EE 219 Project 4

Yifan Shu, Chengshun Zhang, Xuan Yang

Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, we explore basic regression models on a given dataset, along with basic techniques to handle over- fitting; namely cross-validation, and regularization. With cross-validation, we test for overfitting, while with regularization we penalize overly complex models.

Dataset 1

For this dataset, we explored the Network Backup Dataset, which consist of simulated traffic data on backup system. The system monitors the files residing in a destination machine and copies their changes in four hour cycles. The dataset have the following features:

- Week #
- Day of Week
- Backup Start Time - Hour of Day
- Work-Flow-ID
- File Name

And we were using these features to predict the size of backup (GB). The models we used in this part includes linear regression, random forest, neural network, and K nearest neighbors (KNN).

Q1: Load the dataset.

(a) For the first twenty-day period (x-axis unit is day number) plot the backup sizes for all workflows (color coded on the y-axis).

The backup sizes for the first twenty-day period can be shown in figure 1-1.

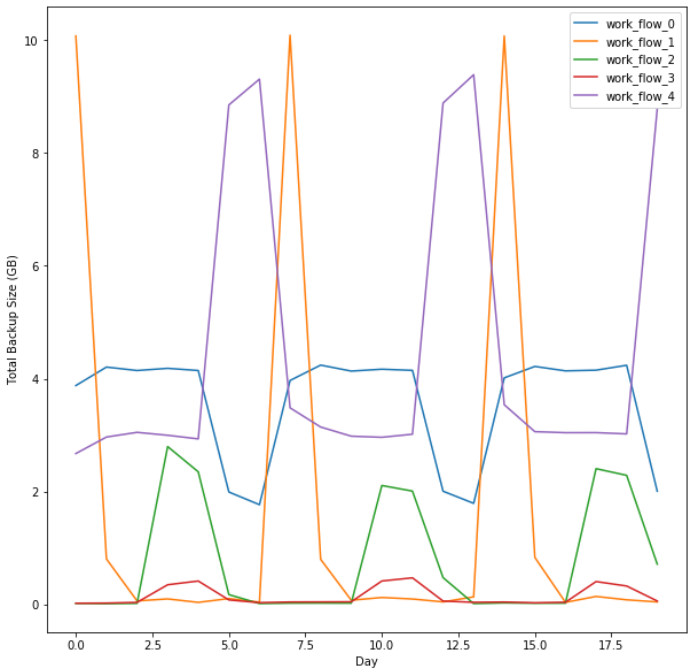

Figure 1-1. Backup Sizes for the First Twenty-Day Period

(b) Do the same plot for the first 105-day period.

The backup sizes for the first twenty-day period can be shown in figure 1-2.

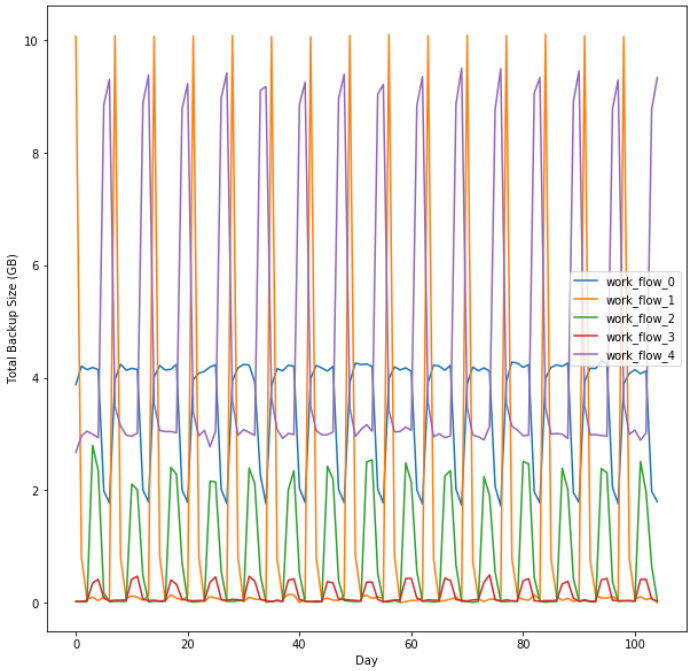

Figure 1-2. Backup Sizes for the First 105-Day Period

(c) Can you identify any repeating patterns?

As shown in the figure, all work flows have periodical pattern and thus make it predictable.

Q2: Examining Different Models

(a) Linear Regression Model:

For this part, we first investigated the linear regression model. We first used the scalar encoding method to transforming the non-numerical values into numeric one. And then directly used the data to fit the model.

The training result and the testing result can be shown in the table 1-1.

	Training RMSE	Test RMSE
Error	0.103584726740266	0.10362189439542643

Table 1-1. The Training and Testing RMSE for the Linear Regression Model

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-3) as well as the the residuals versus fitted values image (which can be shown in figure 1-4) .

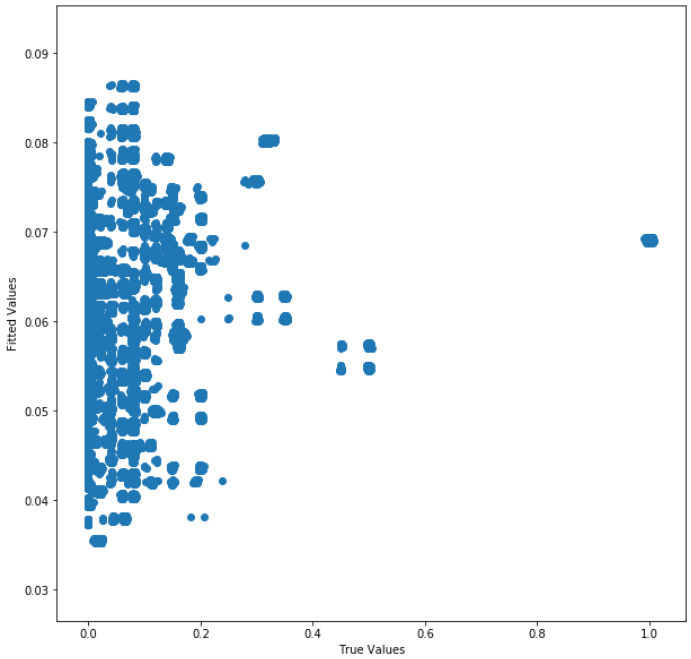

Figure 1-3. Fitted Values Against True Values of Linear Regression Model

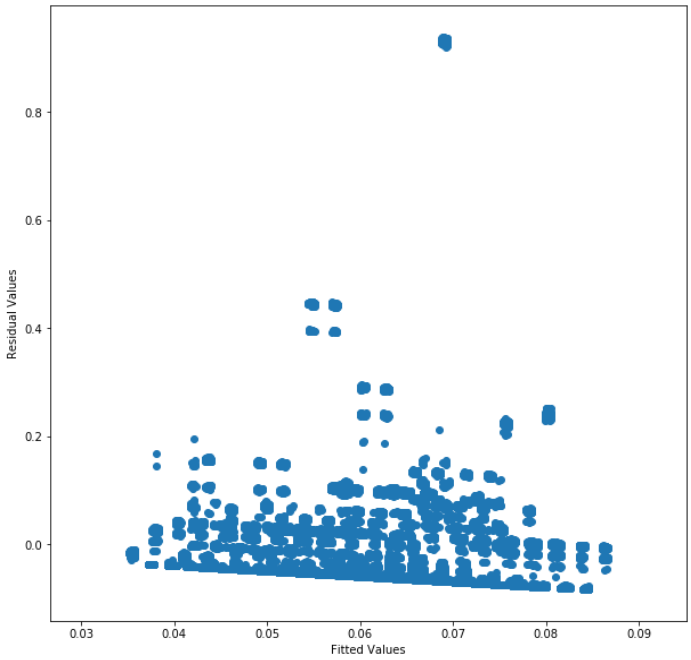

Fig 14. Residuals Against Fitted Values of Linear Regression Model

(b) Random Forest Model:

In this part we were using the random forest model to predict the backup size (GB). We used the scalar encoding method to transforming the non-numerical values into numeric one.

(i)

We first used the random forest with the following hyper-parameters.

- Number of trees: 20
- Depth of each tree: 4
- Bootstrap: True
- Maximum number of features: 5

After using these hyper-parameters, the training RMSE, test RMSE, and Out Of Bag (OOB) error can be shown in table 1-2.

	Training RMSE	Test RMSE	OOB Error
Error	0.06041202096090612	0.060086618271074144	0.3395467700268943

Table 1-2. The Training RMSE, Testing RMSE, and OOB error for the Random Forest Model

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-5) as well as the the residuals versus fitted values image (which can be shown in figure 1-6) .

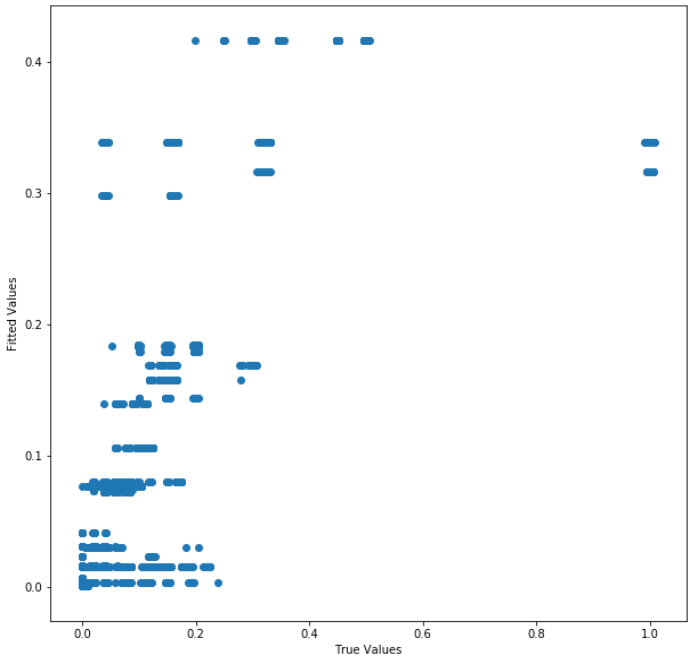

Figure 1-5. Fitted Values Against True Values of Random Forest Model

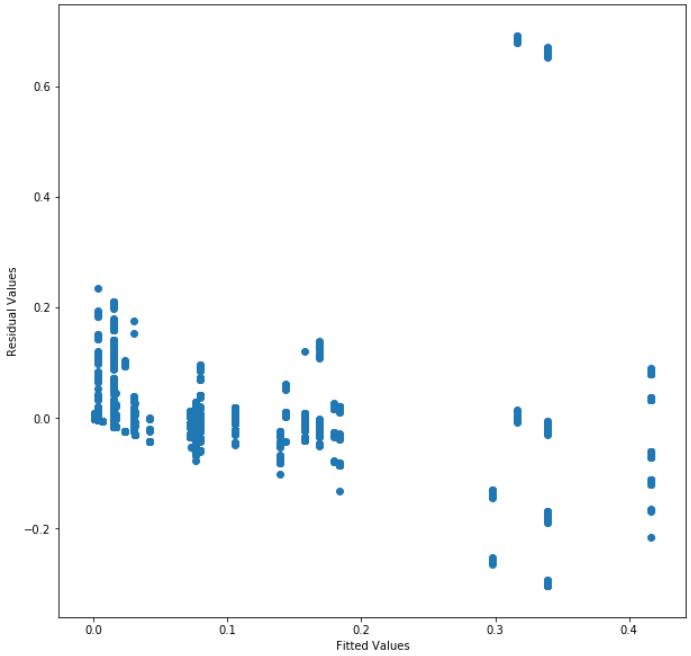

Figure 1-6. Residuals Against Fitted Values of Random Forest Model

(ii)

We then swept over the number of trees from 1 to 200 and maximum number of features from 1 to 5. We made other hyper-parameter unchanged. The average RMSE versus the number of trees/features plot can be shown in figure 1-7 and the average OOB error versus the number of trees/features can be shown in figure 1-8.

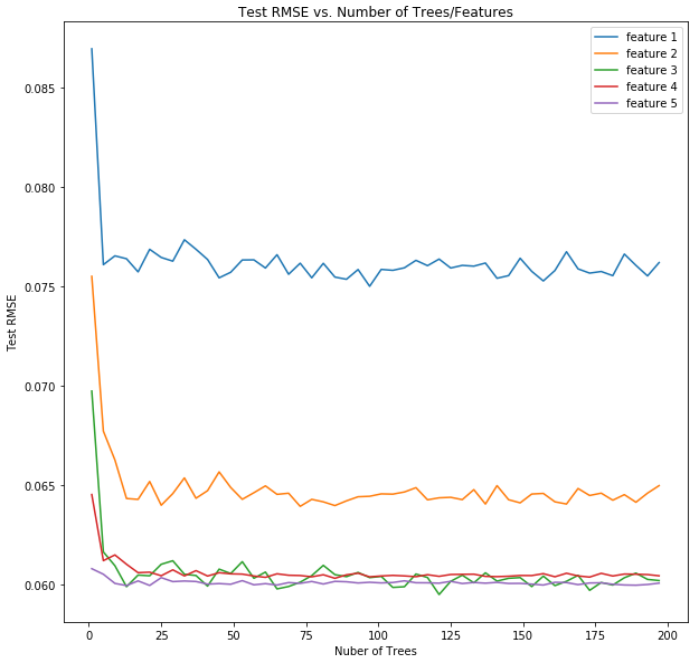

Figure 1-7 The Average RMSE Versus the Number of Trees/Features

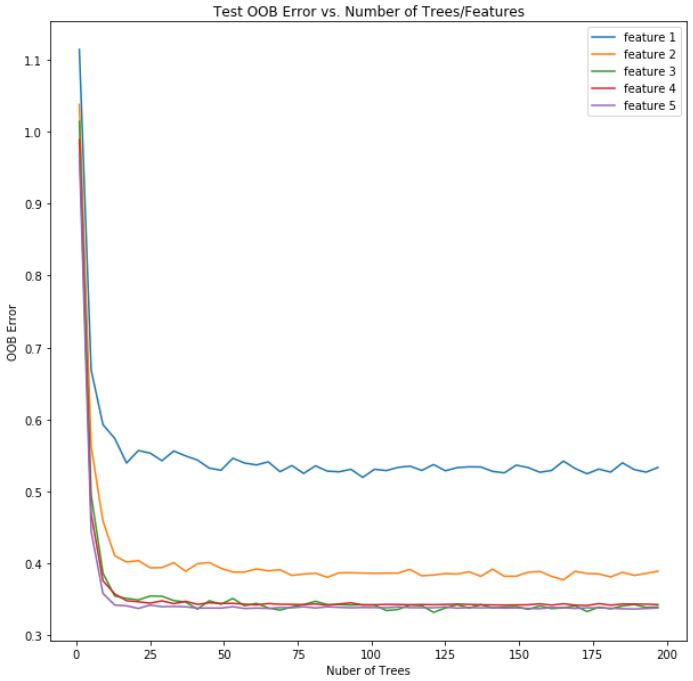

Figure 1-8 The OOB Error Versus the Number of Trees/Features

(iii)

For this part, we swept over the depth from 1 to 200 and maximum number of features from 1 to 5. From Q1-2bii, we picked the number of trees as 16 since it shows the best performance in the previous part. The

average RMSE versus the depth plot can be shown in figure 1-9 and the average OOB error versus the depth can be shown in figure 1-10.

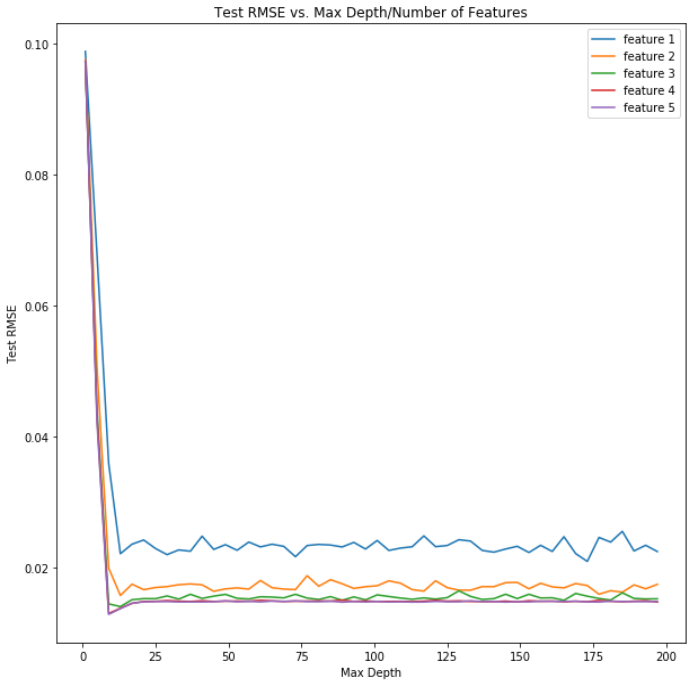

Figure 1-9 The Average RMSE Versus the Depth

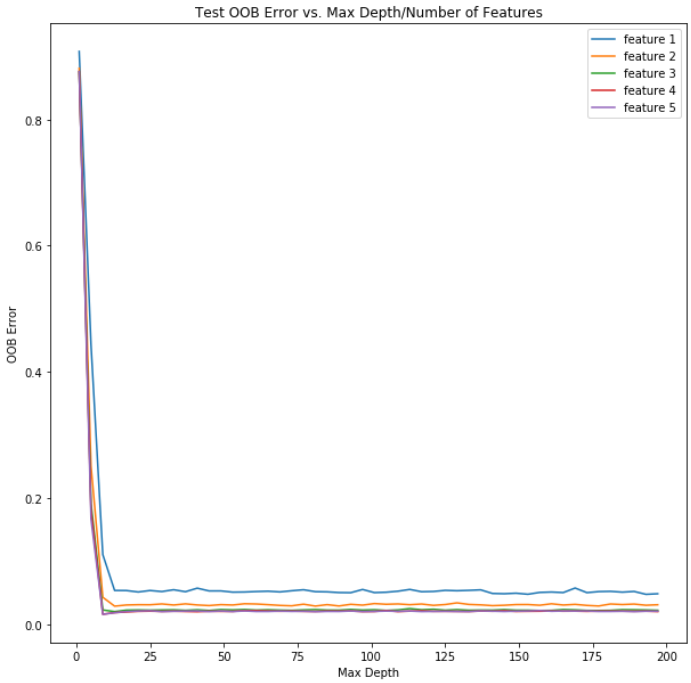

Figure 1-10 The Average OOB Error Versus the Depth

After trying different different hyper-parameters, we found the following hyper-parameters perform the best:

- Number of trees: 16
- Depth of each tree: 9
- Bootstrap: True
- Maximum number of features: 5

After using these hyper-parameters, the training RMSE, test RMSE, and Out Of Bag (OOB) error can be shown in table 1-3.

	Training RMSE	Test RMSE	OOB Error
Error	0.011578736626005472	0.012893892538313	0.015659298739203232

Table 1-3. The Training RMSE, Testing RMSE, and OOB error for the Random Forest Model with Best Parameter

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-11) as well as the the residuals versus fitted values image (which can be shown in figure 1-12) .

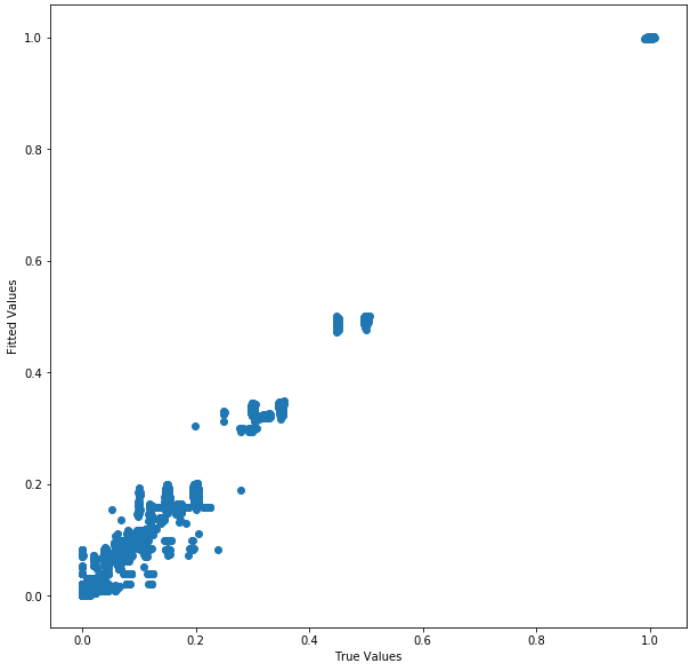

Figure 1-11. Fitted Values Against True Values of Best Random Forest Model

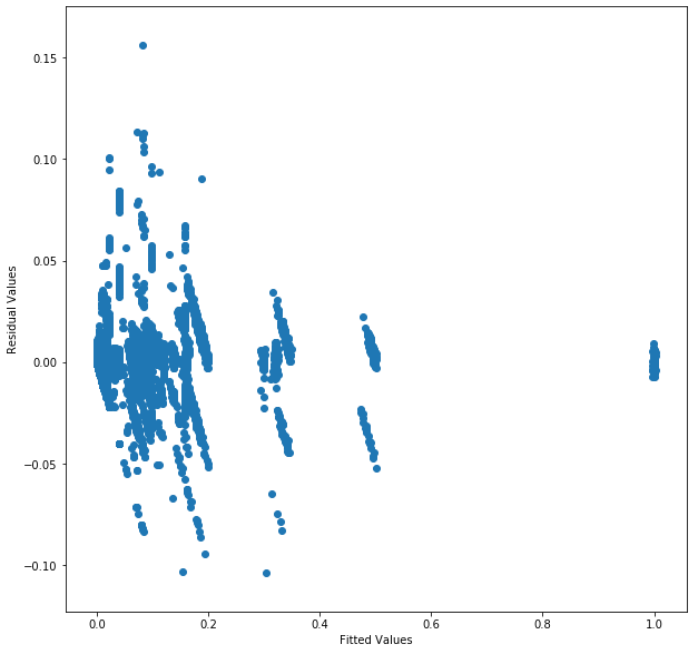

Figure 1-12. Residuals Against Fitted Values of Best Random Forest Model

(iv)

From the previous part, we found the following hyper-parameters perform the best:

- Number of trees: 16
- Depth of each tree: 9
- Bootstrap: True
- Maximum number of features: 5

With this parameter setting, the feature importances can be shown in table 1-4.

	Feature Importance
Week #	0.00162779
Day of Week	0.19567422
Backup Start Time - Hour of Day	0.39740765
Work-Flow-ID	0.18825149
File Name	0.21703885

Table 1-4 Feature Importances for Each Features

(v)

For part, we used the following parameter setting:

- Number of trees: 16
- Depth of each tree: 4
- Bootstrap: True
- Maximum number of features: 5

After the training process, we visualized the first tree in the random forest. The visualized tree can be shown in figure 1-13.

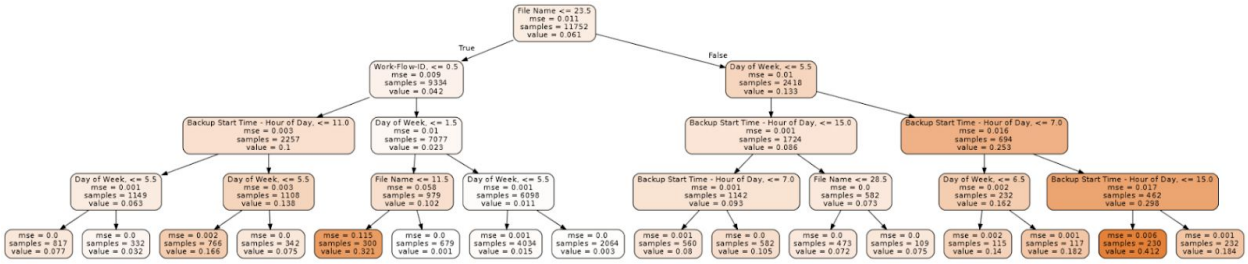

Figure 1-13 The Visualized Decision Tree

From the visualized decision tree, we can find that the root node is “File Name”. By examining the decision trees more carefully, we found that the root node is not the most important feature according to the feature importance reported by the regressor.

(c) Neural Network Model:

For this part, we used the one-hot encoding method to transforming the non-numerical values into numeric one. And then directly used the data to fit the neural network.

We tested different the neural network with different numbers of hidden units (from 2 to 600) as well as different activation function (ReLU, Logistic, and Tanh). The test RMSE result can be shown in figure 1-14.

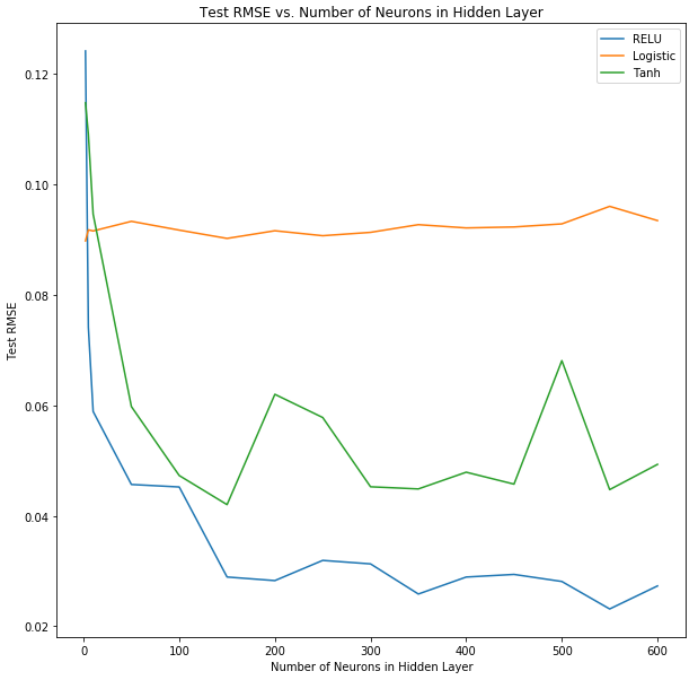

Figure 1-14 Test RMSE Versus the Number of Neurons in Hidden Layer (with Different Activation Function)

Thus, we can found out the neural network with ReLU activation and 550 neurons in the hidden layers performs the best. Thus we used this parameter setting to further do the test.

The training result and the testing result can be shown in the table 1-5.

	Training RMSE	Test RMSE
Error	0.011914308395812156	0.024671382691626113

Table 1-5. The Training and Testing RMSE for the Best Neural Network Model

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-15) as well as the the residuals versus fitted values image (which can be shown in figure 1-16) .

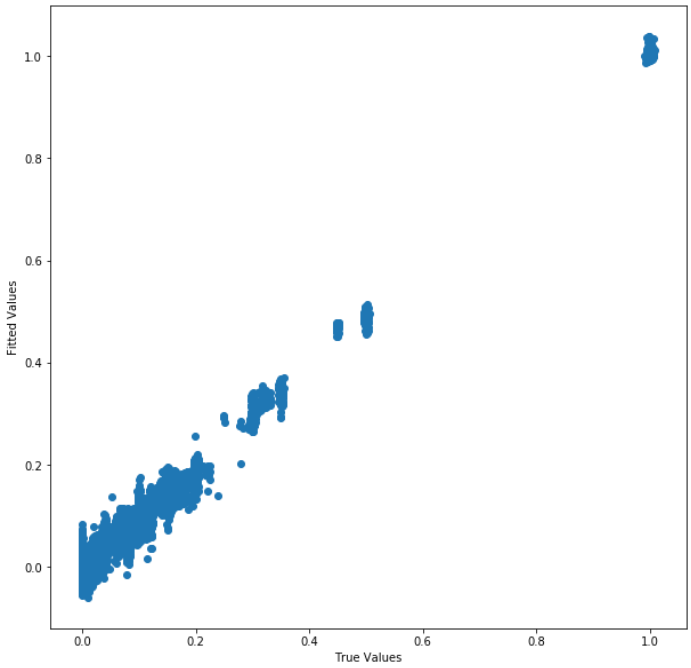

Figure 1-15. Fitted Values Against True Values of Best Neural Network Model

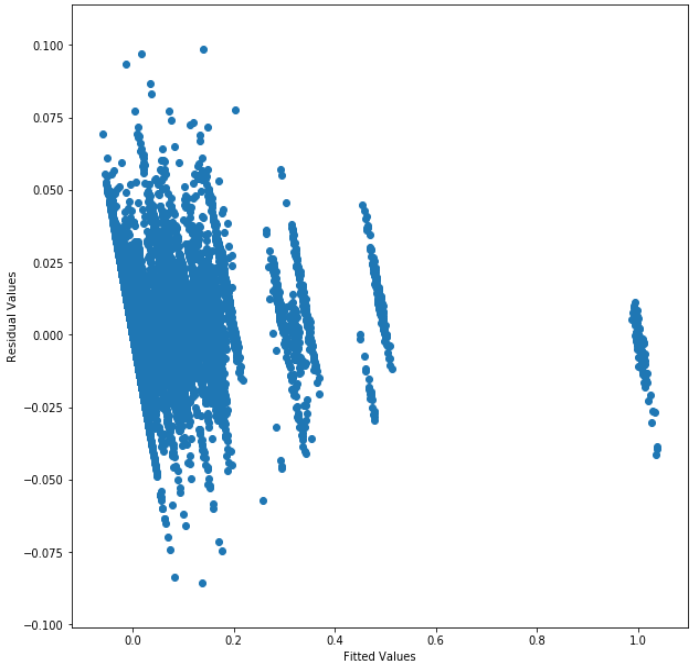

Figure 1-16. Residuals Against Fitted Values of Best Neural Network Model

(d) Linear Regression with Separate Workflow:

For this part, we used the scalar encoding method to transforming the non-numerical values into numeric one. However, for this part we separated the data according to the "Work-Flow-ID" feature, fitting the

model and testing model for each “Work-Flow-ID” set. Finally, to calculated the total RMSE, we used weighted average using the number for each “Work-Flow-ID” set and their corresponding RMSE.

(i)

The training result and the testing result for each workflow as well as the weighted average can be shown in the table 1-6.

	Training RMSE	Test RMSE
Workflow 1	0.035835233633998875	0.03586346655296642
Workflow 2	0.14874414236869582	0.14709758822040112
Workflow 3	0.04290319403417192	0.04257533084504969
Workflow 4	0.007242981713369212	0.007186683488524774
Workflow 5	0.08591402576289187	0.08533975477118065
Weighted Average	**0.06349460706251145**	**0.06298709748694042**

Table 1-6. The Training and Testing RMSE of Each Workflow using the Linear Regression Model

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-17) as well as the the residuals versus fitted values image (which can be shown in figure 1-18) .

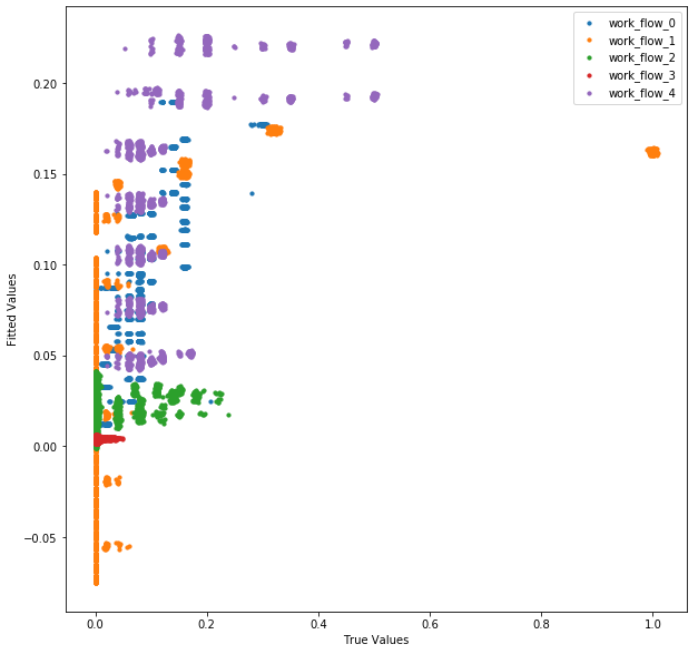

Figure 1-17. Fitted Values Against True Values of Linear Regression

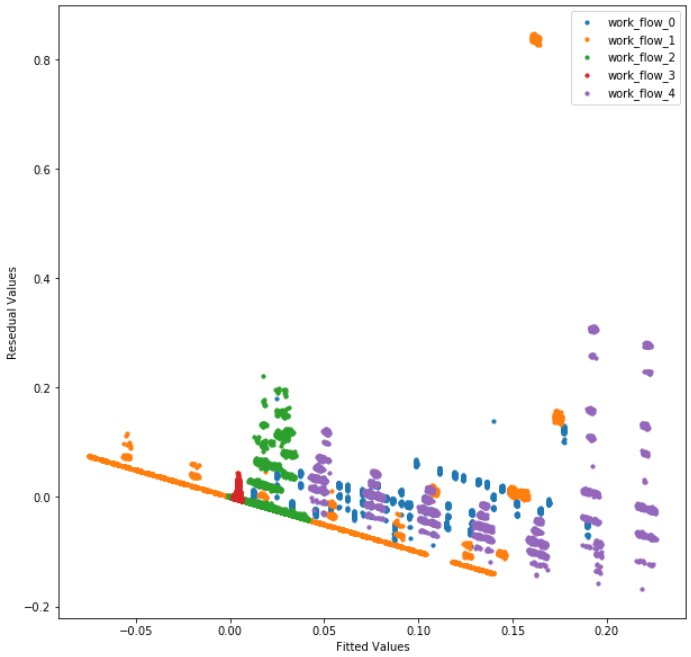

Figure 1-18. Residuals Against Fitted Values of Linear Regression

From the test result, it can be shown except the workfolow 2, the linear regression within the same workflow perform significantly better than the linear regression on the whole dataset. In general, predicting for each of the work flow separately performs better (Training RMSE 0.063 and Test RMSE 0.063) than predicting the backup size for all the workflows (Training RMSE 0.104 and Test RMSE 0.104).

(ii)

For this part, we used different polynomial degree to improve the fit. We plotted the training/test RMSE versus the polynomial degree. The result can be shown in figure 1-19.

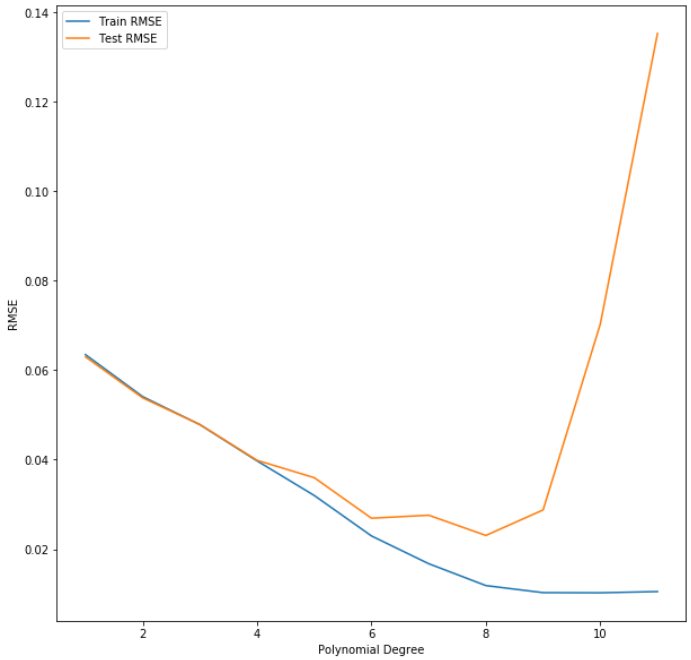

Figure 1-19 Training and Test RMSE Versus the Polynomial Degree

From this result, it can be shown that the training RMSE keep decreasing, while the test RMSE decreases first then increases. The threshold on the degree of the fitted polynomial beyond which the generalization error of your model gets worse is 8.

By applying cross validation, we can prevent the model from getting too complex. Because for training RMSE, the more complex the model is, the less training RMSE tends to be. However, this is not always happens for test RMSE. Becauses the model may get too complex so that it is not general enough, and thus perform poor on the test RMSE. To sum up, using the cross validation can prevent the model becoming too complex, which is also known as overfitting problem.

Finally, we tested the best parameter setting we found for this part. The best polynomial degree is 8. The training result and the testing result for each workflow as well as the weighted average can be shown in the table 1-7.

	Training RMSE	Test RMSE
Workflow 1	0.007924603263452543	0.014238454820523435
Workflow 2	0.008073166846162948	0.013095668673381727
Workflow 3	0.019311495562698094	0.03266696731068468
Workflow 4	0.004239204258165436	0.00834915635097767
Workflow 5	0.01951281496075079	0.04637969882140139

Weighted Average	**0.011852181537878904**	**0.02306454634217282**

Table 1-7. The Training and Testing RMSE of Each Workflow using the Linear Regression Model with Best Polynomial Degree

We also scatter plotted the fitted values against true values image (which can be shown in figure 1-20) as well as the the residuals versus fitted values image (which can be shown in figure 1-21) .

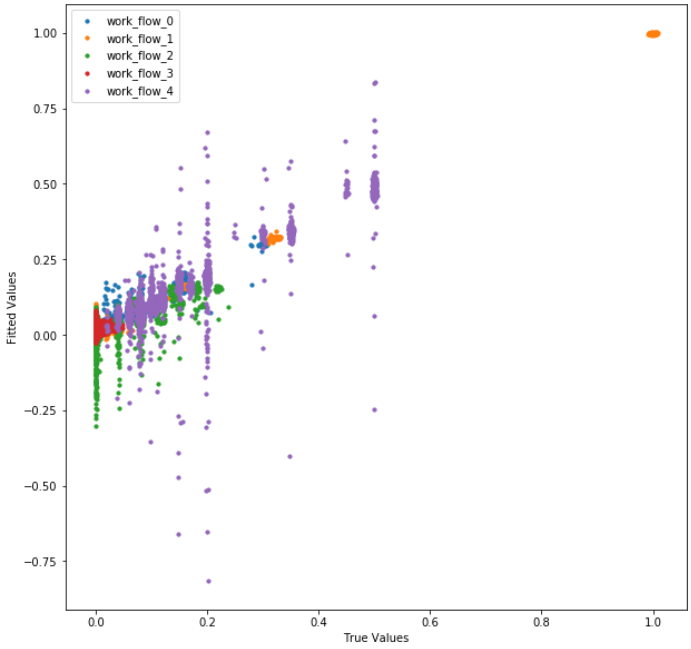

Figure 1-20. Fitted Values Against True Values of Linear Regression Model with Best Polynomial Degree

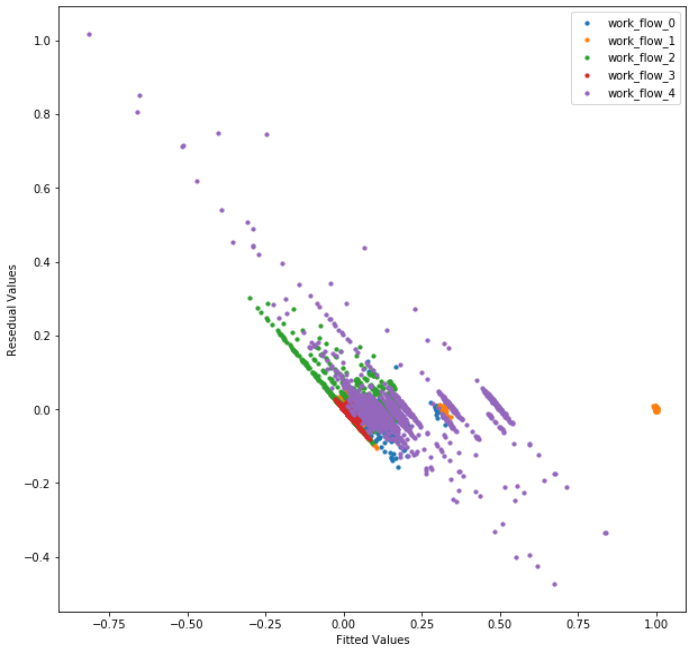

Figure 1-21. Residuals Against Fitted Values of Linear Regression Model with Best Polynomial Degree

(e) K-Nearest Neighbor Regression :

For this part, we used k-nearest neighbor regression model to train the data. First we still transformed data feature into numeric feature as before using scalar encoding. Then fitted the data to k-nearest neighbor with k varying from 1 to 200 in order to find out the best parameter k.

To evaluate the performance, average train and test RMSE were calculated using 10-fold cross validation. Plot of average train and test RMSE against different k was plotted as figure 1-22:

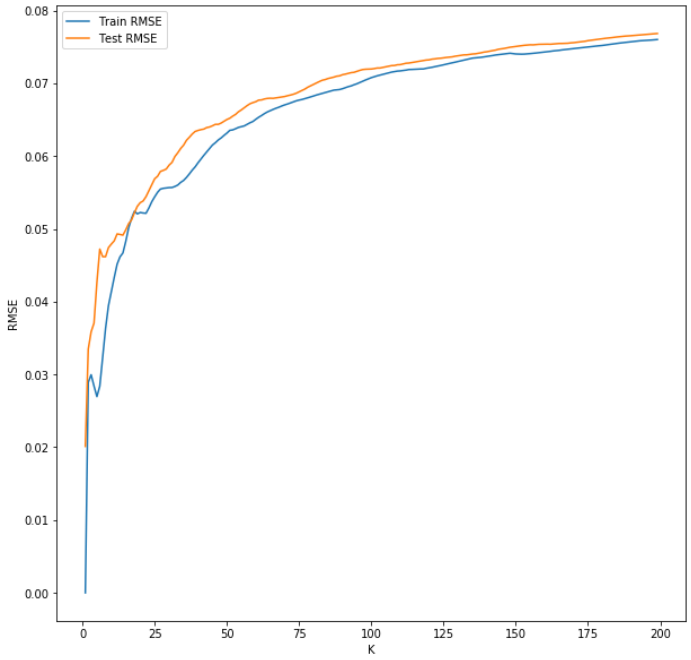

Figure 1-22. Average train and test RMSE using k-nearest neighbor regression model vs parameter k

As we could see from the result above, the performance was better when k = 1. So in the following steps, we chose k = 1 to fit the data to the model.

The average train and test RMSE was calculated using 10-fold validation. The result can be shown in table 1-8.

	Training RMSE	Test RMSE
Error	0.020119075932672044	0.020119075932672044

Table 1-8. The Training and Testing RMSE for the K-Nearest Neighbours with Best K

We also made scatter plot of fitted values against true values (can be shown in figure 1-23) and scatter plot of residual values against fitted values (can be shown in figure 1-24).

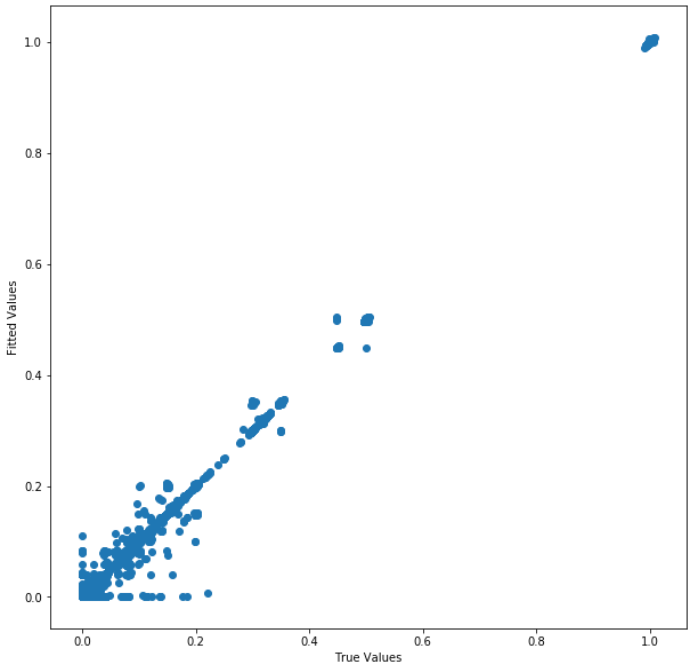

Figure 1-23. Scatter plot of fitted values vs true values

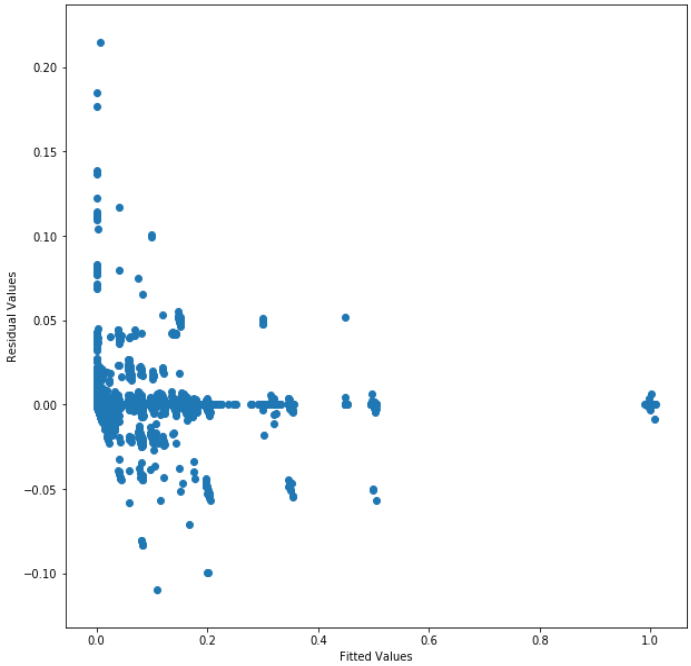

Figure 1-24. Scatter plot of residual values vs fitted values

Q3: Comparison Among Different Models

For dataset 1, we tested several models and the result can be summarized as table 1-9.

Models	Training RMSE	Test RMSE
Linear Regression (Predict the Whole Dataset)	0.103584726740266	0.10362189439542643
Linear Regression (Predict the Dataset for Each Workflow Separately)	0.06349460706251145	0.06298709748694042
Linear Regression with Best Polynomial Degree (Predict the Dataset for Each Workflow Separately)	0.011852181537878904	0.02306454634217282
Random Forest with Best Parameter Setting	0.011578736626005472	0.012893892538313
Neural Network with Best Parameter Setting	0.011914308395812156	0.024671382691626113
K-Nearest Neighbours with Best K	0.020119075932672044	0.020119075932672044

Table 1-9. The Training and Test RMSE Using Different Models

From table 1-9, it can be shown that the random forest perform the best overall, giving a good performance with test RMSE equal to 0.01.

Besides, the neural network regression model is better in taking care of sparse features. And random forest is good at handling categorical features.

Dataset 2

This dataset concerns housing values in the suburbs of the greater Boston area. There are 506 data points with the following features:

- CRIM: per capita crime rate by town
- ZN: proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS: proportion of non-retail business acres per town
- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX: nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior to 1940
- DIS: weighted distances to five Boston employment centers
- RAD: index of accessibility to radial highways

- TAX: full-value property-tax rate per $10,000
- PTRATIO: pupil-teacher ratio by town
- B: $1000\,(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT: % lower status of the population
- MEDV: Median value of owner-occupied homes in $1000's

Q1: Load the dataset.

First a few rows of the dataset and their corresponding values are shown below in Fig 2-0.

index by column numbers:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	

	B	LSTAT	MEDV
0	396.90	4.98	24.0
1	396.90	9.14	21.6
2	392.83	4.03	34.7
3	394.63	2.94	33.4
4	396.90	5.33	36.2

Figure 2-0. First Five Rows of Dataset

Q2: Fit a linear regression model:

(a) We set MEDV as the target variable and the other attributes as the features and ordinary least square as the penalty function.

(b) We performed a 10 fold cross validation and got the averaged Root Mean Square Error (RMSE) shown below:

Averaged Train RMSE: 4.609066917948425

Averaged Test RMSE: 5.180845679340264

We then plotted

1) fitted values against true values as scatter plots using the whole dataset (the y-axis is the fitted value and x-axis is the true value). The plot is shown in Fig 2-1.

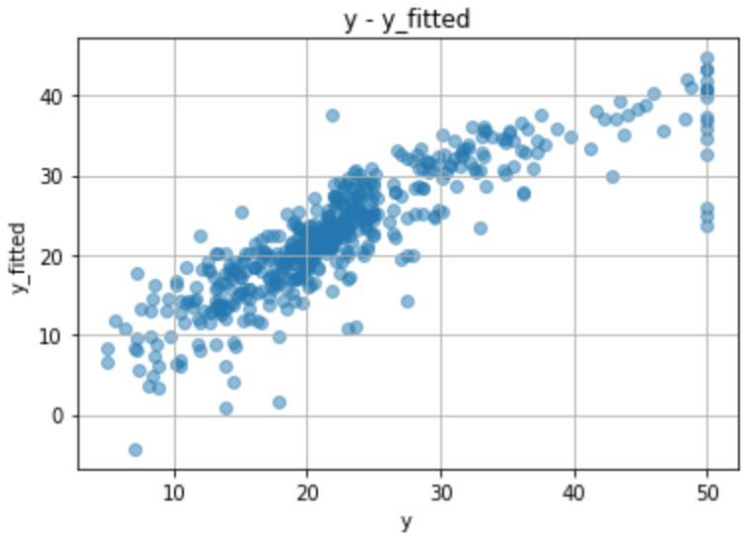

Figure 2-1. Fitted Values Against True Values

2) residuals versus fitted values as scatter plots using the whole dataset (the y-axis is the residual and x-axis is the fitted value). The plot is shown in Fig 2-2.

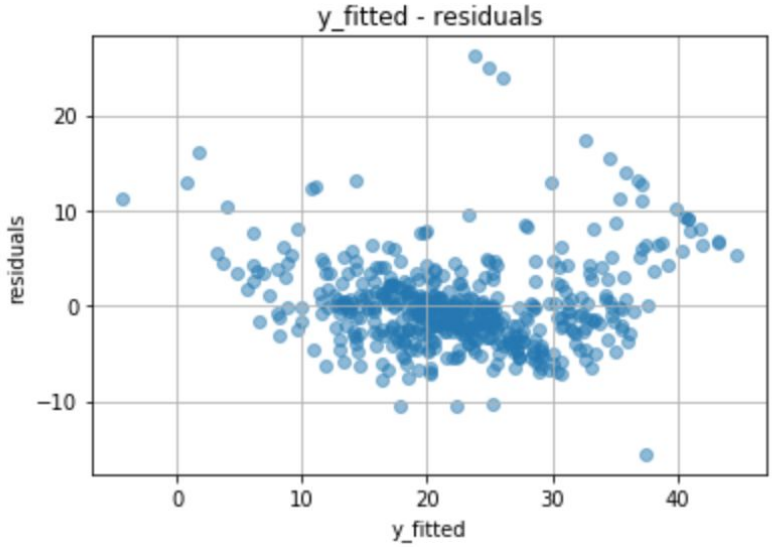

Fig 2-2. Residuals Against Fitted Values

Q3: Try to control overfitting via regularization of the parameters:

(a) We performed one-hot encoding on all features, then fit the model and tested it. The train RMSE and test RMSE are shown below.

Averaged Train RMSE: 3.5601545574706854e-14

Averaged Test RMSE: 8.003143832522671

(b) It is obvious that the training RMSE is almost 0 but we get really large test RMSE. This is the result of overfitting, when on the training set, the predicted value is almost the same as the true

value, but on test set the predicted value has a large difference to true value. To reduce the effect of overfitting, one way is to try regularization on our models.

(c) We tried the following regularizations:

(1) Ridge Regularizer:

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

We tried the parameter α from 0.1 to 1.0 with an interval of 0.05 and used RidgeCV to find the best α. Then we applied 10 fold cross validation to obtain the averaged train and test RMSE. The best α found is

$$\alpha = 0.1$$

And the averaged train and test RMSE are shown below:

Average Train RMSE: 0.0910103758003489

Average Test RMSE: 8.00442085716902

(2) Lasso Regularizer:

$$\min_{\beta} \|Y - X\beta\|^2 + \alpha \|\beta\|_1$$

We tried the parameter α from 0.1 to 1.0 with an interval of 0.05 and used LassoCV to find the best α. Then we applied 10 fold cross validation to obtain the averaged train and Test RMSE. The best α found is

$$\alpha = 0.1$$

And the averaged train and test RMSE are shown below:

Average Train RMSE: 6.240467370263949

Average Test RMSE: 8.200368237047693

(3) Elastic Net Regularizer:

$$\min_{\beta} \|Y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$$

We tried the parameter α from 0.1 to 1.0 with an interval of 0.05, l1 ratio with following values: 0.1, 0.5, 0.7, 0.8, 0.9, 0.92, 0.94, 0.96, 0.98, 0.99, 1.0 and used ElasticNetCV to find the best α and l1 ratio. Then we applied 10 fold cross validation to obtain the averaged train and test RMSE. The best α and l1 ratio found are

$$\alpha = 0.1$$

$$\text{l1 ratio} = 0.1$$

Thus $\lambda_1 = 0.01$, $\lambda_2 = 0.045$ respectively.

And the averaged train and test RMSE are shown below:

Average Train RMSE: 6.15734337709661

Average Test RMSE: 8.105309661930113

We can see that with regularization, the overfitting problem is well solved. Ridge regularizer still has kind of overfitting, but Lasso regularizer and Elastic Net regularizer solve the problem well.

Dataset 3

Dataset 3 is one car insurance dataset. The models were trained to predict user's car insurance charges given the 6 features from the user's profile. As shown below, each row contains one user's 6 features and car insurance charges. Among those features, ft1, ft2 and ft3 are numerical features, while ft4, ft5 and ft6 are categorical features.

	ft1	ft2	ft3	ft4	ft5	ft6	charges
0	19	27.900	0	female	yes	southwest	16884.92400
1	18	33.770	1	male	no	southeast	1725.55230
2	28	33.000	3	male	no	southeast	4449.46200
3	33	22.705	0	male	no	northwest	21984.47061
4	32	28.880	0	male	no	northwest	3866.85520

Figure 3-0. Car insurance DataFrame

Q1: Feature Preprocessing

In this part, dataset 3, one car insurance dataset, was preprocessed using three different methods respectively: Feature Encoding (Use one-hot-encoding for ft4, ft5, ft6 categorical features), Standardization (Standardize all other numerical features which are ft1, ft2, ft3) with ft4, ft5, ft6 one-hot-encoded, Division for ft1 (Divide ft1 into 3 ranges, assign new values like 1 for ft1 < 30, 2 for ft1 $\in [30, 50]$ and 3 for ft1 > 50) with ft2, ft3 standardized and ft4, ft5, ft6 one-hot-encoded. Then we fitted those preprocessed data to a linear regression model. Root Mean Squared Error (RMSE) was used to evaluate the performance of the model.

The following results show the average training RMSE and average test RMSE for 10-fold cross validation.

	Feature Encoding	Standardization	Division for ft1
Training RMSE	6039.342370581925	6039.342370581925	6197.851883305363
Test RMSE	6063.643882478562	6063.643882478562	6221.8316433761975

Table 3-1. Average RMSE from training set and test set using three different feature preprocessing methods

The scattered figures were plotted using the whole dataset by different feature preprocessing methods respectively. One figure shows fitted values against true values, while the other shows residuals versus fitted values as shown below:

(1) Feature Encoding method:

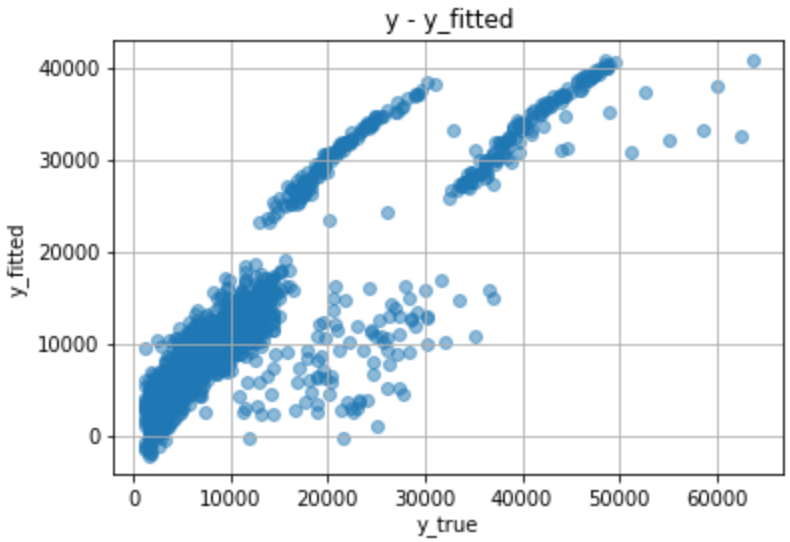

Figure 3-1. Scatter plot of fitted values vs true values using feature encoding method

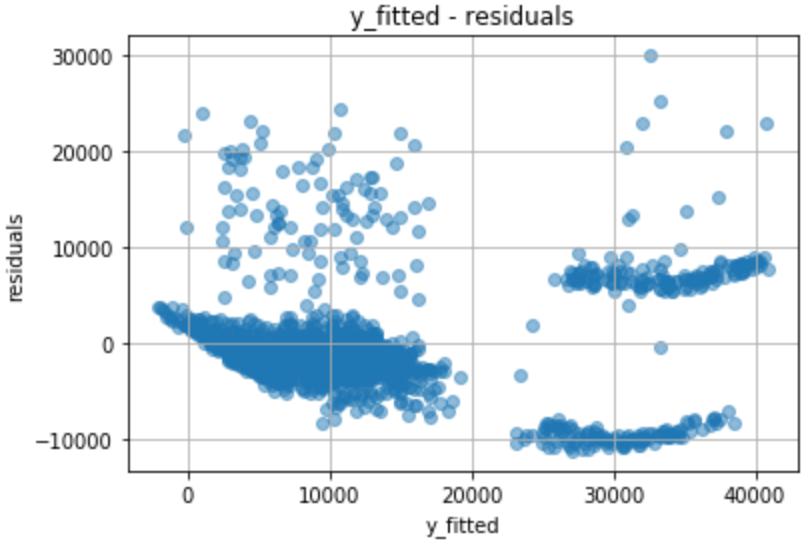

Figure 3-2. Scatter plot of residuals vs fitted values using feature encoding method

(2) Standardization method:

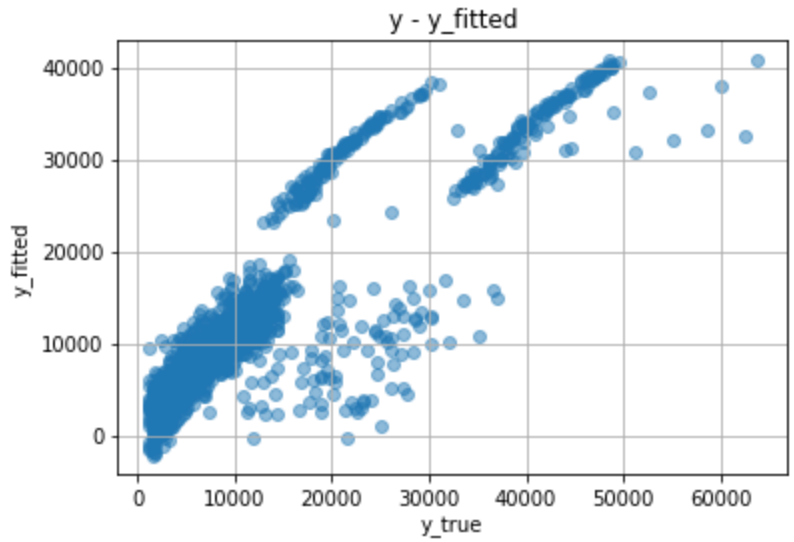

Figure 3-3. Scatter plot of fitted values vs true values using standardization method

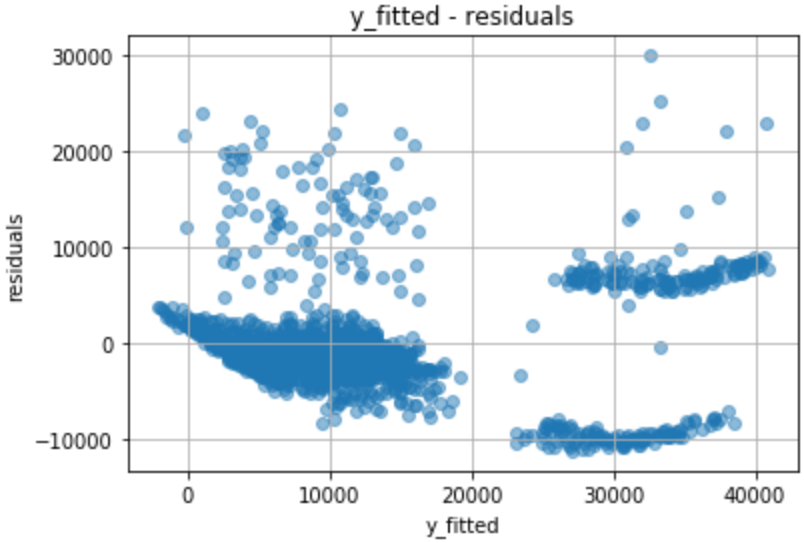

Figure 3-4. Scatter plot of residuals vs fitted values using standardization method

(3) Division for ft1 method:

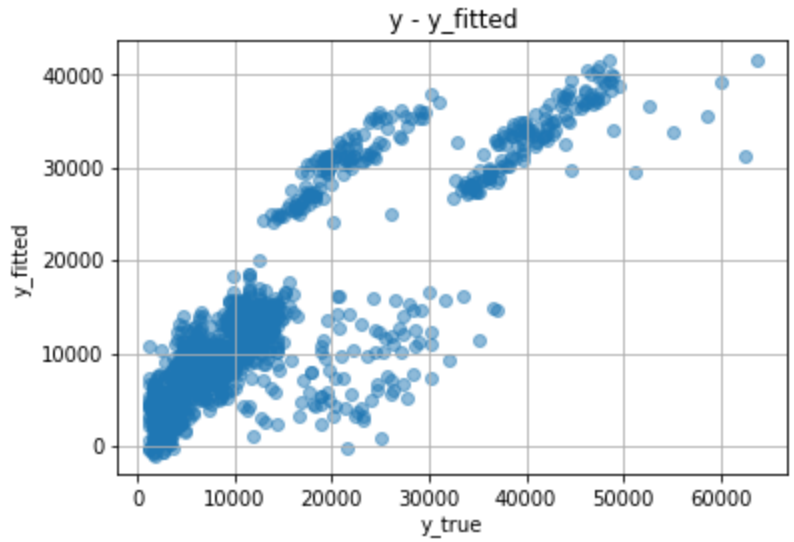

Figure 3-5. Scatter plot of fitted values vs true values using division for ft1 method

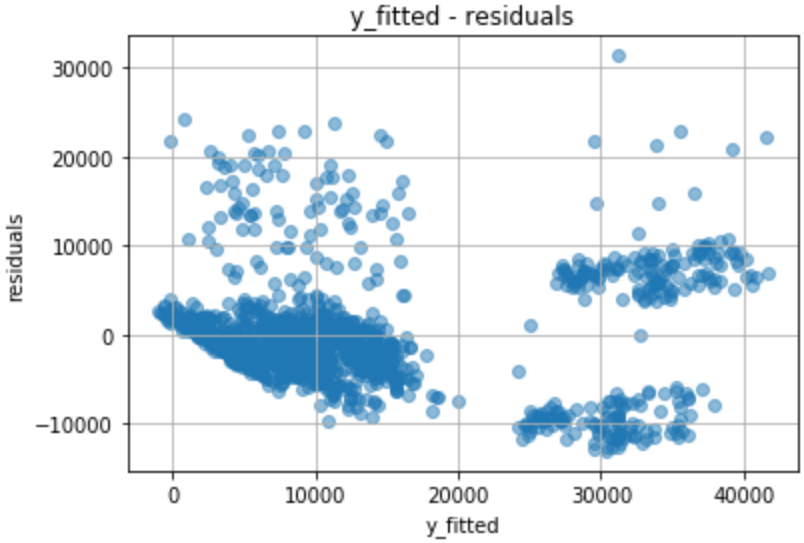

Figure 3-6. Scatter plot of residuals vs true values using division of ft1 method

Q2: Correlation Exploration

In this part, each categorical feature was converted into a one dimensional numerical value. And we have 6 numerical features: [ft1, ft2, ft3, ft4, ft5, ft6]. In order to select two most important variables, two methods were used here: f_regression and mutual information regression. The two most important variables were calculated using two methods respectively, and the results are shown as below:

	f_regression	Mutual-info regression

	Selected ft	f-score	p-value	Selected ft	Mutual-info
Top 1 ft	ft 5	2177.61486806	8.271436e-283	ft 1	1.49869931567
Top 2 ft	ft 1	131.174012580	4.8866933e-29	ft 5	0.36917105310

Table 3-2. Top 2 important features selected by two different methods

As the results shown above, the top 2 most relevant features were selected the same between both methods, ft1 and ft5.

Then we plotted charges (y axis) vs ft2 (x axis) as scatter plot and charges (y axis) vs ft1 (x axis) as scatter plot with color points based on ft5 (Yes or No). The plots are shown as below:

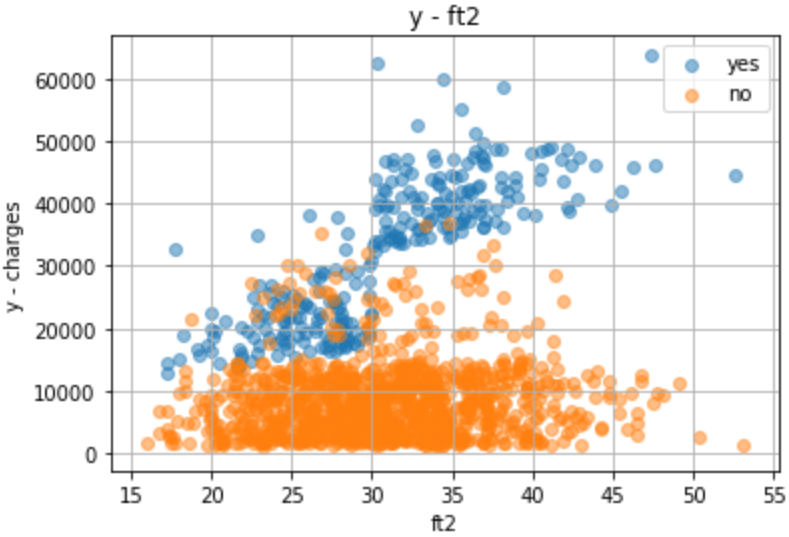

Figure 3-7. Scatter plot of charges vs ft2 based on ft5

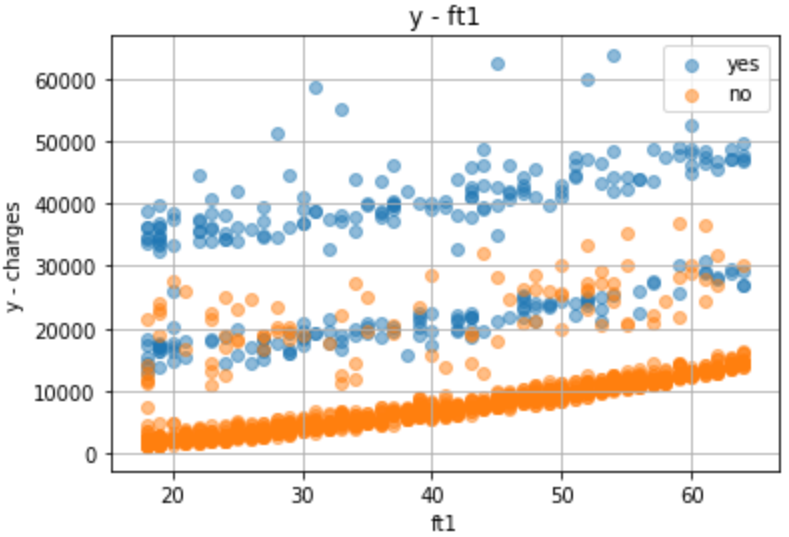

Figure 3-8. Scatter plot of charges vs ft1 based on ft5

Q3: Target Variable Modification

As we have been so far, the target variable: charges (y) spans a wide range, so here instead of fitting the original value, we consider fitting $log(y)$. Instead of calculating the difference between predicted value ($log(y)_{predict}$) and transformed target values ($log(y)$), we calculated the difference between $exp(log(y)_{predict})$ and y to set up a fair comparison.

Here we chose feature encoding method from Q1 to preprocess the data and train a linear regression model on this new target. The results are shown as below:

Averaged Train RMSE: 8358.940954573565

Averaged Test RMSE: 8373.795726572054

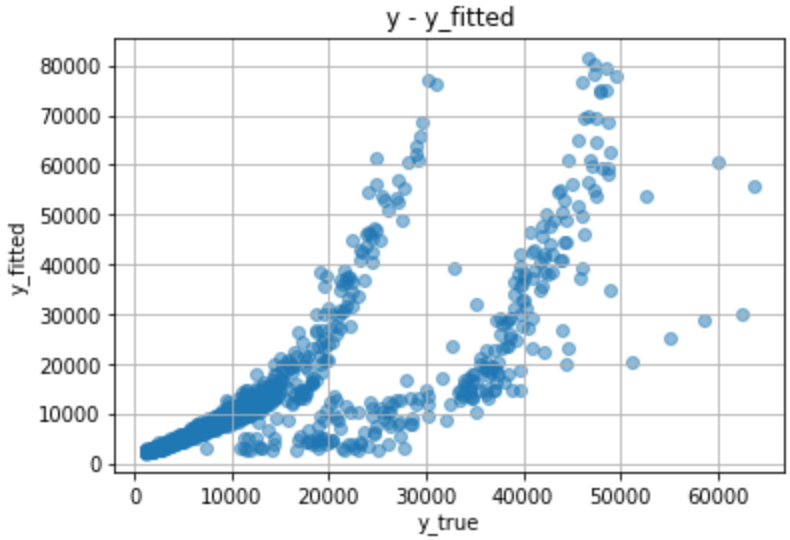

Figure 3-9. Scatter plot of fitted values vs true values with new target

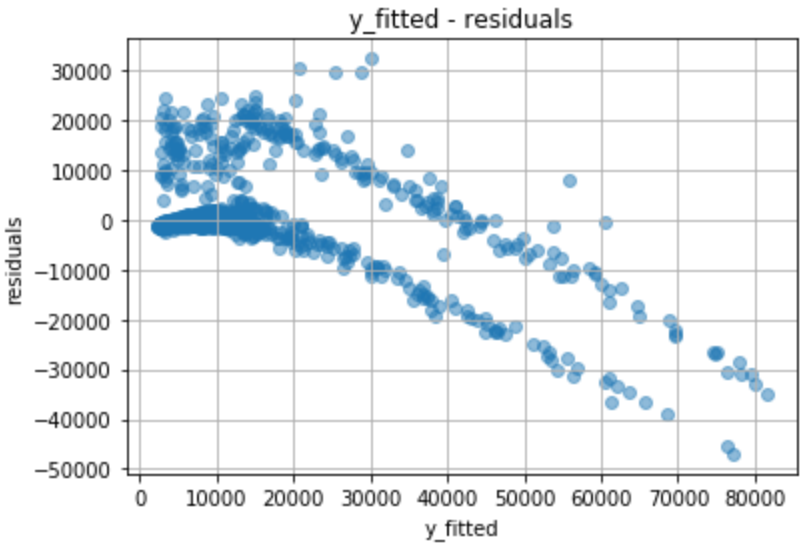

Figure 3-10. Scatter plot of residuals vs fitted values with new target

Then the correlation exploration part was repeated using the new target. The new plots are shown as below:

	f_regression			Mutual-info regression	
	Selected ft	f-score	p-value	Selected ft	Mutual-info
Top 1 ft	ft 5	1062.12392292	6.307646e-172	ft 1	1.50285110685
Top 2 ft	ft 1	515.977081123	7.4773852e-97	ft 5	0.36939335267

Table 3-3. Top 2 important features selected by methods in Q2 using new target values

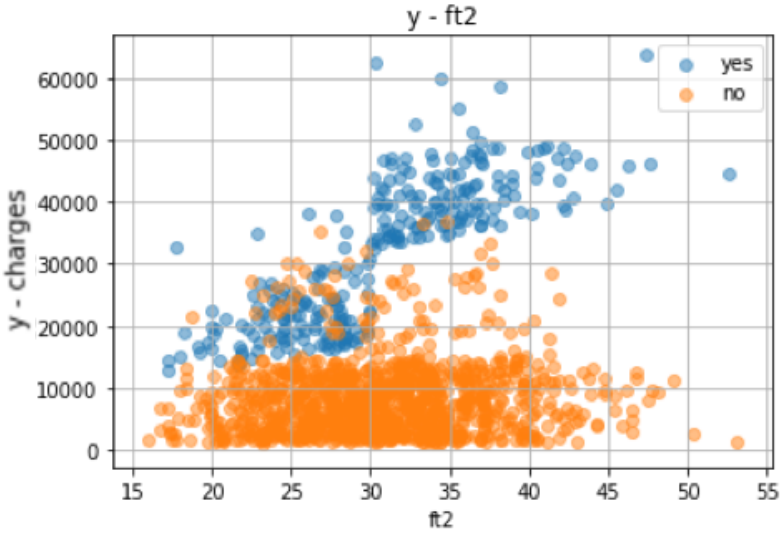

Figure 3-11. Scatter plot of true y values vs ft2 based ft5 using new target

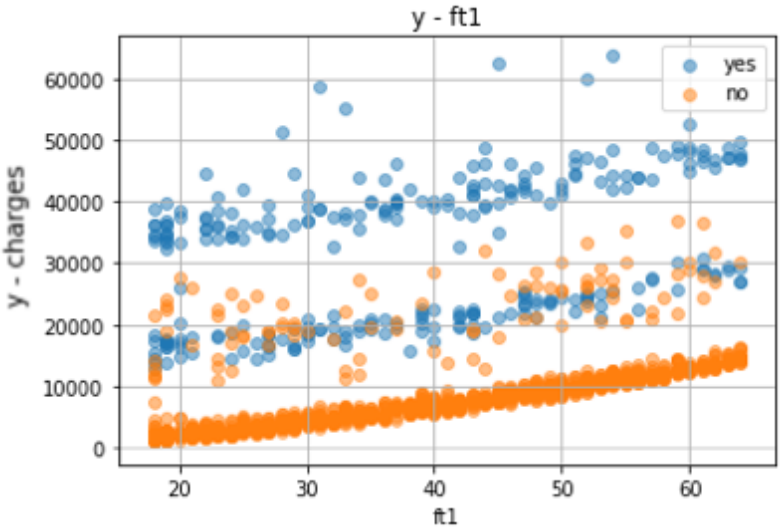

Figure 3-12. Scatter plot of true y values vs ft1 based on ft5 using new target

As the results shown above, comparing to the results using feature encoding preprocessing method from Q1, we could see that averaged train and test RMSE are about 8000 which is bigger than that in Q1 (about 6000). So from this point, the results of using new target is worse than that using original target. From the scatter plots, comparing to the ones using original target, we could see that the predicted values using new target tends to be smaller than true values, while the results using original target tend to overpredict. As for correlation exploration, the results are almost the same. The most relevant features are still ft5 and ft1.

Q4: Bonus questions

Considering current results, we tried to improve our results by using polynomial features to better preprocess the data and fit a linear regression model to train. Here we first performed one-hot-encoding on ft4, ft5, ft6 and keep the numerical features. Then the data were further processed using polynomial featuring with degrees ranging from 1 to 5. For target values here, we used new target ($log(y)$) and calculated the difference between $exp(log(y)_{predict})$ and y as in Q3. To evaluate the performance, we still used 10-fold cross validation to train and fit a linear regression model as well as calculating the average train and test RMSE value. The results are shown as below:

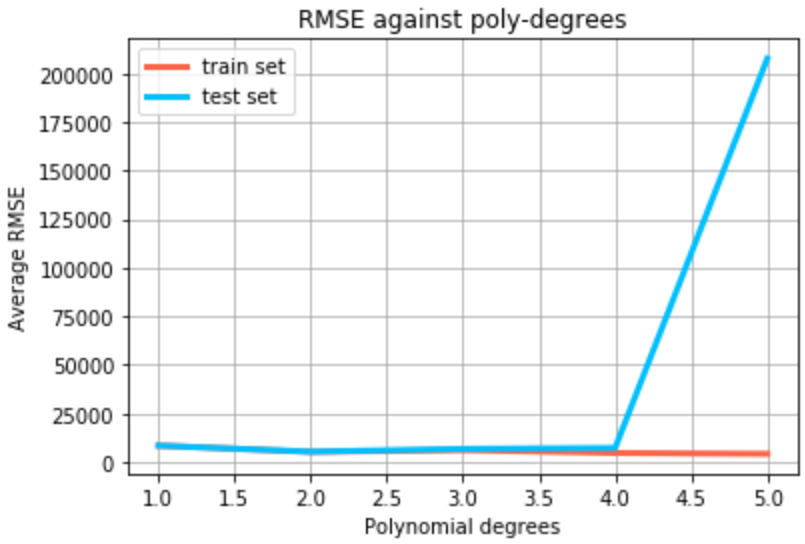

Figure 3-13. Average train and test RMSE vs degrees from polynomial featuring

Polynomial degrees	Avg train RMSE	Avg test RMSE
1	8358.940954573554	8373.795726572038
2	5180.103424400011	5270.074041837238
3	6041.856013541929	6706.863479360283
4	4581.788743408271	7197.909619878288
5	4158.5824683004885	207711.47665301443

Table 3-4. Avg train and test RMSE calculated using different polynomial degrees

From the results shown above, both train and test RMSE shows relatively better performance when degree equals to 2. And the results are better than that in Q1, which means the feature encoding is better than before when using polynomial featuring with degree=2. But as degree increases, the performance tends to be overfitting.

Now we tried to further improve our results by picking a better model. Here three more models were tried here: random forest, neural network and gradient boosting tree model. And grid-search method was used here to find out better parameters of different models we tried. After finding out the best parameters of each model, we still performed 10-fold cross validation to fit the data to the model. And the data was still preprocessed by the feature encoding method in Q1. The modified new target as in Q3 was used here to set up a fair comparison between the new model and previous used linear regression model in Q3. Finally, average train and test RMSE were calculated to evaluate the performance. The results are also shown below:

	Random Forest	Neural Network	Gradient Boosting Tree
Best Params	'max_depth': 7, 'max_features': 7, 'n_estimators': 65	'activation': 'tanh', 'hidden_layer_sizes': 150	'max_depth': 3, 'n_estimators': 53
Avg Train RMSE	3767.2162683898628	5614.695420919403	4246.623025662259
Avg Test RMSE	4463.216223705728	5707.820858413784	4525.7691549919255

Table 3-5. Best parameters and average train and test RMSE from 3 different models

As the results shown above, all of the three models show better results comparing to the linear regression model used in Q3 with RMSE about 8000. Among the three models, random forest model gives the best performance with number of trees = 65, depth of each tree = 7, maximum number of features = 7 and bootstrap = True.