12 22’
Chosen Suppliers	37	37	…….	37

Table-11. Test Set Prediction of Base Ridge Regression Model (Full Table-in Appendix 3)				
TaskID	‘2020 01 03’	‘2020 01 07’	…….	‘2021 12 14’
Chosen Suppliers	37	37	…….	37

Section 5: Cross Validation - Ridge Regression

Cross validation with “Leave One Group Out Method” was leveraged to assess the performance of the Base Ridge Regression Model (Pedregosa, 2011).

Figure-17. The Leave One Group Out Method

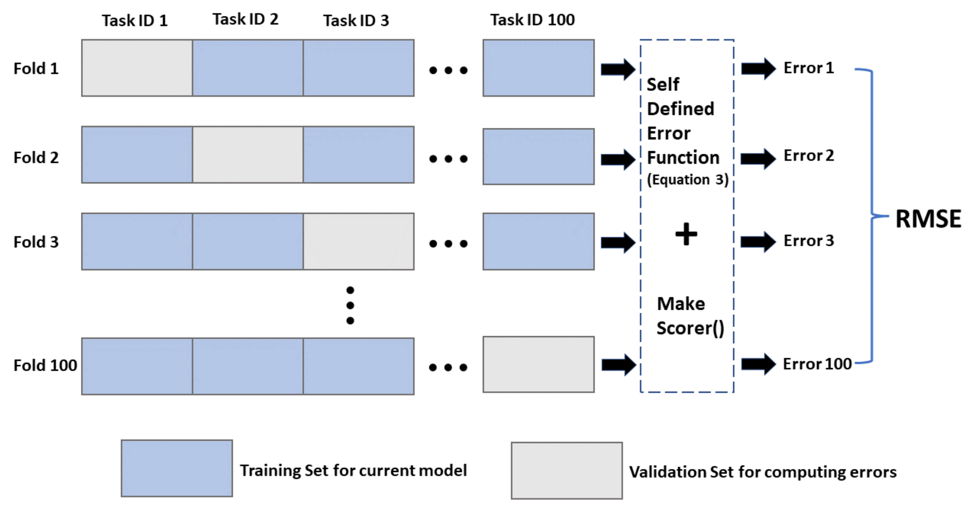


Figure-17 illustrates the leave one group out method. An observation in the Train Set was selected as the validation set, while the remaining observations act as the Train Set for the respective model. Remarkably, this process repeats until every observation was validated once.

The number of folds for this cross validation is equivalent to the number of unique groups in the dataset. In this case, the value is 100, which was derived from the unique TaskIDs of the Train Set. Each validation set was used to derive an error specific to the respective TaskID based on Equation-3. During implementation, this was achieved by designing a customised scoring function that was assigned to the ‘scoring’ parameter of the ‘cross_val_score’ function. With the 100 errors, the RMSE according to Equation-2 could be aggregated.

The final model RMSE was summarised in Table-12. Comparing results from Table-9, the Train Set RMSE from cross validation was slightly higher than the non cross validated counterpart and was noticeably lower than the Test Set RMSE.

Table-12. RMSE from Leave one group Cross Validation for base Ridge Regression	
Train Set RMSE (from Table-9)	0.0392203752
Test Set RMSE (from Table-9)	0.0443700216

Train Set RMSE (from CV, Table-16)	0.0398682855

Section 6: GridsearchCV - Ridge Regression

In typical regression, the costs will be driven more significantly by large coefficients, which might lead to overfitting. In Ridge Regression, the magnitude of coefficients is controlled by alpha (penalty term) to avoid overfitting (Datacamp, 2022). The formula is shown in Equation-4.

Equation-4: Equation-for Ridge Regression
$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2$$ *lambda(λ) is denoted as alpha(α) parameter in the Ridge Regression function

Table-13 summarises the hyper-parameters, respective default values and ranges of Ridge Regression in Sklearn.

Table-13. Common Hyperparameters for Ridge Regression		
Hyperparameter	Default Value	Range and use
alpha	1.0	Float from [0, inf); Constant that multiplies the L2 term and control regression strength Remark: when alpha = 0, it is equivalent to a MLR model
sample_weight	none	Array of Float; Individual weights for each data sample (weighted regression)
solver	‘auto’	string: {‘auto’, ‘svd’, ‘cholesky’, ‘lsqr’, ‘sparse_cg’, ‘sag’, ‘saga’, ‘lbfgs’}; specifies the solver used
max_iter	1) none 2) 1000: “sag” & “saga” solver 3) 15000: “lbfgs” solver	int; maximum number of iteration for solver

Alpha and solver are the hyper-parameters that were selected to be tuned using GridsearchCV.

Alpha is a hyper-parameter that penalises the coefficients in order to control the impact on the magnitude of the coefficients. The larger the value of alpha, the smaller the size of coefficients, which reduces the model complexity and overfitting as identified in the sections above.

Solver is a hyper-parameter for computational routines. In GridsearchCV, except ‘lbfgs’, all the solvers were set because ‘lbfgs’ can only be used when positive, a hyper-parameter, is set to true, which forces the coefficients to be positive. However, in this case, the coefficients are not necessary to be positive.

According to Table-14, Figure-18 and Figure-19, based on the smallest RMSE, there are five smallest RMSEs with the same alpha but different solvers in the first grid of grid search, so the default solver was selected as the best parameter. In the second grid of grid search, there are two alphas having the same RMSE value, so the smaller alpha was selected as the best parameter.

Table-14. Summary of Results of Hyper-parameter Optimisation using GridsearchCV

Round of GridsearchCV	Range of Alpha	Best Parameter	Lowest RMSE
1	[1e-5, 1e-3, 1e-1, 1e1, 1e3, 1e5]	alpha=100000.00000 solver=’auto’	0.0353020000
2	[1000, 2500, 5000, 7500, 10000, 25000, 50000, 75000, 100000, 250000, 500000]	alpha=250000 solver=’lsqr’	0.0333150000

Figure-18. Plot of First Round of Hyper-parameter Optimisation

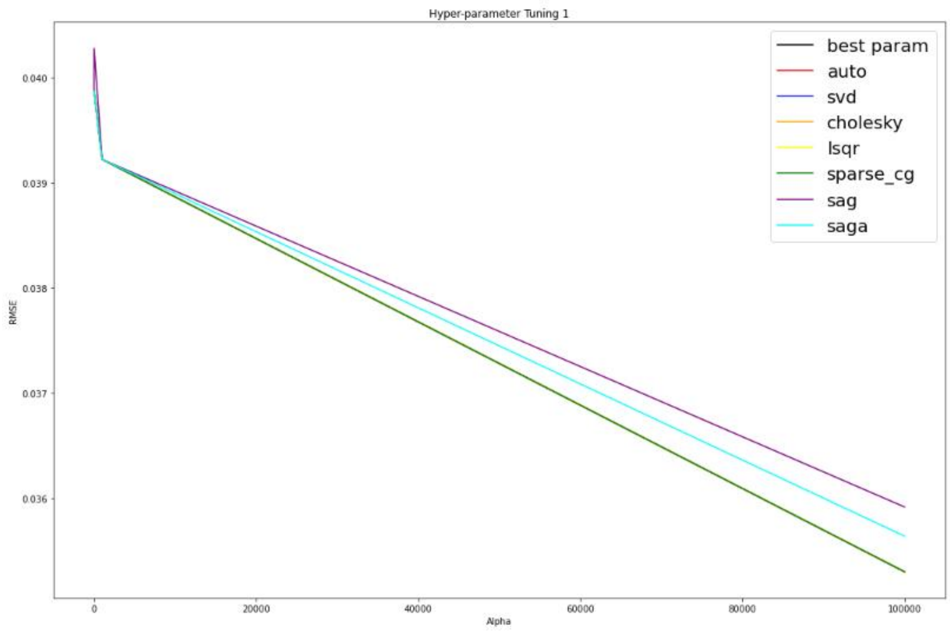
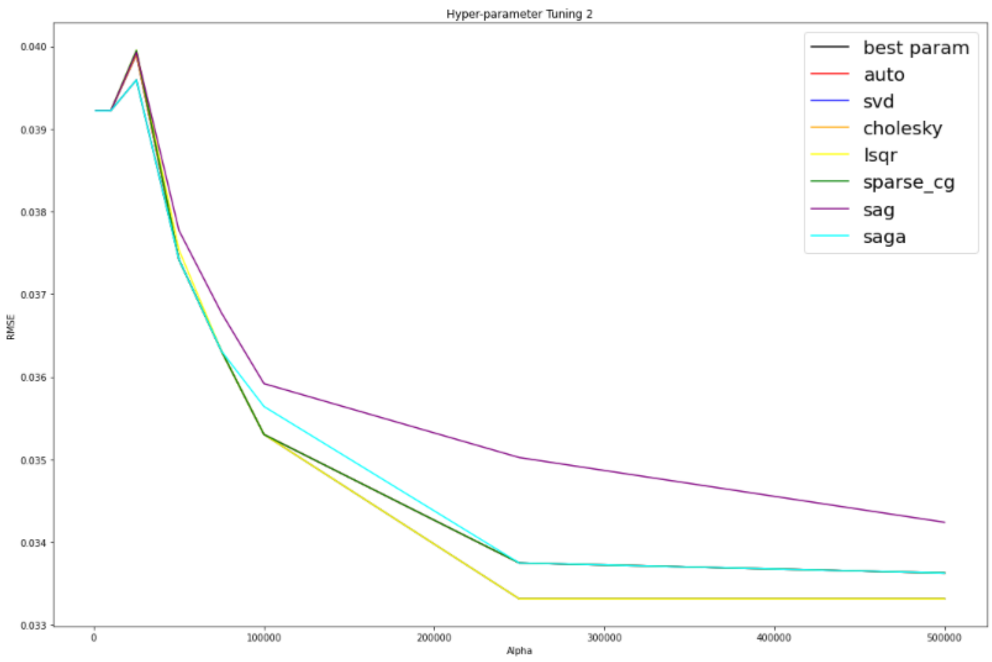


Figure-19. Plot of Second Round of Hyper-parameter Optimisation



The Base model RMSE after hyper-parameter optimisation was summarised in Table-15.

Table-15. RMSE from Hyper-parameter Optimisation for Base Ridge Regression Model	
Train Set RMSE	0.0320298984
Test Set RMSE	0.0332723884

Table-16. Comparison of RMSE among Naive Model, Base Ridge Regression, Base Ridge Regression after Cross-validation, and Base Ridge Regression after Hyper-parameter Optimisation		
Naive Model	Mean RMSE	0.0499221455
	Lowest RMSE	0.0255943038
Base Ridge Regression	Train Set RMSE	0.0392203752
	Test Set	0.0443700216
Base Ridge Regression after Cross-validation	Train Set RMSE	0.0398682855
Ridge Regression after Hyper-parameter Optimisation	Train Set RMSE	0.0320298984
	Test Set RMSE	0.0332723884

According to Table-16, RMSE has been improved after hyper-parameter optimisation. However, the predictions of the Train and Test Set return Supplier 49 as the chosen supplier across all TaskIDs (Table-17 and Table-18). Hence, it shows that the prediction of the Base model after hyper-parameter optimisation is still not suitable for this dataset. Thus, other models are required to conduct more accurate predictions.

Table-17. Train Set Prediction of Base Ridge Regression Model (Full Table-in Appendix 4)				
TaskID	‘2019 05 30’	‘2019 09 26’	…….	‘2021 12 22’
Chosen Suppliers	49	49	…….	49

Table-18. Test Set Prediction of Base Ridge Regression Model (Full Table-in Appendix 4)				
TaskID	‘2020 01 03’	‘2020 01 07’	…….	‘2021 12 14’
Chosen Suppliers	49	49	…….	49

Section 7: Alternative Model

5 model algorithms were compared to identity which is better in predicting the minimum cost (Table-19).

Table-19. Five Alternative Models
Lasso Regression: Ridge Regression is similar to Lasso Regression, except that Lasso Regression can shrink the useless covariates to exactly 0, but Ridge Regression can only shrink to nearly 0. It was picked to check whether the tiny difference could make an improvement in prediction.
Multiple Linear Regression (MLR): From the result of Section 6, when alpha is set to 0, it returns high RMSE. Whether MLR is suitable for the prediction is to be confirmed.
Decision Tree Regressor (DCR): Starting from DCR, whether tree-based models are better than the linear regression families is to be checked, as tree-based models can capture the non-linear relationships better.
Random Forest Regressor (RFR): RFR makes random predictions with a large number of trees and combines results through a voting process. It can reduce the possibility of overfitting issues in the DCR.
Gradient Boosting Regressor (GBR): GBR can generally generate more accurate results than RFR as it corrects the error terms in the previous tree.

As shown in Table-20, GBR has the lowest RMSE, while RFR has a close RMSE with GBR; but, the computational time is 2.8 times longer than GBR. Hence, GBR was chosen for further development due to higher predictability and shorter computational time.

Table-20. Summary of results of different model algorithms using Train Set with Leave One Group method

ML Model	Average Error	Standard Deviation	RMSE (from CV)	Computational Time
Lasso Regression	-0.0681643598	0.0383582484	0.0782159522	0.3956408500
Multiple Linear Regression	-0.0360625763	0.0169991406	0.0398682855	0.3178389072
Decision Tree Regressor	-0.0299762527	0.0185410280	0.0352469210	1.1703600883
Random Forest Regressor	-0.0218878257	0.0138186582	0.0258849808	55.3734130859
Gradient Boosting Regressor	-0.0209447868	0.0147772461	0.0256330079	19.6801970005

According to Table-21 to 23, the base GBR model has a better performance than Ridge Regression as Train Set RMSE (0.0229283478) is lower than the lowest RMSE of the Naive model (0.0255943038). Moreover, different suppliers for different TaskIDs were selected according to the result of the Base model.

Table-21. Results of Base Gradient Boosting Regressor Model	
Train Set R squared	0.8665144434
Test Set R squared	0.4825652640
Train Set RMSE	0.0229283479
Train Set RMSE (from CV)	0.0256330079
Test Set RMSE	0.0185653605

Table-22. Train Set Prediction of Base Gradient Boosting Regressor (Full Table-in Appendix 5)				
TaskID	‘2019 05 30’	‘2019 09 26’	…….	‘2021 12 22’
Chosen Suppliers	54	32	…….	13

Table-23. Test Set Prediction of Base Gradient Boosting Regressor (Full Table-in Appendix 5)					
TaskID	‘2020 01 03’	‘2020 01 07’	…….	‘2021 11 05’	‘2021 12 14’
Chosen Suppliers	54	54	…….	31	54

To find out the optimal parameters in GBR, n_estimator, learning_rate, max_depth and loss were chosen for hyper-parameter tuning (Pedregosa, 2011). Table-24 summarises the use of each hyperparameter.

Table-24. Common Hyper-parameters for Gradient Boosting Regressor		
Hyperparameter	Default Value	Range and use
n_estimators	100	Integer from [0, inf); the number of boosting stages to perform
learning_rate	0.1	Float from [0.0, inf); the rate to shrink the contribution of each tree
max_depth	3	Integer [0, inf) or None; Maximum depth of the individual regression estimators
loss	'squared_error'	string: {'squared_error', 'absolute_error', 'huber', 'quantile'}; the loss function to be optimised

There is a trade-off between n_estimator and learning_rate. The former describes the number of trees, where a larger number usually results in better performance, while the latter is the rate used to shrink the contribution of each tree. If the learning rate is lower, the number of estimators is larger since each estimator contributes less.

The max_depth limits the maximum number of nodes in trees. It was selected to find out the optimal value based on the input variables.

'squared_error', 'absoluet_error' and 'huber' were chosen in the loss function. 'Huber' refers to the combination of 'squared_error' and 'absolute_error'.

Table-25 and Table-26 are the two sets of grid search that was conducted.

Table-25. Gradient Boosting Regressor Grid Search 1	
n_estimators	[50, 100, 250, 500]
learning_rate	[0.01, 0.05, 0.10,0.5]
max_depth	[2,4,8,16]
loss	['squared_error', 'absolute_error', 'huber']

Table-26. Gradient Boosting Regressor Grid Search 2	
n_estimators	[250, 375, 500, 625]
learning_rate	[0.025, 0.05, 0.075, 0.10, 0.125]
max_depth	[2,4,6,8]
loss	['squared_error', 'absolute_error', 'huber']

Table-27. Summary of best parameters found by Grid Search for Gradient Boosting Regressor	
Train Set RMSE with Cross Validation: 0.0218527110	
n_estimators	625
learning_rate	0.1
max_depth	2
loss	absolute error

After the optimal hyper-parameters were found (Table-27), the Train Set RMSE with cross validation was improved from 0.0256330079 to 0.0218527110. Also, different suppliers for different TaskIDs were selected by the best parameter model (refer to Appendix 6).

Conclusion

Table-28. Comparison of RMSE of all models		
Naive Model	Mean	0.0499221455
	Lowest	**0.0255943038**
Base Ridge Regression	Train Set	0.0392203752
	Test Set	0.0443700216
Base Ridge Regression after Cross-validation	Train Set	0.0398682855
Ridge Regression after Hyper-parameter Optimisation	Train Set	**0.0320298984**
	Test Set	**0.0332723884**
Base Gradient Boosting Regressor	Train Set	0.0229283479
	Test Set	0.0185653605
Base Gradient Boosting Regressor after Cross-validation	Train Set	0.0256330079
Gradient Boosting Regressor after Hyper-parameter Optimisation	Train Set	**0.0203037809**
	Test Set	**0.0186569994**

In terms of fitting the data, as shown in Table-28, GBR has the lowest RMSE (0.0203037809) in the Train Set with best parameters. Also, when using the best parameters of GBR to predict the Test Set result in RMSE (0.0186569994), which is lower than the RMSE of Supplier 56 (0.0255943038) in the Naive model and the Test Set RMSE with best parameters in Ridge Regression (0.0332723884).

Appendix 1. References List

Datacamp (2022). Lasso and Ridge Regression Tutorial. *Datacamp*. Available at: https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression/ (Accessed 01/12/2022).

Pedregosa, F *et al*. (2011). Sklearn.linear_model.ridge_regression. *Scikit Learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ridge_regression.html/ (Accessed 01/12/2022).

Pedregosa, F *et al*. (2011). Sklearn.ensemble.GradientBoostingRegressor. *Scikit Learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html/ (Accessed 01/12/2022).

Pedregosa, F *et al*. (2011). Sklearn.model_selection.LeaveOneGroupOut. *Scikit Learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneGroupOut.html/ (Accessed 02/12/2022).

Appendix 2. Absolute Correlation Comparison between Task Features

Pairs	Task Features	Absolute Correlation*[1]	Notes
Pair 1	TF55	0.8110425793977414	Removed
	TF59	0.8110425793977413	
Pair 2	TF40	0.9988909994467344	Removed
	TF56	0.9981219776209505	
Pair 3	TF56	0.9740016663889063	Removed
	TF48	0.9653900264073014	
Pair 4	TF41	0.9953025812914736	Removed
	TF58	0.9924076400761431	
Pair 5	TF49	0.9840572681250679	Removed
	TF58	0.9729247808545908	
Pair 6	TF82	0.9747984278920591	Removed
	TF83	0.9667481755438203	
Pair 7	TF43	0.8770427843202940	Removed

1

	TF59	0.8672675064696489	
Pair 8	TF72	0.9799466045628048	Removed
	TF29	0.9719395352304223	
Pair 9	TF4	0.9180700006753050	Removed
	TF6	0.9086694408083391	
Pair 10		0.9691237599528224	Removed
	TF85	0.9688982731368679	
Pair 11	TF90	0.9406735452337999	Removed
	TF89	0.9344146593479390	
Pair 12	TF60	0.9732019325936954	Removed
	TF12	0.9693342402176274	
Pair 13	TF99	0.9719395352304223	Removed
	TF98	0.9597886515513453	
Pair 14	TF58	0.9661670438860596	Removed
	TF65	0.9405954853615561	
Pair 15	TF12	0.9653900264073014	Removed
	TF8	0.9593624229781221	

Pair 16	TF93	0.9679071752970413	Removed
	TF85	0.9552869419446401	
Pair 17	TF68	0.7759135242899322	Removed
	TF67	0.6543455885133369	
Pair 18	TF10	0.9649227941161401	Removed
	TF62	0.9629941022134231	
Pair 19	TF83	0.9660058475505151	Removed
	TF81	0.9573961637634248	
Pair 20	TF106	0.8448751815142280	Removed
	TF105	0.7078074112230371	
Pair 21	TF62	0.9128205690290812	Removed
	TF14	0.8982826576840698	

Pair 22	TF110	0.8672675064696489	Removed
	TF109	0.8060989587805196	
Pair 23	TF97	0.9573961637634248	Removed
	TF98	0.9456057650858147	
Pair 24	TF8	0.9248964665292476	Removed

	TF44	0.9240239466032660	
Pair 25	TF87	0.9552869419446401	Removed
	TF95	0.9258520067944824	
Pair 26	TF2	0.9538685131307588	Removed
	TF48	0.9087936947445289	
Pair 27	TF29	0.9456057650858147	Removed
	TF28	0.8858466463658032	
Pair 28	TF107	0.7846590914780129	Removed
	TF52	0.7500899854772625	
Pair 29	TF94	0.9445369404557559	Removed
	TF85	0.9261533244875307	
Pair 30	TF81	0.9344146593479390	Removed
	TF91	0.9329246098136401	
Pair 31	TF101	0.9261533244875307	Removed
	TF102	0.8839728344135906	
Pair 32	TF24	0.9405954853615561	Removed
	TF3	0.8684941518691267	
Pair 33	TF113	0.7631107464512419	Removed

	TF115	0.6605923690559699	

Pair 34	TF73	0.9183115655074119	Removed
	TF74	0.9149296542244406	
Pair 35	TF89	0.8965713601673037	Removed
	TF91	0.8960747043834688	
Pair 36	TF44	0.8972176371434420	Removed
	TF36	0.8488629314707686	
Pair 37	TF25	0.9163605337384625	Removed
	TF78	0.8800325576471323	
Pair 38	TF22	0.8102088356433150	Removed
	TF21	0.7932229197170784	
Pair 39	TF95	0.8839728344135906	Removed
	TF85	0.8636330601471591	
Pair 40	TF74	0.9107486967983663	Removed
	TF30	0.8817408045414112	
Pair 41	TF6	0.8722169642868608	Removed
	TF48	0.8688969880511583	
Pair 42	TF70	0.8362338524736923	Removed

	TF69	0.8221230322438442	
Pair 43	TF34	0.8074342556406171	Removed
	TF33	0.7546573617736100	
Pair 44	TF98	0.8858466463658032	Removed
	TF91	0.8333508495999413	
Pair 45	TF54	0.8764823613289116	Removed
	TF111	0.8110425793977413	

Pair 46	TF102	0.8636330601471591	Removed
	TF76	0.8499268763135125	
Pair 47	TF78	0.8797969392366368	Removed
	TF65	0.8794979179411249	
Pair 48	TF14	0.8688969880511583	Removed
	TF65	0.8684941518691267	
Pair 49	TF18	0.7454986387537825	Removed
	TF1	0.6879939602314934	
Pair 50	TF65	0.8575823480091973	Removed
	TF3	0.8512031570984033	

Pair 51	TF48	0.8488629314707686	Removed
	TF28	0.8333508495999413	
Pair 52	TF80	0.8041967658767369	Removed
	TF23	0.7783982144554177	
Pair 53	TF3	0.8277793641248382	Removed
	TF45	0.7692428545795352	
Pair 54	TF76	0.7554852740571888	Removed
	TF77	0.5184103649916632	
Pair 55	TF28	0.8268820364828429	Removed
	TF91	0.7467251401579726	
Pair 56	TF69	0.6956025674012403	Removed
	TF71	0.6888654821189323	
Pair 57	TF59	0.8060989587805196	Removed
	TF111	0.7783982144554177	
* 16 decimal points is kept to compare the value of absolute correlation.			

Appendix 3. Predictions of Base Ridge Regression Model

Train_Test	Task_ID	Selected_Supplier	Errors
train	2019 05 30	37	-0.03727

train	2019 09 26	37	-0.02878
train	2019 11 29	37	-0.01644
train	2020 01 08	37	-0.04329
train	2020 01 09	37	-0.0312
train	2020 01 10	37	-0.05819
train	2020 01 14	37	-0.04182
train	2020 01 15	37	-0.03909
train	2020 01 16	37	-0.02009
train	2020 01 22	37	-0.03324
train	2020 02 18	37	-0.02578
train	2020 02 20	37	-0.00648
train	2020 03 01	37	-0.05645
train	2020 03 03	37	-0.0519
train	2020 03 28	37	-0.04368
train	2020 05 06	37	-0.0317
train	2020 06 09	37	-0.04702
train	2020 07 22	37	-0.0299
train	2020 08 04	37	-0.01197
train	2020 09 01	37	-0.01195
train	2020 10 23	37	-0.02463
train	2021 03 18	37	-0.02535
train	2021 03 23	37	-0.02776
train	2021 03 25	37	-0.03711
train	2021 03 29	37	-0.05584
train	2021 04 08	37	-0.06361
train	2021 04 12	37	-0.04798
train	2021 04 20	37	-0.04411

train	2021 04 22	37	-0.02112
train	2021 04 28	37	-0.04593
train	2021 05 05	37	-0.0277
train	2021 05 06	37	-0.07074
train	2021 05 13	37	-0.04586
train	2021 05 20	37	-0.05227
train	2021 05 24	37	-0.03121
train	2021 05 31	37	-0.02287
train	2021 06 01	37	-0.02164
train	2021 06 08	37	-0.03683
train	2021 06 14	37	-0.05519
train	2021 06 17	37	-0.03426
train	2021 06 21	37	-0.03198
train	2021 06 24	37	-0.06282
train	2021 06 30	37	-0.03666
train	2021 07 15	37	-0.01864
train	2021 07 19	37	-0.02796
train	2021 07 22	37	-0.04743
train	2021 08 30	37	-0.02376
train	2021 09 09	37	-0.04823
train	2021 09 14	37	-0.0309
train	2021 09 17	37	-0.05326
train	2021 09 20	37	-0.05554
train	2021 09 21	37	-0.03008
train	2021 09 23	37	-0.03654
train	2021 09 24	37	-0.03397
train	2021 09 27	37	-0.01106

train	2021 09 28	37	-0.04865
train	2021 09 30	37	-0.0182
train	2021 10 01	37	-0.03547
train	2021 10 02	37	-0.00808
train	2021 10 04	37	-0.0003
train	2021 10 05	37	-0.00596
train	2021 10 07	37	-0.01257
train	2021 10 11	37	-0.0016
train	2021 10 13	37	-0.04775
train	2021 10 14	37	-0.02609
train	2021 10 18	37	-0.03024
train	2021 10 19	37	-0.05288
train	2021 10 20	37	-0.05934
train	2021 10 21	37	-0.04425
train	2021 10 26	37	-0.02395
train	2021 10 27	37	-0.02619
train	2021 10 28	37	-0.01891
train	2021 10 29	37	-0.04986
train	2021 11 02	37	-0.06188
train	2021 11 03	37	-0.03779
train	2021 11 08	37	-0.01225
train	2021 11 10	37	-0.04072
train	2021 11 11	37	-0.02451
train	2021 11 12	37	-0.0169
train	2021 11 15	37	-0.01612
train	2021 11 16	37	-0.02759
train	2021 11 17	37	-0.02495

train	2021 11 18	37	-0.04427
train	2021 11 22	37	-0.02653
train	2021 11 23	37	-0.03101
train	2021 11 25	37	-0.04734
train	2021 11 26	37	-0.02513
train	2021 11 29	37	-0.01791
train	2021 11 30	37	-0.0702
train	2021 12 01	37	-0.02796
train	2021 12 02	37	-0.01599
train	2021 12 03	37	-0.04631
train	2021 12 09	37	-0.04146
train	2021 12 10	37	-0.04565
train	2021 12 15	37	-0.0667
train	2021 12 16	37	-0.0825
train	2021 12 17	37	-0.05913
train	2021 12 21	37	-0.04406
train	2021 12 22	37	-0.03807
test	2020 01 03	37	-0.01171
test	2020 01 07	37	-0.01317
test	2020 01 17	37	-0.02462
test	2020 01 24	37	-0.02975
test	2020 02 10	37	-0.05703
test	2020 02 27	37	-0.04671
test	2020 10 28	37	0
test	2020 10 30	37	-0.03649

test	2021 03 15	37	-0.03046
test	2021 03 22	37	-0.02454
test	2021 04 06	37	-0.0523
test	2021 07 08	37	-0.02021
test	2021 09 22	37	-0.02258
test	2021 09 29	37	-0.03507
test	2021 10 06	37	-0.04345
test	2021 10 08	37	-0.037
test	2021 10 15	37	-0.03095
test	2021 10 22	37	-0.06503
test	2021 11 05	37	-0.12198
test	2021 12 14	37	-0.03271

Appendix 4. Predictions of Tuned Ridge Regression Model

Train_Test	Task_ID	Selected_Supplier	Errors
train	2019 05 30	49	-0.03315
train	2019 09 26	49	-0.04715
train	2019 11 29	49	-0.01692
train	2020 01 08	49	-0.02184
train	2020 01 09	49	-0.0119
train	2020 01 10	49	0
train	2020 01 14	49	-0.01195
train	2020 01 15	49	-0.02254
train	2020 01 16	49	-0.00622
train	2020 01 22	49	-0.02846
train	2020 02 18	49	-0.03024
train	2020 02 20	49	-0.00225
train	2020 03 01	49	-0.01399
train	2020 03 03	49	-0.03453
train	2020 03 28	49	-0.04154
train	2020 05 06	49	-0.05325
train	2020 06 09	49	-0.05559
train	2020 07 22	49	-0.0299
train	2020 08 04	49	-0.03074
train	2020 09 01	49	-0.04307
train	2020 10 23	49	-0.02516
train	2021 03 18	49	-0.03473

train	2021 03 23	49	-0.01424
train	2021 03 25	49	-0.02094
train	2021 03 29	49	-0.05619
train	2021 04 08	49	-0.04055
train	2021 04 12	49	-0.01008
train	2021 04 20	49	0
train	2021 04 22	49	-0.01563
train	2021 04 28	49	-0.01318
train	2021 05 05	49	-0.03338
train	2021 05 06	49	-0.02043
train	2021 05 13	49	-0.00401
train	2021 05 20	49	-0.03983
train	2021 05 24	49	-0.02932
train	2021 05 31	49	-0.03967
train	2021 06 01	49	-0.0209
train	2021 06 08	49	-0.02315
train	2021 06 14	49	-0.03297
train	2021 06 17	49	0
train	2021 06 21	49	-0.04993
train	2021 06 24	49	-0.01171
train	2021 06 30	49	-0.03479
train	2021 07 15	49	-0.02432
train	2021 07 19	49	-0.04025
train	2021 07 22	49	-0.02658
train	2021 08 30	49	-0.01833
train	2021 09 09	49	-0.05046
train	2021 09 14	49	-0.05367

train	2021 09 17	49	-0.01829
train	2021 09 20	49	-0.07133
train	2021 09 21	49	-0.04781
train	2021 09 23	49	-0.00332
train	2021 09 24	49	-0.01962
train	2021 09 27	49	-0.02216
train	2021 09 28	49	-0.03414
train	2021 09 30	49	-0.03615
train	2021 10 01	49	-0.03141
train	2021 10 02	49	-0.01219
train	2021 10 04	49	-0.02002
train	2021 10 05	49	-0.03935
train	2021 10 07	49	-0.02648
train	2021 10 11	49	-0.04897
train	2021 10 13	49	-0.06003
train	2021 10 14	49	-0.04517
train	2021 10 18	49	-0.05041
train	2021 10 19	49	-0.02303
train	2021 10 20	49	0
train	2021 10 21	49	-0.04129
train	2021 10 26	49	-0.03612
train	2021 10 27	49	-0.02122
train	2021 10 28	49	-0.02135
train	2021 10 29	49	-0.03158
train	2021 10 31	49	-0.0327
train	2021 11 02	49	-0.0384
train	2021 11 03	49	-0.01953

train	2021 11 08	49	-0.03874
train	2021 11 10	49	-0.01162
train	2021 11 11	49	-0.02161
train	2021 11 12	49	-0.01708
train	2021 11 15	49	-0.00417
train	2021 11 16	49	-0.03064
train	2021 11 17	49	-0.02042
train	2021 11 18	49	-0.02495
train	2021 11 22	49	-0.02166
train	2021 11 23	49	-0.02969
train	2021 11 25	49	-0.02555
train	2021 11 26	49	-0.03344
train	2021 11 29	49	-0.01956
train	2021 11 30	49	-0.04567
train	2021 12 01	49	-0.03387
train	2021 12 02	49	-0.01929
train	2021 12 03	49	-0.02878
train	2021 12 09	49	-0.03242
train	2021 12 10	49	-0.05392
train	2021 12 15	49	-0.04387
train	2021 12 16	49	-0.044
train	2021 12 17	49	-0.01658
train	2021 12 21	49	-0.01649
train	2021 12 22	49	-0.02767
test	2020 01 03	49	-0.01403
test	2020 01 07	49	-0.0161
test	2020 01 17	49	-0.01523

test	2020 01 24	49	-0.00062
test	2020 02 10	49	-0.03678
test	2020 02 27	49	-0.0088
test	2020 10 28	49	-0.0167
test	2020 10 30	49	-0.01431
test	2021 03 15	49	-0.01665
test	2021 03 22	49	-0.02708
test	2021 04 06	49	-0.02801
test	2021 07 08	49	-0.03136
test	2021 09 22	49	-0.03275
test	2021 09 29	49	0
test	2021 10 06	49	0
test	2021 10 08	49	-0.03619
test	2021 10 15	49	0
test	2021 10 22	49	-0.03877
test	2021 11 05	49	-0.10527
test	2021 12 14	49	-0.04237

Appendix 5. Predictions of Base Gradient Boosting Regression Model

Train_Test	Task_ID	Selected_Supplier	Errors
train	2019 05 30	54	-0.01125
train	2019 09 26	32	-0.03195
train	2019 11 29	54	-0.0242
train	2020 01 08	54	0
train	2020 01 09	54	0
train	2020 01 10	54	-0.03326
train	2020 01 14	54	-0.00102
train	2020 01 15	54	0
train	2020 01 16	54	-0.00158
train	2020 01 22	54	0
train	2020 02 18	54	-0.00535
train	2020 02 20	54	-0.0129
train	2020 03 01	54	-0.0007
train	2020 03 03	54	-0.00463
train	2020 03 28	9	-0.03018
train	2020 05 06	9	0
train	2020 06 09	9	-0.00131
train	2020 07 22	13	-0.00127
train	2020 08 04	54	-0.01818
train	2020 09 01	54	-0.0226
train	2020 10 23	54	-0.01527
train	2021 03 18	54	0
train	2021 03 23	54	-0.02237

train	2021 03 25	54	0
train	2021 03 29	54	-0.02204
train	2021 04 08	54	-0.01986
train	2021 04 12	13	-0.02117
train	2021 04 20	54	-0.01133
train	2021 04 22	54	-0.00859
train	2021 04 28	54	0
train	2021 05 05	54	-0.00644
train	2021 05 06	54	-0.02193
train	2021 05 13	31	-0.00921
train	2021 05 20	31	-0.02589
train	2021 05 24	31	-0.01238
train	2021 05 31	31	-0.01249
train	2021 06 01	31	-0.00854
train	2021 06 08	54	-0.00646
train	2021 06 14	31	-0.01127
train	2021 06 17	31	-0.00885
train	2021 06 21	31	-0.00093
train	2021 06 24	31	-0.01802
train	2021 06 30	31	-0.01099
train	2021 07 15	54	-0.01128
train	2021 07 19	54	-0.00763
train	2021 07 22	54	-0.01855
train	2021 08 30	13	-0.00408
train	2021 09 09	9	-0.01099
train	2021 09 14	54	0
train	2021 09 17	9	-0.01262

train	2021 09 20	54	-0.03453
train	2021 09 21	54	-0.03407
train	2021 09 23	54	-0.02036
train	2021 09 24	54	-0.03804
train	2021 09 27	54	-0.02716
train	2021 09 28	54	-0.03569
train	2021 09 30	54	-0.04621
train	2021 10 01	54	-0.02377
train	2021 10 02	54	-0.01927
train	2021 10 04	54	-0.02763
train	2021 10 05	54	-0.02717
train	2021 10 07	54	-0.02882
train	2021 10 11	54	-0.01654
train	2021 10 13	54	-0.02648
train	2021 10 14	54	-0.03147
train	2021 10 18	13	-0.03098
train	2021 10 19	54	-0.02165
train	2021 10 20	54	-0.03767
train	2021 10 21	54	-0.00751
train	2021 10 26	54	-0.02513
train	2021 10 27	54	-0.05661
train	2021 10 28	13	-0.00296
train	2021 10 29	54	-0.02759
train	2021 10 31	54	0
train	2021 11 02	54	-0.05915
train	2021 11 03	54	-0.01392
train	2021 11 08	54	-0.02528

train	2021 11 10	54	-0.00852
train	2021 11 11	54	-0.0201
train	2021 11 12	54	-0.02616
train	2021 11 15	54	-0.01573
train	2021 11 16	54	-0.02262
train	2021 11 17	54	-0.01832
train	2021 11 18	54	-0.0212
train	2021 11 22	54	-0.0159
train	2021 11 23	13	-0.0149
train	2021 11 25	54	-0.00407
train	2021 11 26	54	-0.02753
train	2021 11 29	54	-0.00714
train	2021 11 30	13	-0.04353
train	2021 12 01	54	-0.01017
train	2021 12 02	54	-0.02825
train	2021 12 03	54	-0.034
train	2021 12 09	54	-0.01362
train	2021 12 10	54	-0.05232
train	2021 12 15	54	-0.03052
train	2021 12 16	13	-0.03499
train	2021 12 17	13	-0.04127
train	2021 12 21	54	-0.02839
train	2021 12 22	13	-0.03647
test	2020 01 03	54	0
test	2020 01 07	54	0
test	2020 01 17	54	-0.00701
test	2020 01 24	54	0

test	2020 02 10	54	-0.01041
test	2020 02 27	54	-0.01387
test	2020 10 28	54	-0.03918
test	2020 10 30	54	-0.02332
test	2021 03 15	54	-0.00697
test	2021 03 22	54	-0.01212
test	2021 04 06	54	-0.01002
test	2021 07 08	31	0
test	2021 09 22	54	-0.01047
test	2021 09 29	54	-0.03358
test	2021 10 06	54	-0.01837
test	2021 10 08	54	-0.03786
test	2021 10 15	54	-0.02998
test	2021 10 22	54	-0.00393
test	2021 11 05	31	0
test	2021 12 14	54	-0.01568

Appendix 6. Predictions of Tuned Gradient Boosting Regression Model

Train_Test	Task_ID	Selected_Supplier	Errors
train	2019 05 30	54	-0.01125
train	2019 09 26	32	-0.03195
train	2019 11 29	54	-0.0242
train	2020 01 08	54	0
train	2020 01 09	54	0
train	2020 01 10	54	-0.03326
train	2020 01 14	54	-0.00102
train	2020 01 15	54	0
train	2020 01 16	54	-0.00158
train	2020 01 22	54	0
train	2020 02 18	54	-0.00535
train	2020 02 20	54	-0.0129
train	2020 03 01	54	-0.0007
train	2020 03 03	54	-0.00463
train	2020 03 28	54	-0.02237
train	2020 05 06	32	-0.01501
train	2020 06 09	54	-0.00991
train	2020 07 22	32	-0.03559
train	2020 08 04	32	0
train	2020 09 01	54	-0.0226
train	2020 10 23	31	0
train	2021 03 18	31	-0.00639
train	2021 03 23	54	-0.02237

train	2021 03 25	54	0
train	2021 03 29	54	-0.02204
train	2021 04 08	31	0
train	2021 04 12	31	0
train	2021 04 20	54	-0.01133
train	2021 04 22	54	-0.00859
train	2021 04 28	54	0
train	2021 05 05	54	-0.00644
train	2021 05 06	54	-0.02193
train	2021 05 13	54	-0.02947
train	2021 05 20	31	-0.02589
train	2021 05 24	54	-0.01724
train	2021 05 31	54	-0.02957
train	2021 06 01	54	-0.01886
train	2021 06 08	54	-0.00646
train	2021 06 14	31	-0.01127
train	2021 06 17	31	-0.00885
train	2021 06 21	31	-0.00093
train	2021 06 24	31	-0.01802
train	2021 06 30	31	-0.01099
train	2021 07 15	31	-0.01842
train	2021 07 19	31	-0.01881
train	2021 07 22	31	-0.02307
train	2021 08 30	31	-0.01718
train	2021 09 09	9	-0.01099
train	2021 09 14	54	0
train	2021 09 17	9	-0.01262

train	2021 09 20	9	-0.02658
train	2021 09 21	35	-0.00871
train	2021 09 23	31	-0.00837
train	2021 09 24	25	-0.03862
train	2021 09 27	31	-0.0099
train	2021 09 28	31	-0.00516
train	2021 09 30	31	-0.0536
train	2021 10 01	31	-0.00991
train	2021 10 02	25	-0.01238
train	2021 10 04	54	-0.02763
train	2021 10 05	9	-0.03025
train	2021 10 07	31	0
train	2021 10 11	31	-0.00892
train	2021 10 13	31	-0.01884
train	2021 10 14	54	-0.03147
train	2021 10 18	31	-0.0287
train	2021 10 19	54	-0.02165
train	2021 10 20	54	-0.03767
train	2021 10 21	54	-0.00751
train	2021 10 26	32	-0.01449
train	2021 10 27	31	-0.04577
train	2021 10 28	32	-0.01883
train	2021 10 29	25	-0.02799
train	2021 10 31	31	-0.0334
train	2021 11 02	31	-0.01872
train	2021 11 03	25	-0.01474
train	2021 11 08	31	0

train	2021 11 10	9	-0.00032
train	2021 11 11	9	-0.02639
train	2021 11 12	31	-0.02967
train	2021 11 15	9	-0.00114
train	2021 11 16	9	-0.01046
train	2021 11 17	31	-0.00794
train	2021 11 18	32	-0.00209
train	2021 11 22	9	-0.00793
train	2021 11 23	54	-0.01959
train	2021 11 25	9	-0.00869
train	2021 11 26	9	-0.01393
train	2021 11 29	54	-0.00714
train	2021 11 30	31	-0.02304
train	2021 12 01	31	-0.01277
train	2021 12 02	54	-0.02825
train	2021 12 03	31	-0.01633
train	2021 12 09	31	-0.00382
train	2021 12 10	31	-0.02142
train	2021 12 15	31	-0.02042
train	2021 12 16	31	-0.06604
train	2021 12 17	31	-0.03011
train	2021 12 21	54	-0.02839
train	2021 12 22	31	-0.02041
test	2020 01 03	54	0
test	2020 01 07	54	0
test	2020 01 17	54	-0.00701
test	2020 01 24	54	0

test	2020 02 10	54	-0.01041
test	2020 02 27	54	-0.01387
test	2020 10 28	54	-0.03918
test	2020 10 30	54	-0.02332
test	2021 03 15	54	-0.00697
test	2021 03 22	54	-0.01212
test	2021 04 06	54	-0.01002
test	2021 07 08	31	0
test	2021 09 22	35	-0.02124
test	2021 09 29	31	-0.01772
test	2021 10 06	54	-0.01837
test	2021 10 08	9	-0.03844
test	2021 10 15	54	-0.02998
test	2021 10 22	54	-0.00393
test	2021 11 05	31	0
test	2021 12 14	31	-0.02725