UCLA ECE219 - Project 4

March 6th, 2019

Authors

Ning An - 205034447
Zaurbek Tsorojev - 805029443
Endi Xu - 005030030
Minya Yao - 704161217

Introduction

The goal of this project was to learn about regression analysis. We worked on the Network Backup Dataset and applied several regression models to estimate the relationship between a target variable and a set of potentially relevant variables. Cross-validation was used to handle over-fitting, and regularization was used to penalize overly complex models.

Dataset 1 : Network Backup

Question 1 (a)

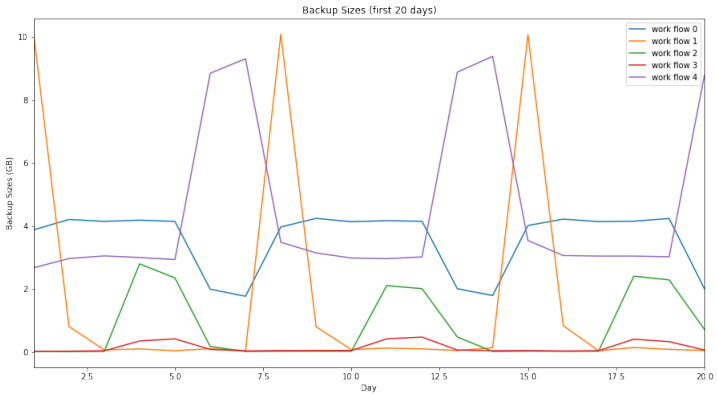


FIGURE 1 – Backup sizes for the first 20-day period

Question 1 (b)

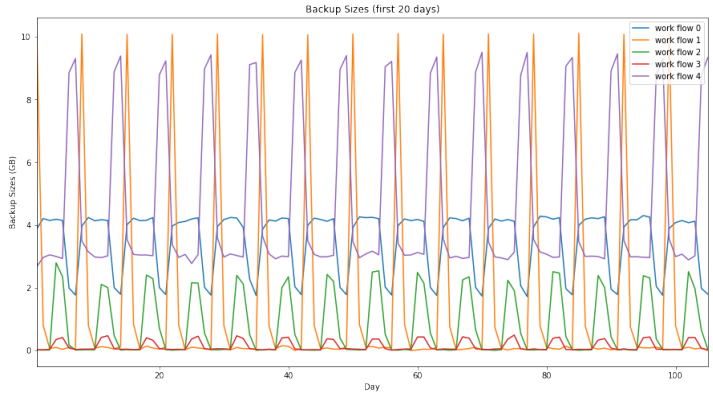


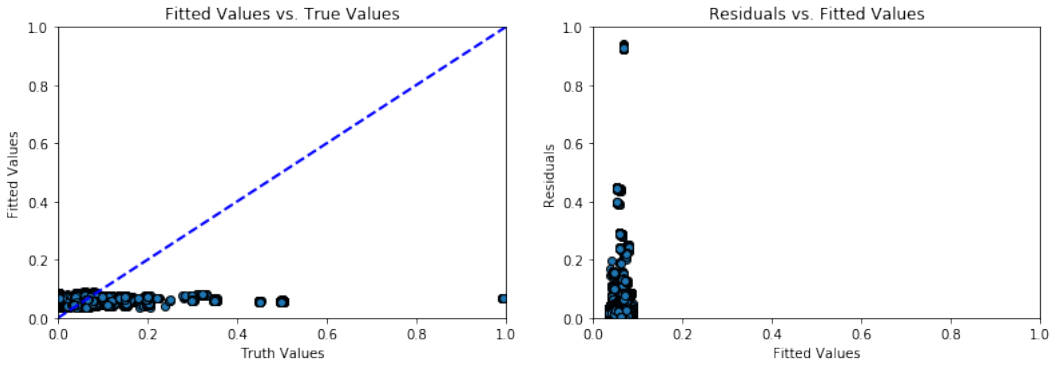
FIGURE 2 – Backup sizes for the first 105-day period

Question 1 (c)

The workflows in both 20-day and 105-day plots are periodic with a period for each workflow to be about 7 days. Workflow_1 always have the largest backup size, followed by workflow_4, workflow_0, workflow_2, and workflow_3.

Question 2 (a)

For this part, we fit a linear regression model to the dataset, and plot fitted values vs. true values and residuals vs. fitted values as scatter plots to visualize how well the model fits the dataset.



Training RMSE	Test RMSE
0.1032	0.1067
0.1039	0.1001
0.1032	0.1068
0.1039	0.1004
0.1032	0.1071
0.1039	0.1004
0.1032	0.1071
0.1039	0.1004
0.1032	0.1071
0.1039	0.0999

Average training RMSE : 0.10358
Average test RMSE : 0.10367

From the plots, we observe that the predicted value using the basic linear regression model centered at around 0.061 and the residuals centered at around 0.064. This implies that our model does not predict very well, as ideally we would want the points from the first plot to lie roughly at the line y = x, and the points (residuals) from the second plot to be as close to 0 as possible.

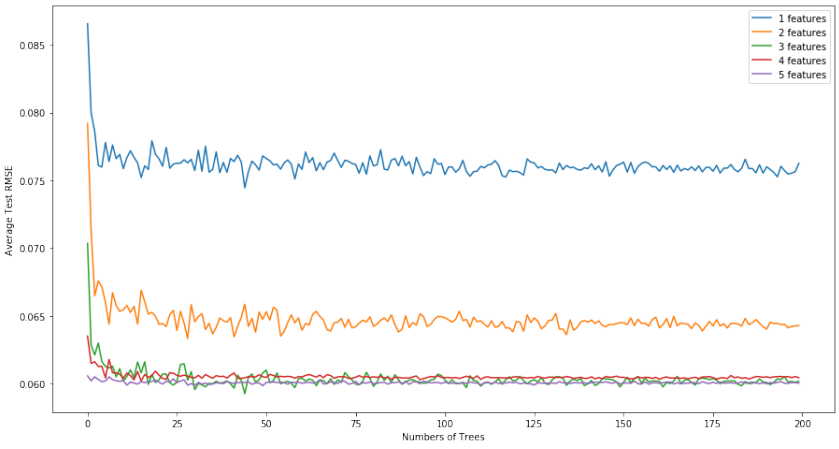
Question 2 (b)

For this part, we fit a random forest regression model to the dataset. We report the training and average test RMSE from 10 fold cross validation and the Out of Bag errors for the initial setting (using number of trees = 20, depth of each tree = 4, and maximum number of features = 5) below.

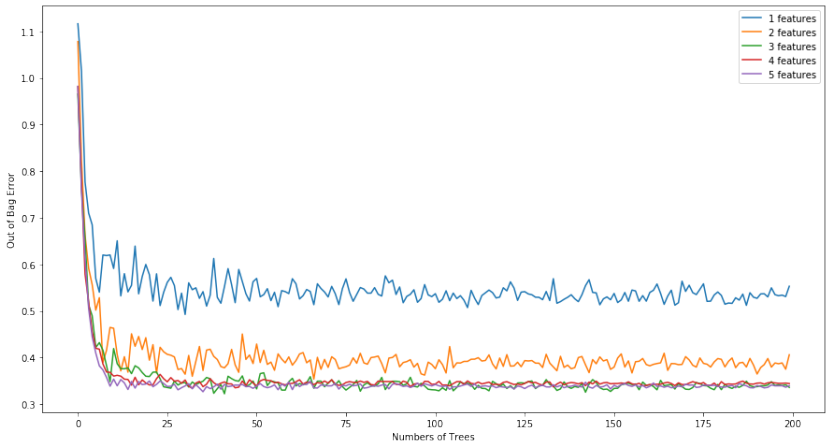
Training RMSE	Test RMSE
0.0601	0.06762
0.0611	0.05276
0.06015	0.06748
0.0602	0.05221
0.0601	0.06745
0.06096	0.05395
0.06012	0.06774
0.05936	0.05127
0.06002	0.06722
0.06077	0.05275

Average training RMSE : 0.060304
Average test RMSE : 0.060514
Out of bag error : 0.340681

Next, we try to obtain the best combination of the hyperparameters for our model. We sweep number of trees from 1 to 200 and maximum number of features from 1 to 5, and plot the average test RMSE and Out of Bag error vs. number of trees using each of the five features.

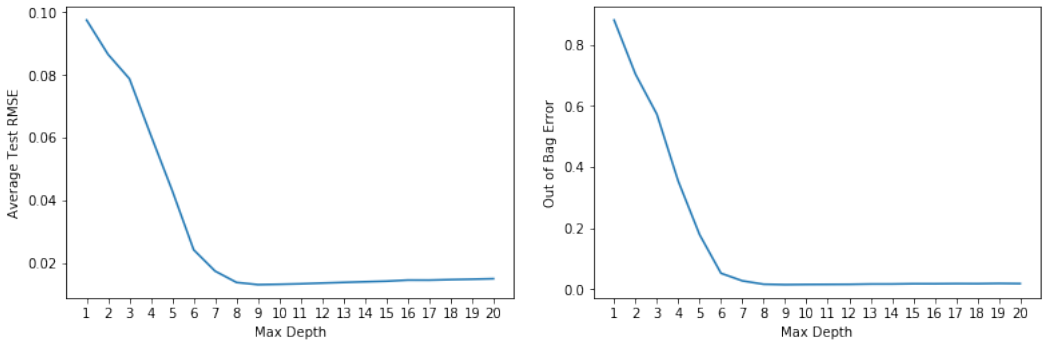


From the above plots, we observe that 4 and 5 features would yield the smallest average test RMSE. For Out of Bag errors, using 3, 4, and 5 features yield almost equally good results. Furthermore, for average test RMSE, using 5 features would give a less fluctuated average test RMSE compared to using 3 features. We also observe that, for both average test RMSE and Out of Bag errors, the values decrease to steady state



errors when the number of trees takes values greater than or equal to 20.

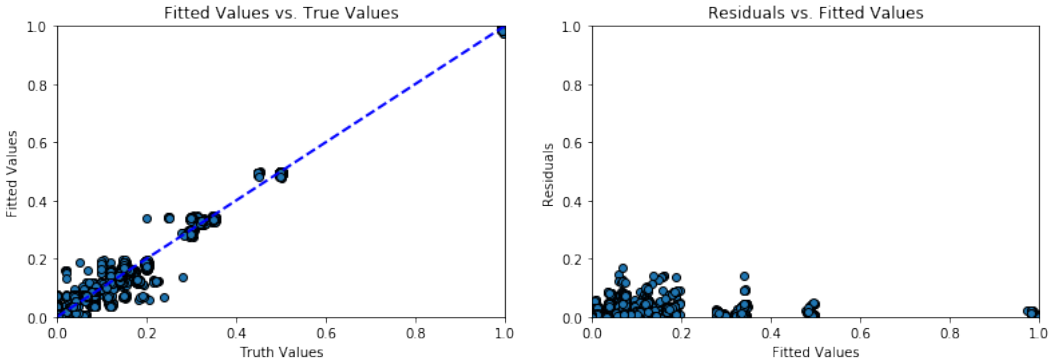
We also experiment on the maximum depth of each tree, for values from 1 to 20, while keeping the number of trees fixed at 25 and number of features fixed at 5. We plot the average test RMSE and Out of Bag error vs. maximum depth below.



From the plots, we observe that the average test RMSE and Out of Bag error decrease to their steady state values for max depths greater than or equal to 8. Therefore, the parameters that achieves the best performance is tree number = 25, number of features = 5, max depth = 8.

Using the above parameters, we yield the following scatter plots :

From the plots, we observe that the predicted value using the random forest regression model predicts better than the basic linear regression model, because the fitted values vs. true values lie rougly at the line y = x. We now compute the average test RMSE and Out of Bag error for this setting and report the results below.



Training RMSE	Test RMSE
0.01297	0.01493
0.01240	0.01415
0.01347	0.01204
0.01248	0.01512
0.01277	0.01168
0.01228	0.01518
0.01304	0.01282
0.01270	0.01381
0.01308	0.01205
0.01274	0.01407

Average training RMSE : 0.0126309
Average test RMSE : 0.0136860
Out of bag error : 0.0158824

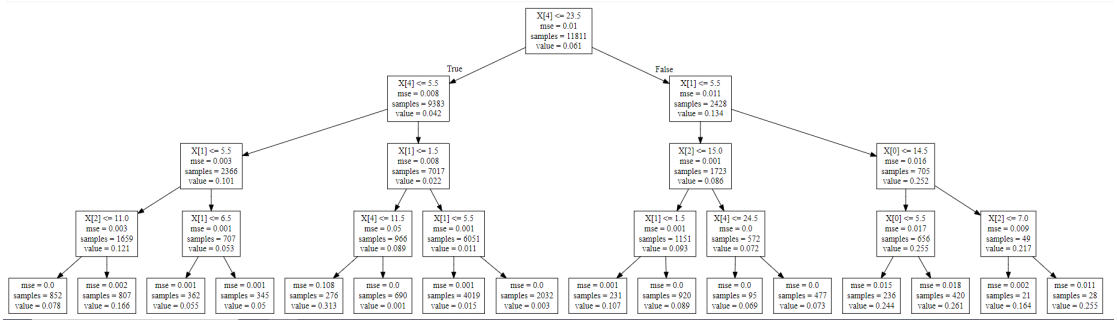
Compared to the average test RMSE and Out of Bag error we obtained from the initial setting (using number of trees = 20, depth of each tree = 4, and maximum number of features = 5), we see that the average test RMSE is improved by $\frac{(0.0605-0.0136)}{0.0605} = 77.52\%$, and the Out of Bag error is improved by $\frac{(0.3407-0.0177)}{0.3407} = 94.80\%$.

Next, we obtain the order of feature importance among the five features, namely, day of the week, hour of the day, work-flow-ID, file name, and week number. We obtain the following results :

```
Feature 2 Importance = 0.39909345209127733
Feature 4 Importance = 0.2532411630695922
Feature 1 Importance = 0.19499279834310884
Feature 3 Importance = 0.15154788044963444
Feature 0 Importance = 0.001124706046387145
```

Based on the importance ranks, we see that the most important features are hour of the day, file name,

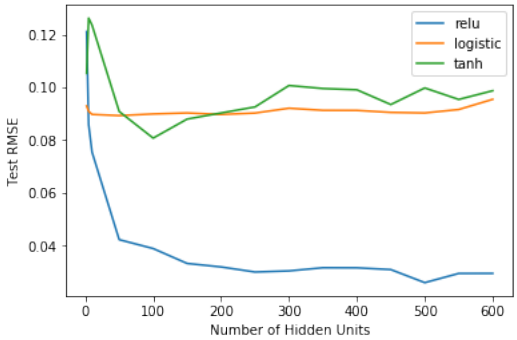
day of the week, work-flow-ID, and week number, with descending order of feature importance. Finally, we visualize our decision tree using graphviz.



The root node in the decision tree corresponds to file name, which is not the most important feature we found in the previous part. This happens because the decision tree we realized in this part is an instance of many trees that are obtained using the random forest algorithm, in which the algorithm uses the many trees and votes on the most important feature based on these results.

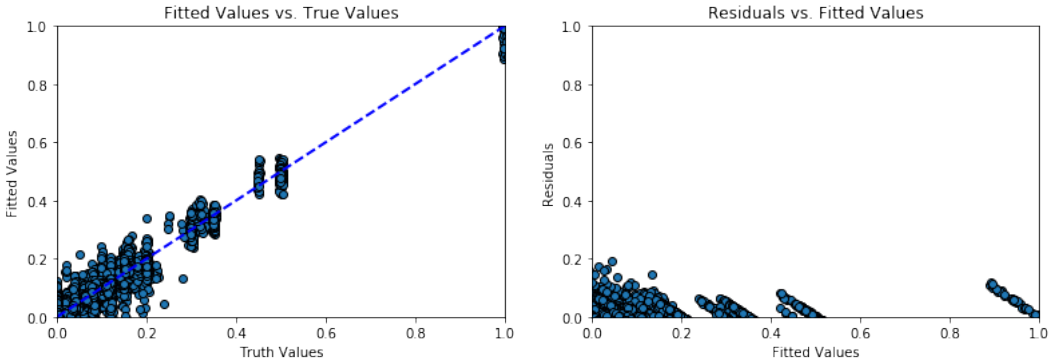
Question 2 (c)

For this part, we fit a neural network regression model with one hidden layer to the dataset. We sweep the number of hidden units from values [2, 5, 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600], and plot the test RMSE vs. number of hidden units used for the hidden layer in our neural network, using the activation functions ReLU, logistic, and tanh.



We observe from the plot that using 500 hidden units and the ReLU activation function, the regression model yields the lowest test RMSE.

Now, we use this setting to fit the dataset and plot fitted values vs. true values and residuals vs. fitted values as scatter plots to visualize how well the model fits the dataset.



Training RMSE	Test RMSE
0.01496	0.03016
0.01363	0.02098
0.01579	0.02752
0.01371	0.04117
0.01562	0.02407
0.01405	0.03481
0.01502	0.03077
0.01411	0.02742
0.01472	0.03099
0.01546	0.02327

Average training RMSE : 0.0147298
Average test RMSE : 0.0296554

From the plots, we observe that although the points from the first plot center roughly around the line y=x, there are many points that do not lie on the line. From the second plot, the residuals or errors of our predicted values range from 0 to a little less than 1. We conclude that the neural network regression model works better than the basic linear regression model, but worse than the random forest regression model, as ideally we would want the points from the first plot to lie roughly at the line y = x, and the points (residuals) from the second plot to be as close to 0 as possible.

The table below shows the performance summary :

	Average Training RMSE	**Average Test RMSE**
Linear Regression	0.1036	0.1037
Random Forest Regression (number of trees = 25, number of features = 4, max depth = 8)	0.0128	0.0136
Neural Network Regression (number of hidden units = 500, activation function = ReLU)	0.0147	0.0297

Question 2 (d)

i - Linear Regression

We use a linear regression model to predict the backup size, but this time, we do it on each of the workflows separately.

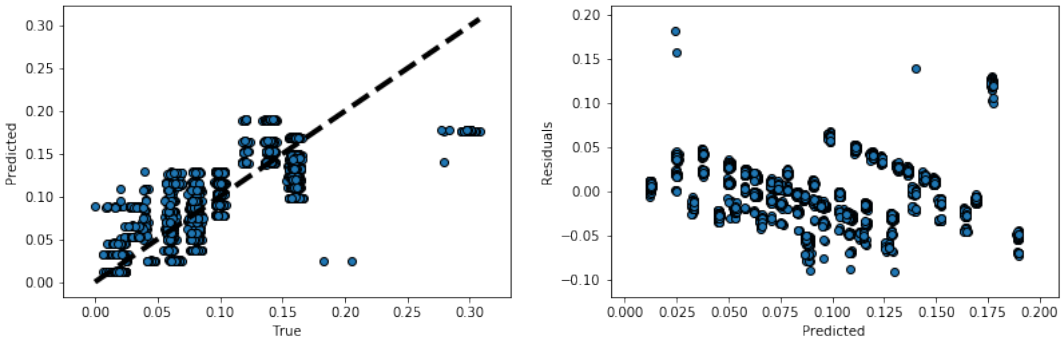


FIGURE 3 – Workflow 0

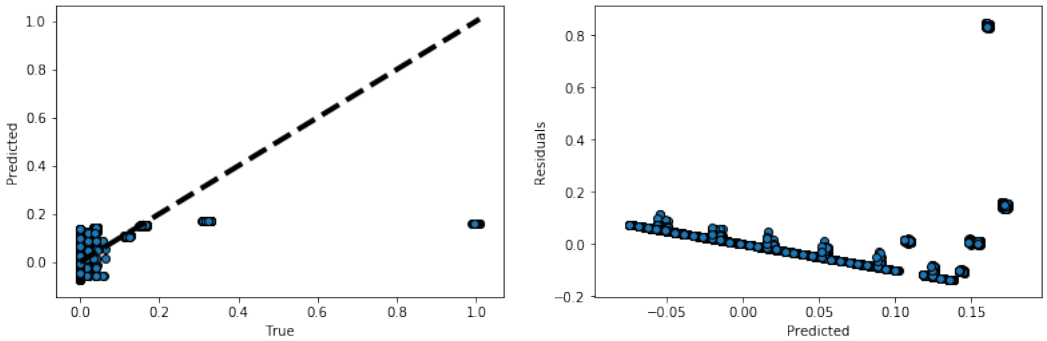


FIGURE 4 – Workflow 1

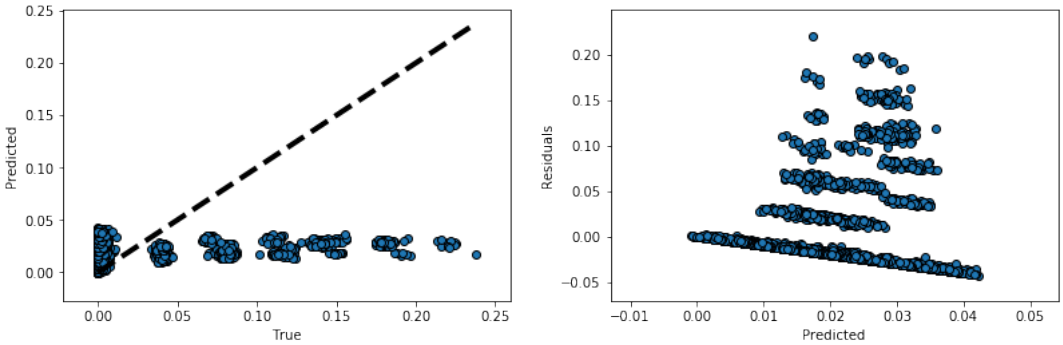


FIGURE 5 – Workflow 2

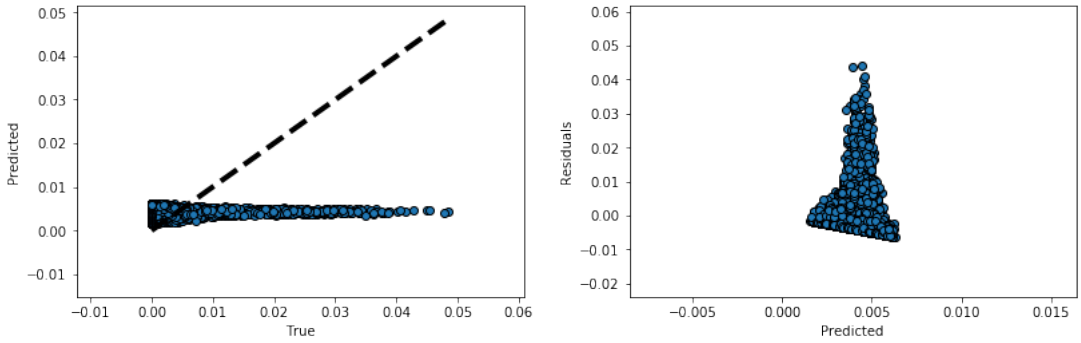


FIGURE 6 – Workflow 3

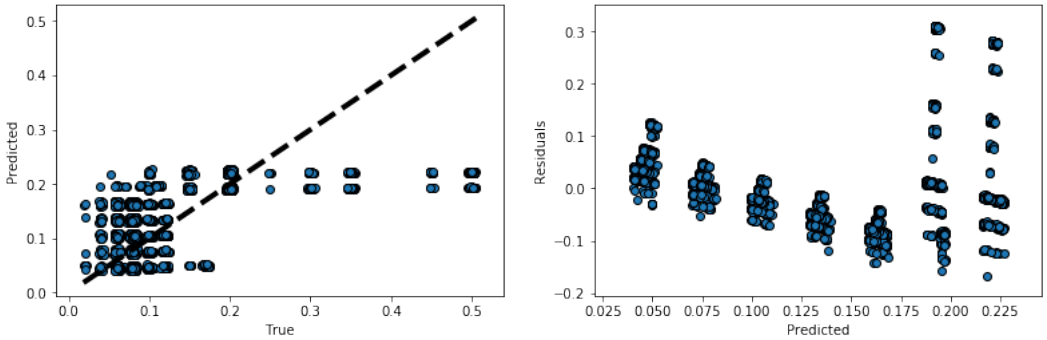


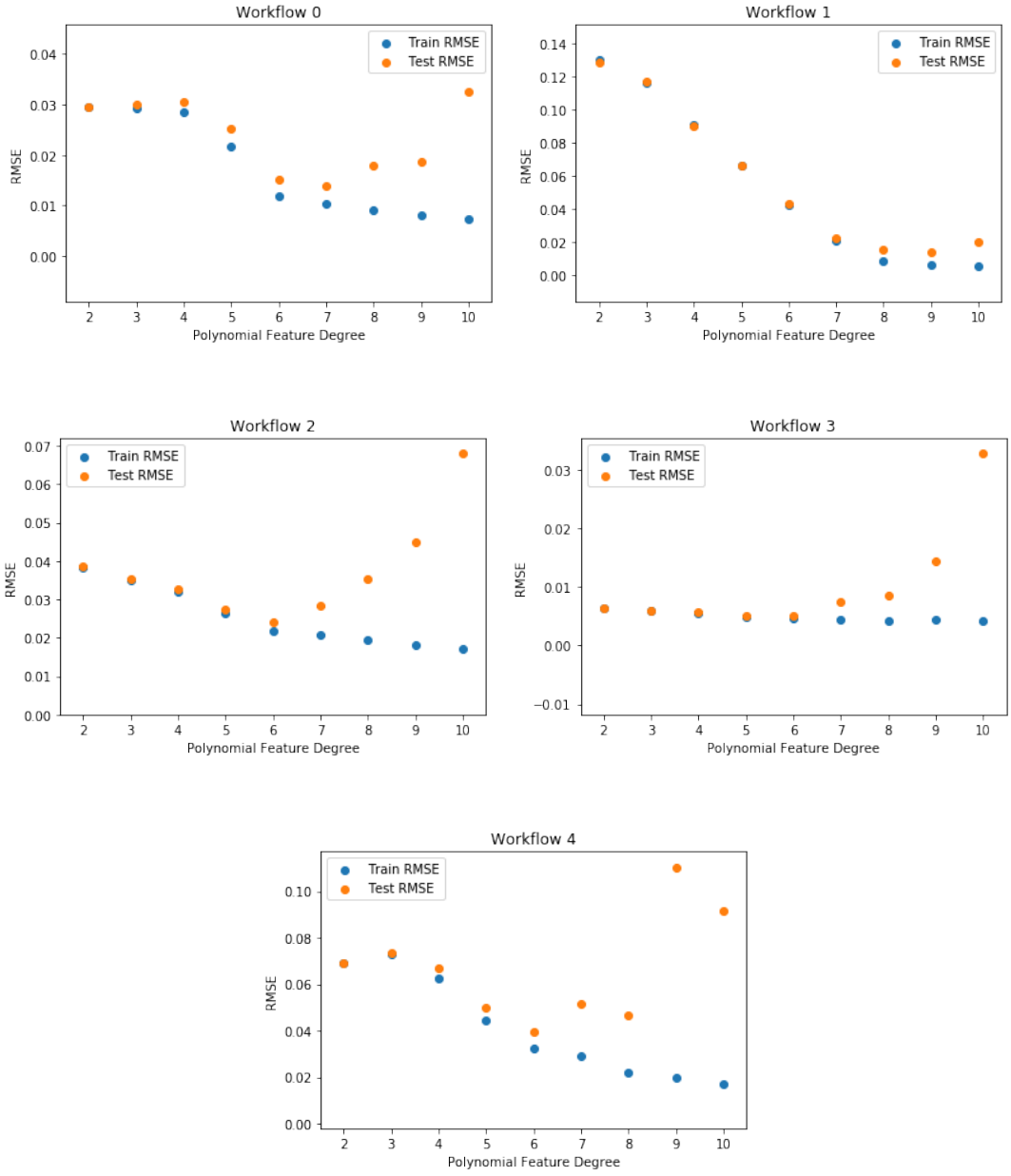
FIGURE 7 – Workflow 4

	Average Train RMSE	Average Test RMSE
Workflow 0	0.03583	0.03586
Workflow 1	0.14874	0.14709
Workflow 2	0.04290	0.04257
Workflow 3	0.00724	0.00719
Workflow 4	0.08591	0.08534

Except for workflow 1, overall we seem to get better results than in point 2(a). Using a linear regression model on each workflow separetely seems to yield lower RMSE.

ii - Polynomial Function

A polynomial function can have more degrees of freedom, which can lead to a better fit of the data. We plot the average train and test RMSE of the trained model against the degree of the polynomial for each workflow.



We notice that overall after the threshold degree of 7, the generalization error of the model gets worse. This is probably due to overfitting as we keep increase the degree to force the polynomial model to fit the data.

Cross validation allows us to detect overfitting when there is a large difference between the RMSE of the train and test dataset. It thus allows us to improve the model without overfitting.

For the optimal degree of 7, we plot on figure 8 the fitted values against true values, as well as the residuals versus fitted values. We get the following error values :

Average training RMSE : 0.0405979
Average test RMSE : 0.0411963

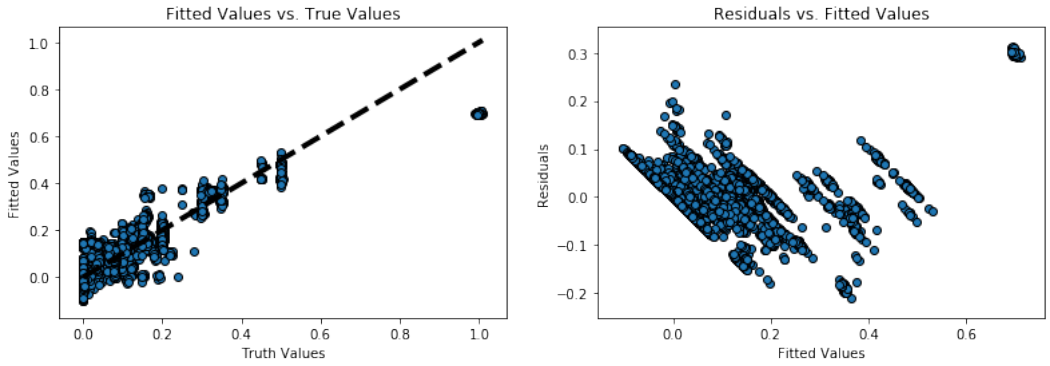


FIGURE 8 – Scatter plots

Question 2 (e)

Finally, we use the k-nearest neighbor regression. To find the best parameter, we plot the average train and test RMSE against the number of neighbors k. The graph is represented on figure 9.

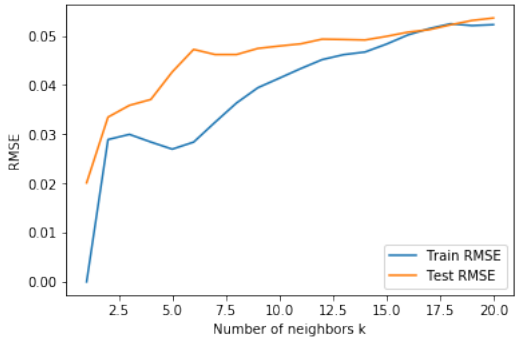


FIGURE 9 – RMSE vs number of neighbors

The best RMSE is obtained when k = 2. We get the following error values :

Average training RMSE : 0.028907
Average test RMSE : 0.033467

The plots of the fitted values against true values, as well as the residuals versus fitted values are represented on figure 10.

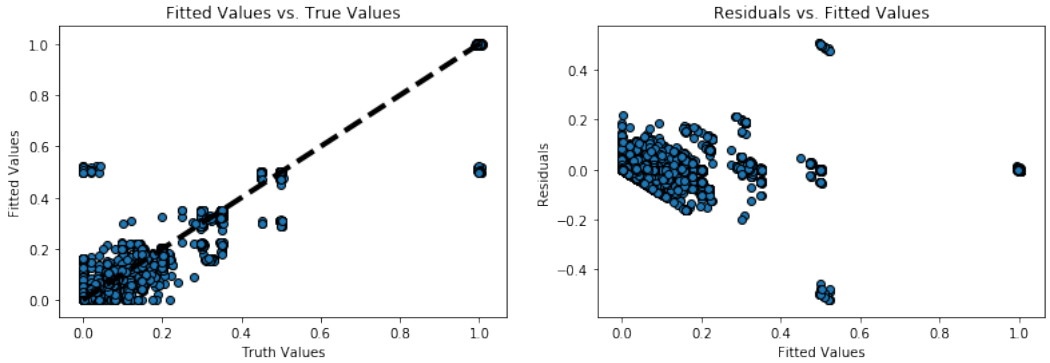


FIGURE 10 – Scatter plots

Question 3

The best overall performance is obtained with the Random Forest Regression model with a max_depth of 8. It has the lowest RMSE value of all : 0.0128

1. Linear Regression : produced the worst results (average RMSE of 0.1036). However, when we used a linear regression model on each workflow separetely, we got much better results for all workflows except workflow 1 for which the average RMSE didn't improve. This regression is the simplest. It has the advantage to be the fastest.

2. Random Forest Regression Model : produced the best results (average RMSE of 0.0128). This method is best for handling categorical features. On top of that, it can evaluate the importance of each feature and calculate the error rate of the algorithm using out of bag data.

3. Neural Network Regression Model : good at handling sparse features as it was mainly dealing with the sparse matrix with all features one-hot encoded. This model works better than the basic linear regression model, but worse than the random forest regression model.

4. Polynomial Function Regression Model : better than linear regression to predict the backup size for the workflows. It works well with numerical features. We need to make sure to select the right degree as threshold to avoid overfitting.

5. K-Nearest Neighbor Regression Model : it provides decent results once we select the right k, even comparable to the best results obtained with polynomial regression. It is not good to handle sparse features.

Dataset 2 : Boston Housing

Question 1 : Linear Regression

Feature Significance Analysis

Dataset 2 concerns housing values in the suburbs of the greater Boston area. The following table shows the significance of different variables with linear regression model through calculating the t-values and p-values.

Coefficient Name	Coefficient	Std Error	T values	P values
Constant	36.459	5.032	7.245	1.623e-12
CRIM	-0.108	0.032	-3.333	9.222e-4
ZN	0.046	0.014	3.429	6.544e-4
INDUS	0.021	0.061	0.339	7.347e-01
CHAS	2.687	0.850	3.162	1.658e-03
NOX	-17.767	3.767	-4.717	3.101e-06
RM	3.810	0.412	9.245	0
AGE	0.001	0.013	0.053	9.576e-01
DIS	-1.476	0.197	-7.503	2.836e-13
RAD	0.306	0.065	4.678	3.721e-06
TAX	-0.012	0.004	-3.326	9.439e-04
PTRATIO	-0.953	0.129	-7.385	6.306e-13
B	0.009	0.003	3.516	4.779e-04
LSTAT	-0.525	0.0500	-10.493	0

Through the tables, we can find that The most significant features are 'RM', 'LSTAT', 'DIS', 'PTRATIO' and the least significant features : 'INDUS', 'AGE'.

10-Fold cross validation

In this question, we will calculate the averaged root mean squared error (RMSE) for trainset and testset through 10-fold cross validation. The following table shows the RMSE for each split cases.

Training RMSE	Test RMSE
4.834	3.047
4.784	3.762
4.818	3.751
4.558	5.934
4.619	5.647
4.729	4.454
4.830	3.154
3.458	12.976
4.646	5.773
4.816	3.311

The averaged test RMSE is 5.89 and the averaged train RMSE is 4.63. Through the table above, we can see one test RMSE is obviously higher than the others and the test RMSE is much higher than train RMSE. This is because of the over-fitting. We will try to control the over-fitting in the next question.

Whole Dataset

In this question, we will fit the whole dataset and predict with linear regression model. The following two figures show the fitted values against trues values and the residuals against fitted values.

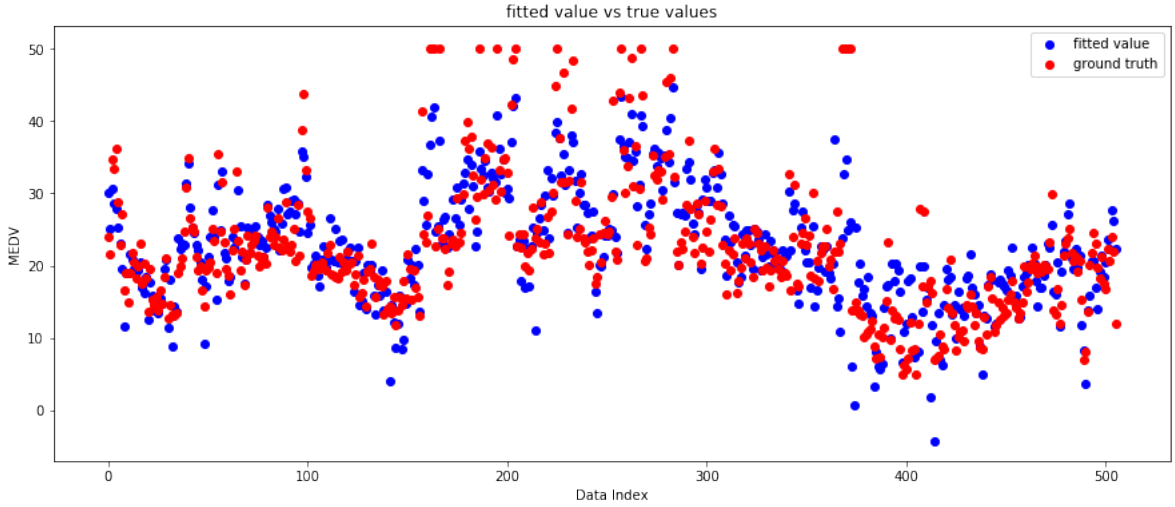


FIGURE 11 – fitted values against true values

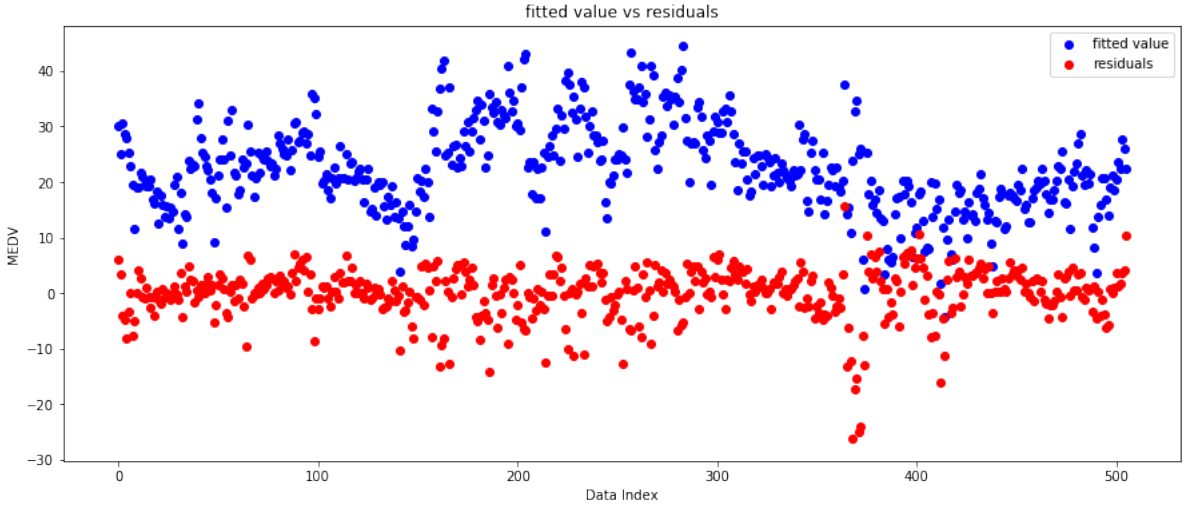


FIGURE 12 – residuals against fitted values

Question 2 : Improvement through Regularization

In question 2, we will try to improve the model via different types of regularizer.

Ridge Regularizer

The ridge regularizer is a combination of linear regression and a regularization given by l2-norm. The equation is shown below :

$$\min_{\beta} ||y - \beta X||_2^2 + \alpha||\beta||_2^2$$

Through testing, the best choice of α is 52.9. The corresponding train RMSE and test RMSE is shown below :

Training RMSE	Test RMSE
4.996	3.202
4.975	3.645
5.005	3.244
4.690	6.600
4.816	5.615
4.909	4.642
4.993	3.383
3.770	10.862
4.789	5.346
4.966	3.690

The average train RMSE is 4.804 and the average test RMSE is 5.498.

Lasso Regularizer

The Lasso regularizer is a combination of linear regression and a regularization given by l1-norm. The equation is shown below :

$$\min_{\beta} \frac{1}{2 * 506}||y - \beta X||_2^2 + \alpha||\beta||_1$$

Through testing, the best choice of α is 0.499. The corresponding train RMSE and test RMSE is shown below :

Training RMSE	Test RMSE
5.069	3.171
5.051	3.733
5.081	3.020
4.787	6.952
4.912	5.932
5.001	5.149
5.068	3.536
3.690	11.543
4.838	5.280
5.031	3.622

The average train RMSE is 4.869 and the average test RMSE is 5.744.

Elastic Net Regularizer

The Elastic Net regularizer is a combination of Ridge regularizer and Lasso regularizer. The equation is shown below :

$$\min_{\beta} \frac{1}{2*506}||y-\beta X||_2^2 + \alpha * L1Ratio * ||\beta||_1 + 0.5 * \alpha * (1 - L1Ratio) * ||\beta||_2^2$$

Through testing, the best choice of α is 0.36 and l1 ratio is 0.02. The corresponding train RMSE and test RMSE is shown below :

Training RMSE	Test RMSE
5.104	3.284
5.074	3.906
5.109	3.368
4.773	6.965
4.899	5.987
4.999	5.092
5.097	3.624
4.127	9.686
5.075	5.043
5.031	3.770

The average train RMSE is 4.927 and the average test RMSE is 5.425.

Coefficients Comparison

In previous questions, we have examined Ridge, Lasso and Elastic Net regularizer. All of the three regularizers improve the test rmse and mitigate the over-fitting problem. Besides, the Elastic Net regularizer return the best proformance and the Lasso return the worst. Now, we will compare the coefficients of these regularizer with linear regression and do further analysis.

Coefficient Name	Linear Regression	Ridge	Lasso	Elastic Net
Constant	36.459	33.262	32.50419	39.804
CRIM	-0.108	-0.102	-0.083	-0.101
ZN	0.046	0.0532	0.050	0.055
INDUS	0.021	-0.052	-0.005	-0.052
CHAS	2.687	0.901	0	0.396
NOX	-17.767	-0.441	0	-0.122
RM	3.810	2.972	2.501	1.770
AGE	0.001	-0.003	0.004	0.008
DIS	-1.476	-1.192	-0.937	-1.073

RAD	0.306	0.304	0.277	0.326
TAX	-0.012	-0.015	-0.015	-0.016
PTRATIO	-0.953	-0.819	-0.759	-0.830
B	0.009	0.010	0.009	0.009
LSTAT	-0.525	-0.628	-0.656	-0.700

Lasso use the l2 norm as its penalty function. It makes the coefficients become sparse (The coefficients is close to zero). However, most of the features in dataset 2 are significant which shouldn't be neglected. So, Lasso return the worst performance among the three regularizers. Compared to Lasso, Ridge uses l1 norm as its penalty function. It will add smaller penalty than Lasso to the results, which is more suitable for this dataset. Elastic Net uses both l1 norm and l2 norm as penalty functions. So, we can adjust the parameters to find the best combination between l1 norm and l2 norm then return the best result.

Dataset 3 : Car Insurance

Question 1 : Feature Pre-processing

In this question, we use 3 different methods on feature pre-processing. Then, we fit a linear regression model and calculate the average train and test RMSE by using 10 fold cross validation respectively. Finally, we trained a linear model using the whole dataset and plot figures about target values and residuals.

a. Feature Encoding

The first feature pre-processing method is one-hot-encoding, which encoded the categorical features ft4, ft5 and ft6 into numerical features.

The average train RMSE is 6039.342 and the average test RMSE is 6063.644.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset.

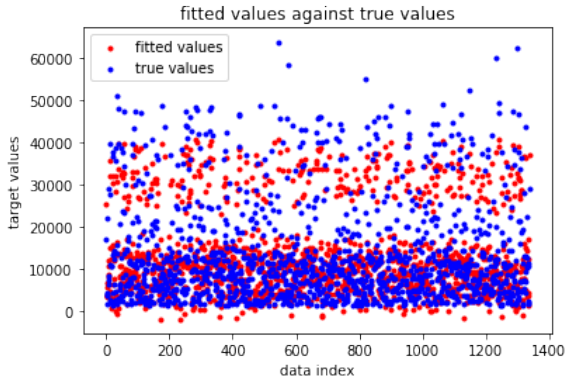


FIGURE 13 – fitted values against true values

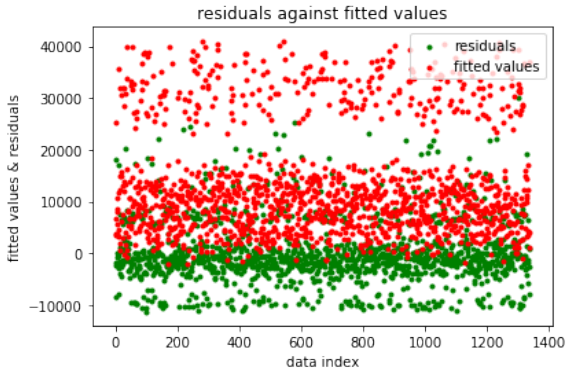


FIGURE 14 – residuals against fitted values

b. Standardization

The second feature pre-processing method is standardizing the numerical features ft1, ft2, ft3 and keep the pre-process we did in part a.

The average train RMSE is 6039.397 and the average test RMSE is : 6061.418.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset.

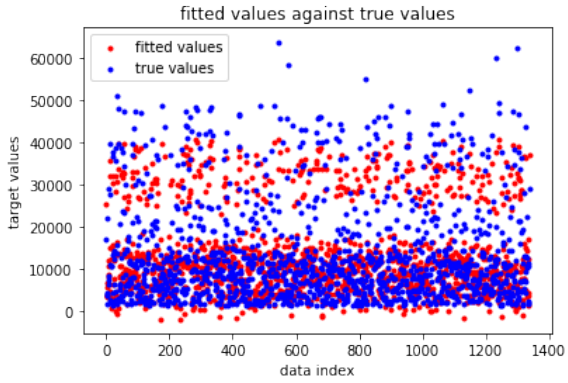


FIGURE 15 – fitted values against true values

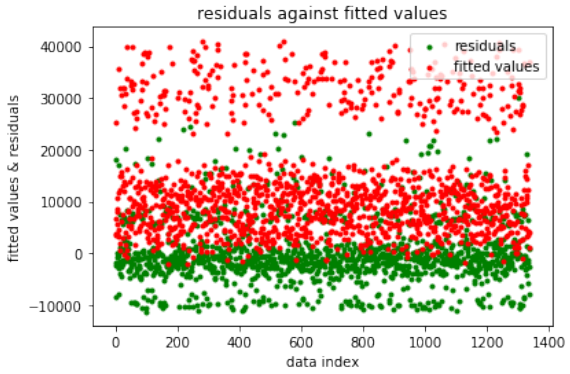


FIGURE 16 – residuals against fitted values

c. Redefine Feature 'ft1'

The third feature preprocessing method is that divide ft1 into 3 ranges : set the new values to 1 for original values below 30, 2 for values between 30 and 50 and 3 for values above 50. Then, keep the pre-processes we did in part a and part b.

The average train RMSE is 6199.414 and the average test RMSE is 6221.844.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset.

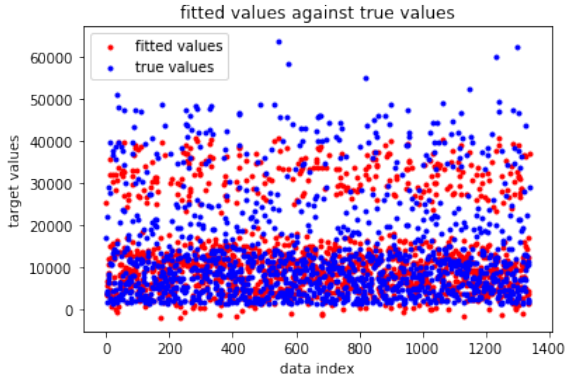


FIGURE 17 – fitted values against true values

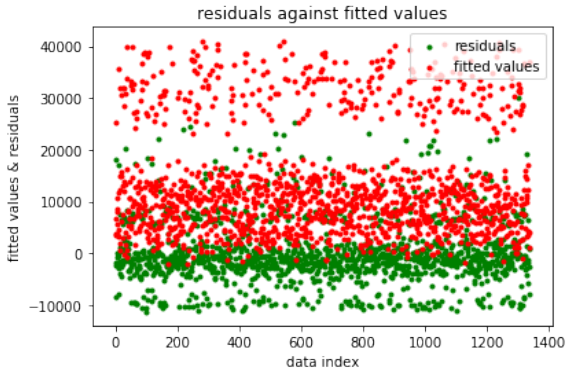


FIGURE 18 – residuals against fitted values

Question 2 : Correlation Exploration

part a

In part a, we convert each categorical feature into a one dimensional numerical value. Then we use f regression and mutual information regression measure to do correlation exploration and select two most important variables respectively.

The following is the result we get from the f regression :

	ft1	ft2	ft3	ft4	ft5	ft6
F score	1868.35	1726.07	524.29	586.80	284.25	610.78
P score	3.95e-256	6.10e-243	3.64e-98	8.98e-108	5.61e-58	2.25e-111

According to f regression, ft1 and ft2 are the two most important features.

The following is the result we get from the mutual info regression :

	ft1	ft2	ft3	ft4	ft5	ft6
score	1.501	0.074	0.161	0.177	0.369	0.076

According to mutual info regression, ft1 and ft5 are the two most important features.

part b-c

The following twos plots show charges (y axis) vs ft2 (x axis) and color points based on ft5 and charges (y axis) vs ft1 (x axis) and color points based on ft5 respectively :

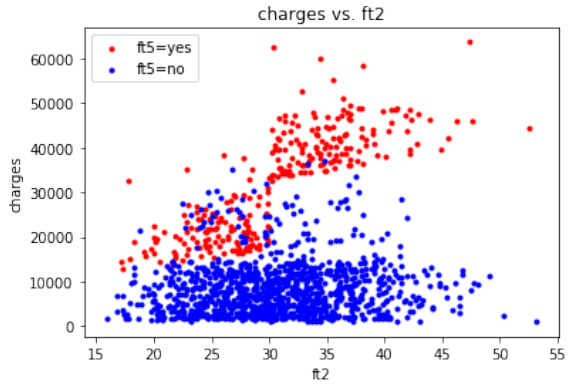


FIGURE 19 – charges vs ft2

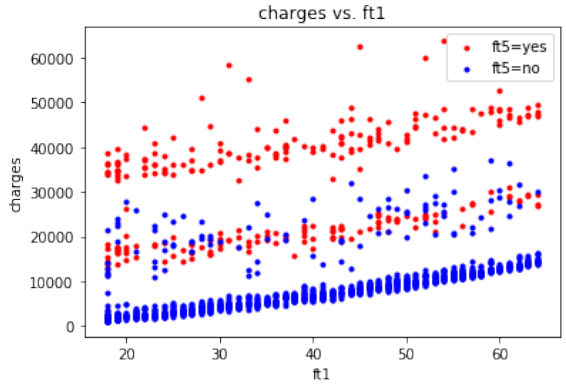


FIGURE 20 – charges vs ft1

Question3 : Modify the target variable

Because charges (y) spans a wide range, we consider fitting log(y) instead of fitting the original value and see whether the result improves.

part a

In this part, we still calculate the average train RMSE and test RMSE but do logarithm transformation for the target variables. Note : instead of calculating the difference between predicted value ($log(y)_{predict}$) and transformed target values (log y), we calculate the difference between $exp(log(y)_{predict})$ and y to set up a fair comparison.

The average train RMSE is 8358.941 and the average test RMSE for testing data is 8373.796. The test RMSE is larger than what we did in part1(a), which means that logarithm doesn't improve the result.

part b

In this part, we repeat what we did in part 2 to do correlation exploration and select two most important variables respectively with the logarithm target variables.

The following is the result we get from the f regression :

	ft1	ft2	ft3	ft4	ft5	ft6
F score	13711.752	30006.417	1156.771	1340.699	3696.606	2156.007

According to f regression, ft1 and ft2 are the two most important features.

The following is the result we get from the mutual info regression :

	ft1	ft2	ft3	ft4	ft5	ft6
score	1.499	0.068	0.161	0.176	0.369	0.0778

According to mutual info regression, ft1 and ft5 are the two most important features.

Based on the results, the important features remain unchanged, which is expected.

The following twos plots show charges (y axis) vs ft2 (x axis) and color points based on ft5 and charges (y axis) vs ft1 (x axis) and color points based on ft5 respectively :

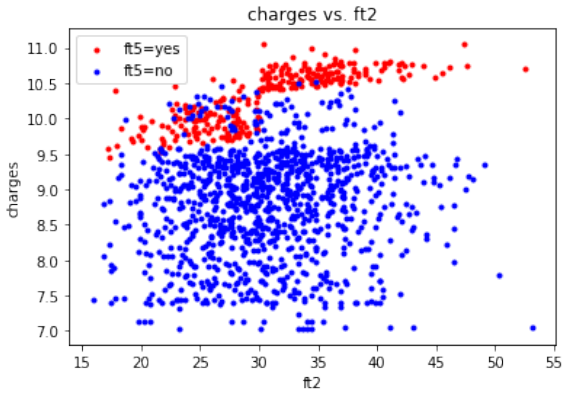


FIGURE 21 – charges vs ft2

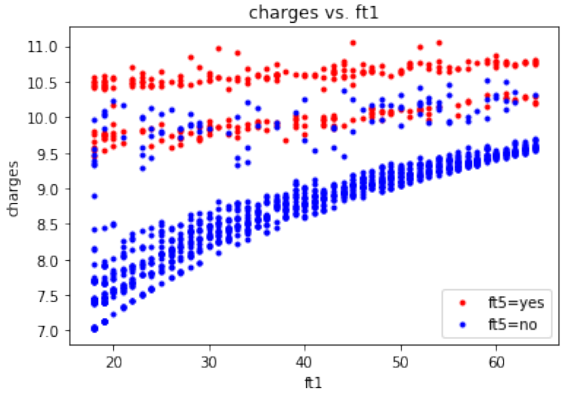


FIGURE 22 – charges vs ft1

Question4 : Further Improvement

Part a : Polymonial Feature

In this question, We use the same feature pre-processing method as that in question 1 part a. To improve the performance of the model, we try polynomial features to do a better feature encoding. We use the polynomial features to fit a linear regression model and calculate the average train RMSE and average test RMSE through 10-fold cross validation when we use different polynomial degrees.

The following table shows the average train RMSE and average test RMSE from degree 1 to 6.

Degree	Train RMSE	Test RMSE
1	6039.342	6063.644
2	5765.784	5875.214
3	6515.548	7156.467
4	4280.461	5747.640
5	3724.351	10208.908
6	3039.633	9161779.981

From the table, when the degree become higher, we can get a smaller train RMSE but the higner test RMSE. This is caused by overfiiting. When degree=4, we have the smallest average test RMSE, which means this model has the best generalization ability in this degree.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset when degree = 4.

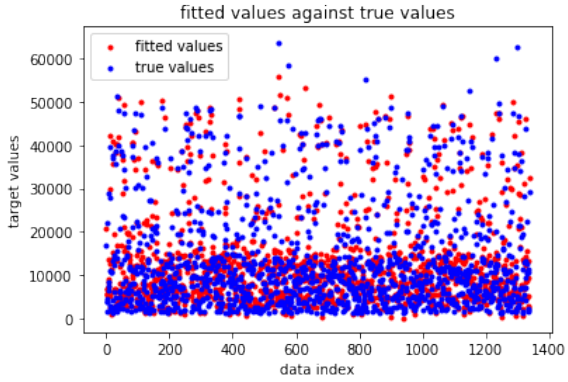


FIGURE 23 – fitted values against true values

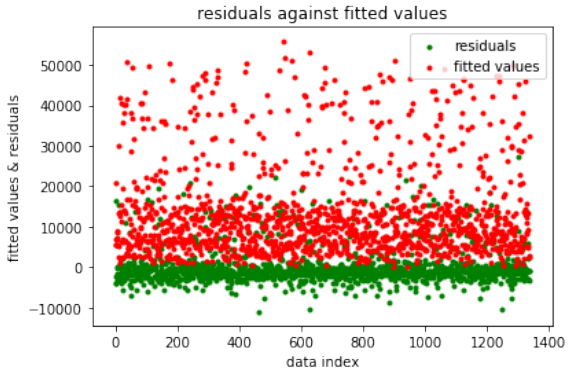


FIGURE 24 – residuals against fitted values

Part b : different models

In this question, We use the same feature pre-processing method as that in part 1a. However, we will use different models to make improvements.

Random Forest

In this part, we will train and fit our data through random forest model. We choose the number of estimators and maximum depth to be the hyper-parameters and use 10-fold cross validation to find the best

hyper-parameters.

The following table shows the average train RMSE and average test RMSE with different estimators and maximum depths in random forest model :

Estimators	Max Depth	Train RMSE	Test RMSE
300	3	4465.098	4573.403
300	5	4011.353	4491.870
300	7	3239.919	4591.257
600	3	4464.066	4571.107
600	5	4012.711	4494.903
600	7	3230.853	4579.144

The best test RMSE is 4491.870 when estimators is 300 and max depth is 5.

By using random forest model we do get a smaller train and test RMSE, which means this model performs better when predicting target values. From the result we can conclude that at first the model tend to have a better performance when we use more estimators but later on the number of estimators doesn't have much effect on the performance of the model. With a larger maximum depth the model can always get a smaller average training RMSE, but the average testing RMSE first decreases and then increased, which means the model begins to overfit data. Based on our result, max depth = 5 will lead to the best generalization ability, and it performs better than the model in part 1a.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset through random forest model.

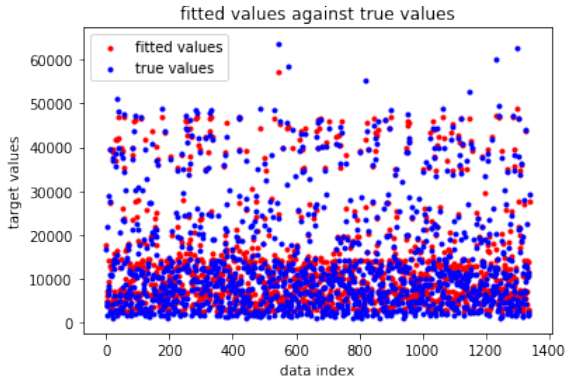


FIGURE 25 – fitted values against true values

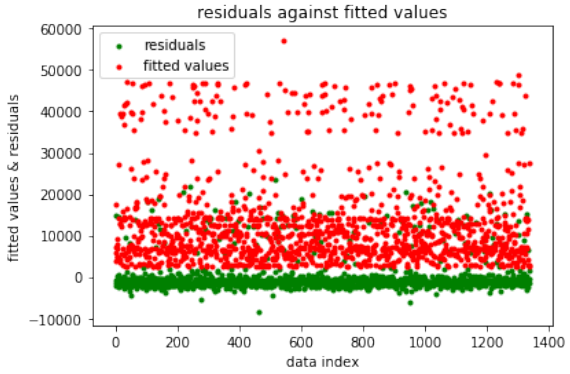


FIGURE 26 – residuals against fitted values

Neural Network

In this part, we will train and fit our data through neural network model with learning rate=0.015, batch size=800, maximum iteration number=3000. Then we calculate the RMSE through 10-cross validation.

Based on our result, the average train RMSE is 5570.012 and the average test RMSE is 5623.807.

We see a decrease in RMSE value compared with the RMSE value in part 1a, which means this neural network model has a better performance than the model we trained in part 1a.

The following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset through neural network model.

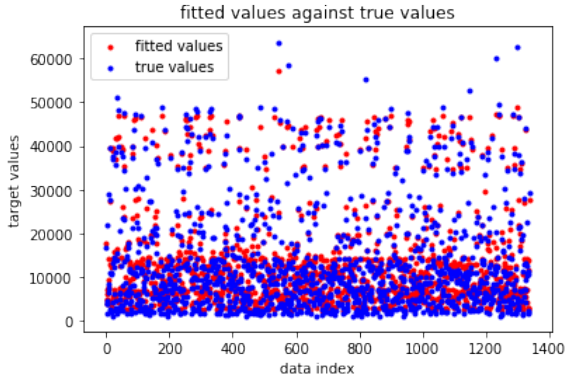


FIGURE 27 – fitted values against true values

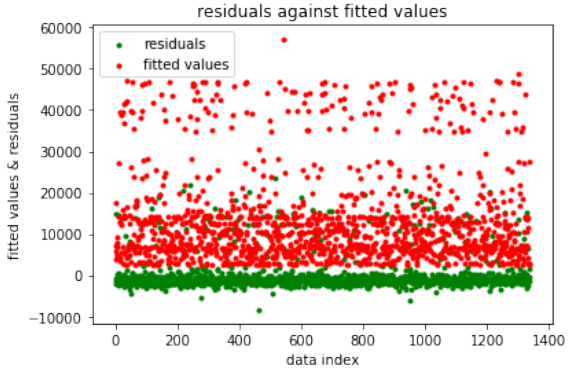


FIGURE 28 – residuals against fitted values

Gradient Boosting Tree

In this part, we will train and fit our data through gradient boosting tree model. We choose the number of estimators and maximum depth to be the hyper-parameters and use 10-fold cross validation to find the best hyper-parameters.

The following table shows the average train RMSE and average test RMSE with different estimators and maximum depths in gradient boosting tree model :

Estimators	Max Depth	Train RMSE	Test RMSE
10	2	6267.876	6283.850
10	3	5953.721	6015.673
10	4	5815.187	5916.270
50	2	4243.467	4499.847
50	3	4073.720	4447.523
50	4	3779.826	4513.407
100	2	4173.564	4470.805
100	3	3786.826	4516.511
100	4	3249.899	4638.780

The best test RMSE is 4447.523 when estimators is 50 and max depth is 3.

Based on the result, when max depth¿3, the model begins to overfit data so that we get a larger testing RMSE and a smaller training RMSE. The minimum test RMSE is about 4448, which is smaller than the result in part 1a. This indicates that the gradient boosting tree model performs better than the linear regression model we trained in part 1a.

he following two figures show the fitted values against trues values and the residuals against fitted values for the whole dataset through gradient boosting tree model.

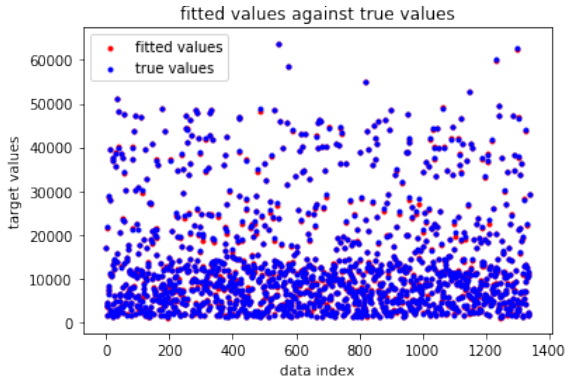


FIGURE 29 – fitted values against true values

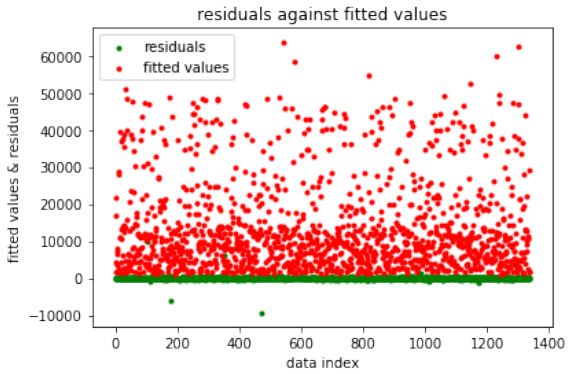


FIGURE 30 – residuals against fitted values

Conclusion

For dataset 1, we compared Linear Regression, Random Forest Regression, Neural Network Regression, Polynomial Function Regression and KNN Regression models. The best overall performance was obtained with the Random Forest Regression model (with a max_depth of 8). This model is best for handling categorical features.

For dataset 2, we used 4 models to fit and train our data, which are Linear Regression, Ridge regularizer, Lasso regularizer and Elastic Net regularizer. Both Ridge and Lasso regularizer add penalty functions (l2 norm for Ridge and l1 norm for Lasso) to the original linear regression. The difference is that Lasso will add more penalty than Ridge, which means Lasso is more suitable for sparse data. Elastic Net regularizer add both L1 norm and L2 norm as its penalty function. So, it will be flexible and suitable for most of the datas.

For dataset 3, we use three feature pre-processing methods, which are one-hot-encoding, standardization and specific feature specification. The first method will improve results when features are categorical. The second method standardize the numerical features. It improves very little on the dataset 3. Standardization is suitable for large numerical values. The third method we simplified ft1 by dividing ft1 into 3 ranges. Because ft1 is an important feature, simplify this will cause information loss. This method only be suitable for less important features. Also, among the three models we tested above, the gradient boosting tree model return the best results.