Project 4: Regression Analysis

Zeyu Bai, 904514671 Yi-Jui Chang, 804587644
Nan Liu, 004434221 Ruizhi Yang, 704514506

Winter 2019

1 Introduction

In this project we applied regression analysis on three different datasets to explore feature encoding techniques and handle overfitting problems. Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. A variety of models including linear regression, random forest, neural network and k-NN are trained and tested. Feature preprocessing is also found to have significant importance in improving model predictions. Standard feature preprocessing techniques such as scalar encoding ,one-hot encoding etc. are implemented. Generally speaking, random forest model gives the best prediction results.

2 Dataset 1: Network backup dataset

1. Load the dataset

Figure 1 plots first 20 day backup size for all workflows from 0 to 4, figure 2 plots all 105 days backup size for all workflows from 0 to 4. From the plots we do observe repeating patterns, and the period for each workflow is around 7 days.

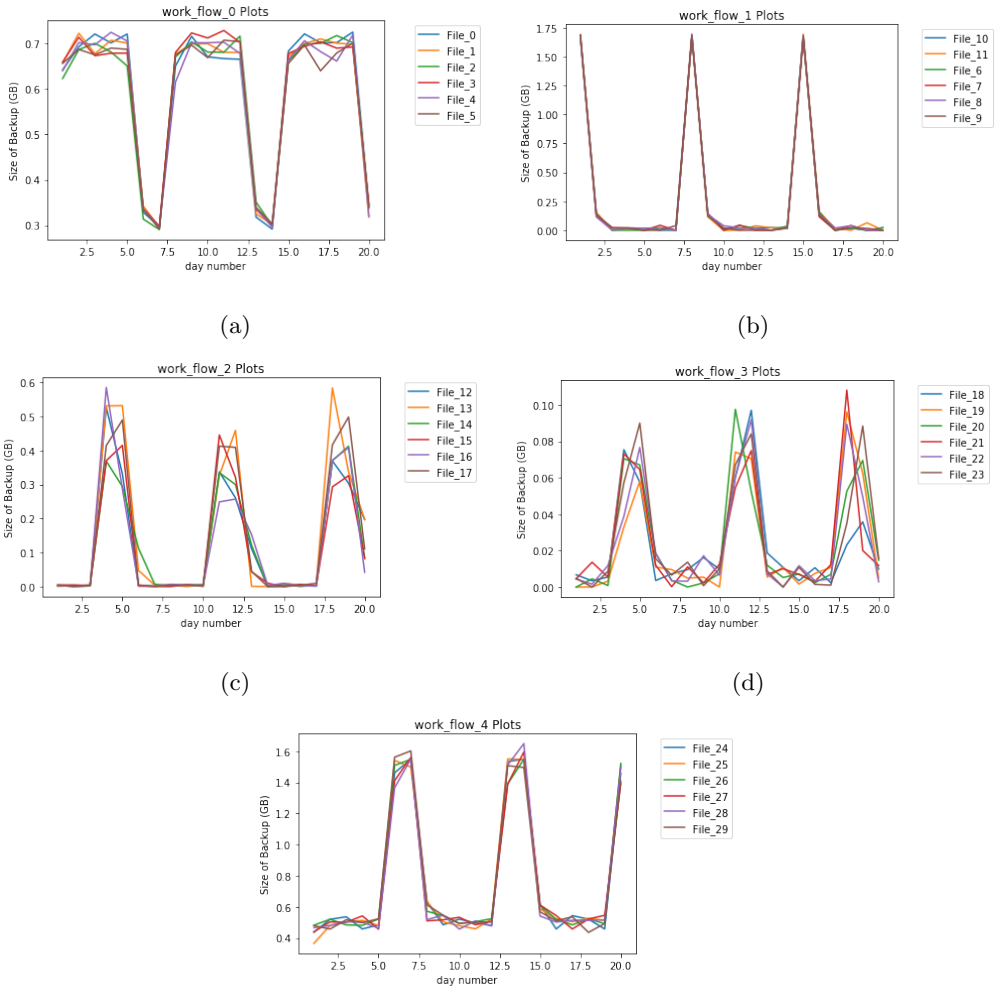


Figure 1: First 20 day backup size for all workflows

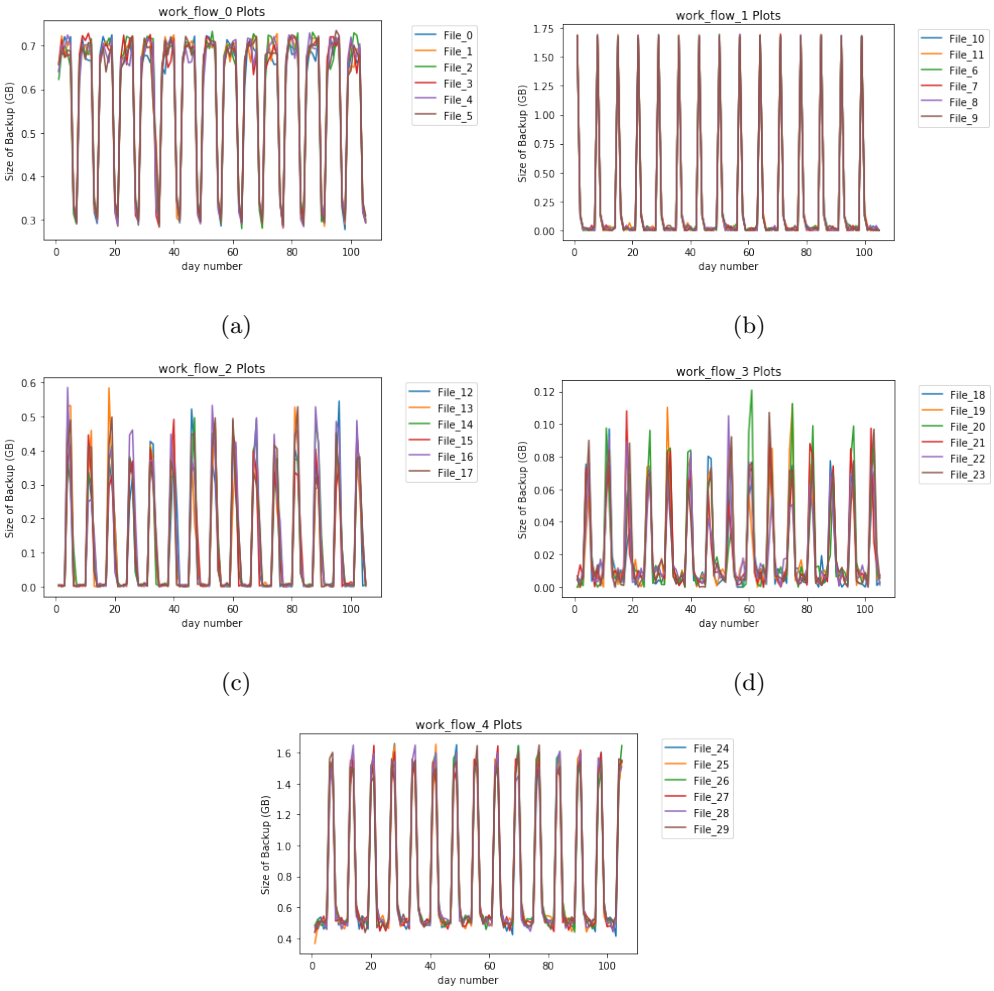


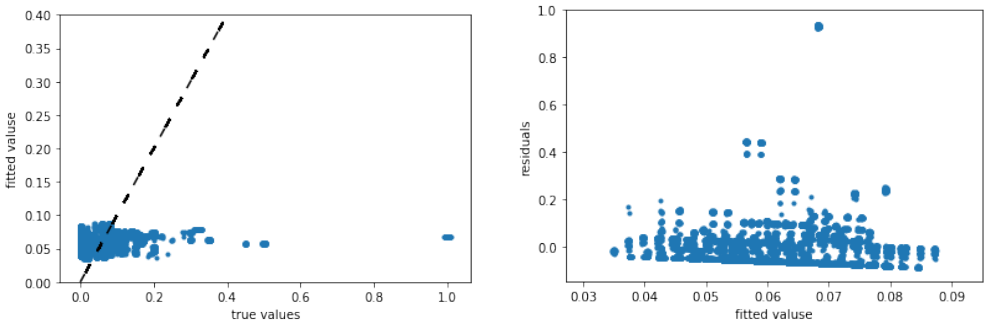
Figure 2: All 105 days backup size for all workflows

2.(a) Linear regression

In this part a linear regression model is built to predict network backup size. Features with strings are removed, and scalar encoding is used to transform week days to numbers 1-7. Training and testing RMSE from 10-fold cross validation are reported in table 1, scatter points of fitted values vs true values and residual vs fitted values are reported in figure 3. Dashed lines in figure 3 indicates the ideal fitting while scatter points show the actual fitting. From the plots we can see that linear regression performance is not very good.

Table 1: RMSE for linear regression

Training RMSE	Testing RMSE
0.1036	0.1034



(a) (b)

Figure 3: Scatter points of (a)fitted values vs true values and (b)residual vs fitted values

2.(b) Random forest

In this part random forest model is built to predict network backup size. Random forest differs from decision tree in the sense of bootstrap aggregation(bagging), which randomly select a subset of data and also a subset of features used to train each tree. An ensemble method is used to combine the predictions from multiple trees, thus variance can be effectively reduced. With the nature of random sampling of data, the part of data that are not selected can be used as internal testing data, which return the out-of-bag error. It turns out that the out-of-bag error has similar evaluation to RMSE. No feature encoding is done in this part since random forest can handle categorical data itself.

i. RMSE from 10-fold CV

Using the following hyper-parameters, training and testing RMSE from 10-fold cross validation are reported in table 2, together with mean out-of-bag error in the same table. It can be observed that RMSE of random forest model is smaller than that of linear regression. Scatter plots are

presented with optimal hyper-parameter found through grid search in later paragraphs.

- Tree number: 20
- Tree depth: 4
- Max features for each node: 5

Table 2: RMSE and OOB for random forest

Training RMSE	Testing RMSE	OOB
0.0606	0.0608	0.3407

ii. Hyper-parameter grid search of tree number and max feature

Here a grid search for tree number from 1 to 200 and maximum feature from 1 to 5 is done to find out best prediction combination. Figure 4 shows out-of-bag error against tree number with different lines for different max feature number. Figure 5 shows testing RMSE versus tree number with different lines for different max feature number. The best set of parameters found by minimum OOB 0.3337 and minimum testing RMSE 0.0602 are the same, which is tree number as 85 and feature number as 3.

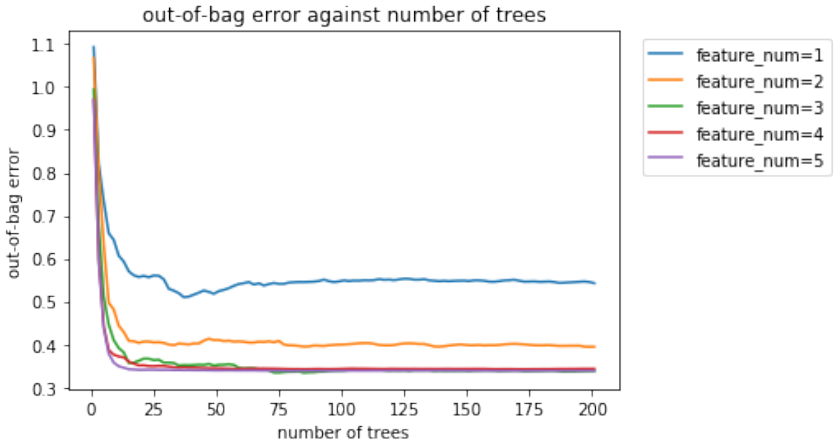


Figure 4

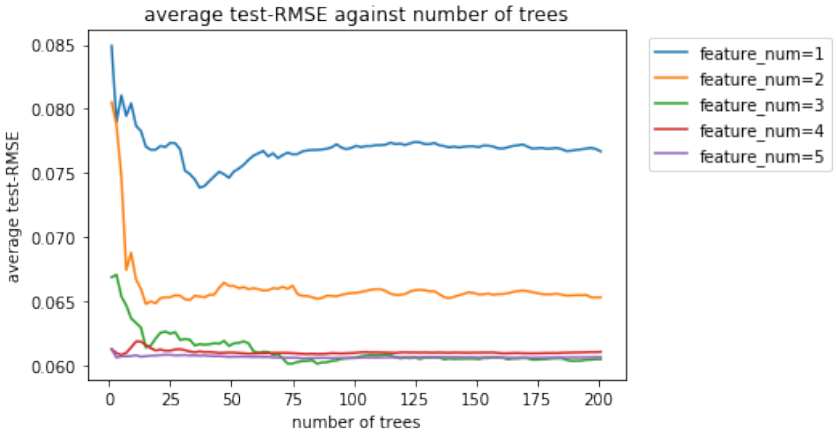


Figure 5

iii. Hyper-parameter grid search of tree depth and max features

Similarly, tree depth is chosen as hyper-parameter for grid search. Choosing optimal tree number as 85 found in the last part, we swept over tree depth from 1 to 21 and max feature from 1 to 5. Figure 6 shows out-of-bag error against tree depth with different lines for different max feature number. Figure 7 shows testing RMSE versus tree depth with different lines for different max feature number. The best set of parameters found by minimum OOB 0.0150 and minimum testing RMSE 0.0127 are the same, which is tree depth as 9 and feature number as 5 . Combined with results from last part, the best hyper-parameters found are

- Tree number: 85
- Tree depth: 9
- Max features for each node: 5

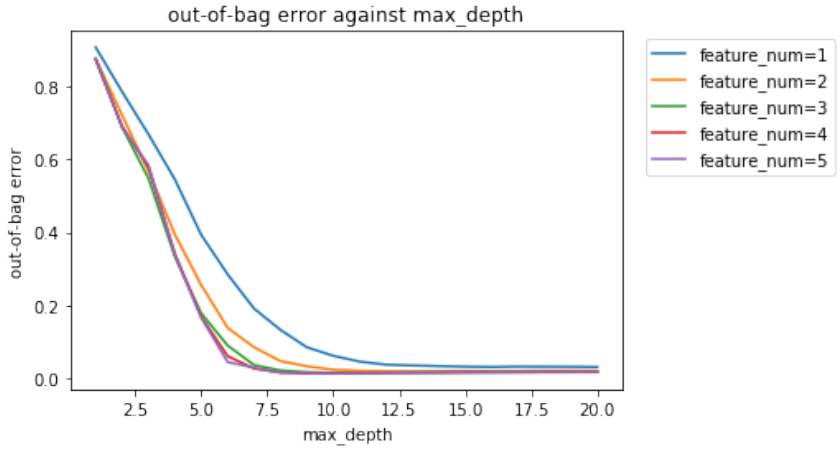


Figure 6

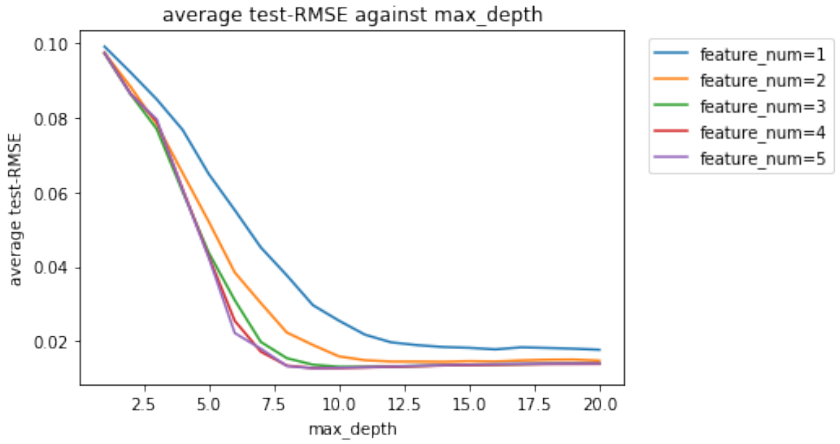


Figure 7

iv. Feature Importance

In scikit-learn, feature importance calculated with random forest model is implement as gini importance or mean decrease impurity. It is defined as the total decrease in node impurity weighted by the probability of reaching that node averaged over all trees of the ensemble. Roughly speaking, the more proportion of samples reach that node, the higher value of feature importance. So higher feature importance reveals that the feature is more importance. Sorted by value, results are shown in table 3. From the table we can see that week number is of least importance. It makes sense since backup difference from week to week are not large.

Table 3: feature importance of random forest model with tree depth=9

	hour of day	file name	day of week	work-flow-ID	week number
Importance ranking	1	2	3	4	5
score	0.3966	0.2766	0.1998	0.1254	0.0016

v. Decision tree visualization

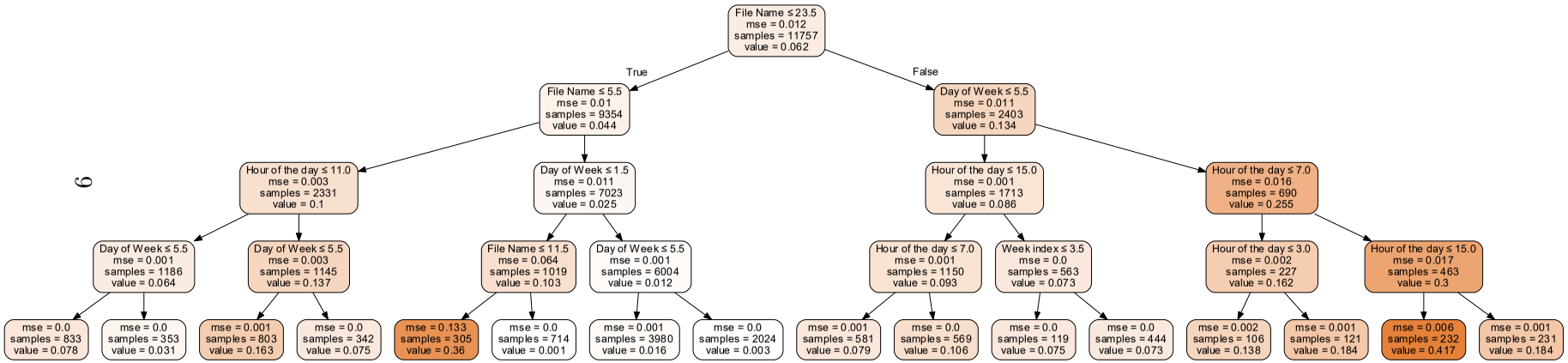
Visualization of one of the decision trees is shown in figure 8. Hyper-parameters for this optimal random forest model are: tree number 85, max feature number 5 and tree depth 4. Since hyper-parameters for this part is different from what we found as optimal, feature importance is re-calculated and shown in table 4. The root node in figure 8 is on file name smaller than 23.5 or not, which is consistent with file name ranked 1st from feature importance calculation in table 4.

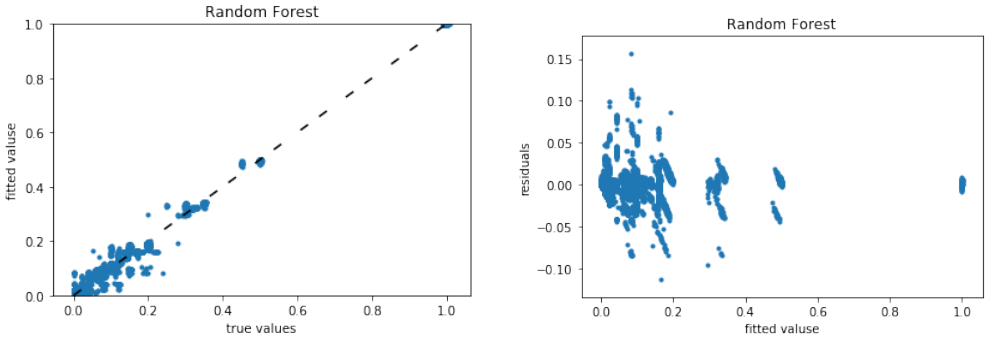
Table 4: feature importance of random forest model with tree depth=4

	file name	day of week	work-flow-ID	hour of day	week number
Importance ranking	1	2	3	4	5
score	0.3864	0.2726	0.189	0.152	0.00

vi. Scatter plot

Using the best random forest parameters found in part iii, scatter points of fitted values vs true values and residual vs fitted values are reported in figure 9.


File Name ≤ 23.5
mse = 0.012
samples = 11757
value = 0.062
True
False
File Name ≤ 5.5
mse = 0.01
samples = 9354
value = 0.044
Day of Week ≤ 5.5
mse = 0.011
samples = 2403
value = 0.134
Hour of the day ≤ 11.0
mse = 0.003
samples = 2331
value = 0.1
Day of Week ≤ 1.5
mse = 0.011
samples = 7023
value = 0.025
Hour of the day ≤ 15.0
mse = 0.001
samples = 1713
value = 0.086
Hour of the day ≤ 7.0
mse = 0.016
samples = 690
value = 0.255
Day of Week ≤ 5.5
mse = 0.001
samples = 1186
value = 0.064
Day of Week ≤ 5.5
mse = 0.003
samples = 1145
value = 0.137
File Name ≤ 11.5
mse = 0.064
samples = 1019
value = 0.103
Day of Week ≤ 5.5
mse = 0.001
samples = 6004
value = 0.012
Hour of the day ≤ 7.0
mse = 0.001
samples = 1150
value = 0.093
Week index ≤ 3.5
mse = 0.0
samples = 563
value = 0.073
Hour of the day ≤ 3.0
mse = 0.002
samples = 227
value = 0.162
Hour of the day ≤ 15.0
mse = 0.017
samples = 463
value = 0.3
mse = 0.0
samples = 833
value = 0.078
mse = 0.0
samples = 353
value = 0.031
mse = 0.001
samples = 803
value = 0.163
mse = 0.0
samples = 342
value = 0.075
mse = 0.133
samples = 305
value = 0.36
mse = 0.0
samples = 714
value = 0.001
mse = 0.001
samples = 3980
value = 0.016
mse = 0.0
samples = 2024
value = 0.003
mse = 0.001
samples = 581
value = 0.079
mse = 0.0
samples = 569
value = 0.106
mse = 0.0
samples = 119
value = 0.075
mse = 0.0
samples = 444
value = 0.073
mse = 0.002
samples = 106
value = 0.138
mse = 0.001
samples = 121
value = 0.184
mse = 0.006
samples = 232
value = 0.417
mse = 0.001
samples = 231
value = 0.184



(a) (b)

Figure 9: Scatter points of (a)fitted values vs true values and (b)residual vs fitted values

2.(c) Neural network

A one-layer fully connected neural network model is used for backup size prediction in this part. Number of hidden units range from 2 to 600, three different kinds of activation functions(sigmoid, tanh and relu) are applied. One-hot encoding is implemented for feature preprocessing to make data size larger. Plot of test-RMSE vs the number of hidden units for each activation function is shown in figure 10. Best number of hidden units is found to be 600 and best activation function is found to be relu. We can easily find out that sigmoid and tanh didn't perform well since they tend to saturate during gradient descent. The required range for hidden units is limited to be 600 in the problem statement, however if we try larger number, testing performance will not increase but might go up due to overfitting. Training and testing RMSE using the best number of hidden units and activation function from 10-fold cross validation are reported in table 5, scatter points of fitted values vs true values and residual vs fitted values are reported in figure 11. Dashed lines in figure 11 indicates the ideal fitting while scatter points show the actual fitting.

Table 5: RMSE for neural network

Training RMSE	Testing RMSE
0.0114	0.0184

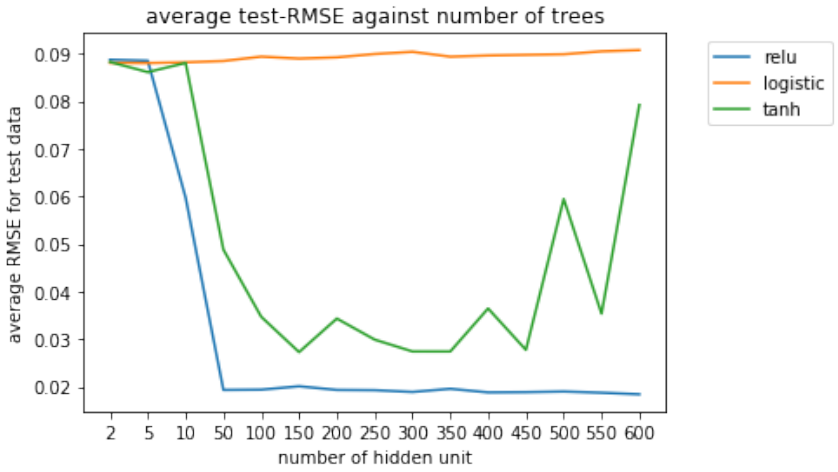
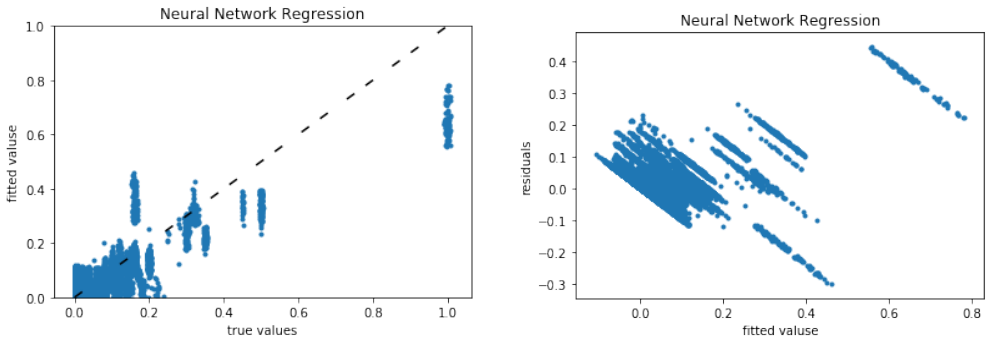


Figure 10



(a) (b)

Figure 11: Scatter points of (a)fitted values vs true values and (b)residual vs fitted values

2.(d) Predict for different workflow separately using linear regression and polynomial features

We firstly split the dataset into five groups by the workflow ID, and then applied linear regression on each group. The RMSE for each groups are listed in table 6. The RMSE here is much smaller than it in non-separately dataset which shown in table 1. If we further define an average test RMSE that weights RMSE of each group by their amounts, the average test RMSE would be $RMSE_{avg} = 0.0633$. The results shows that separating dataset in a smart way indeed improved the fitting, which match our expectation since workflow ID is an important feature

that containing intrinsic information. In addition, this method means we use five different linear regression model to predict the backup size so that better results should be achieved,

Table 6: RMSE for different Workflows with linear regression

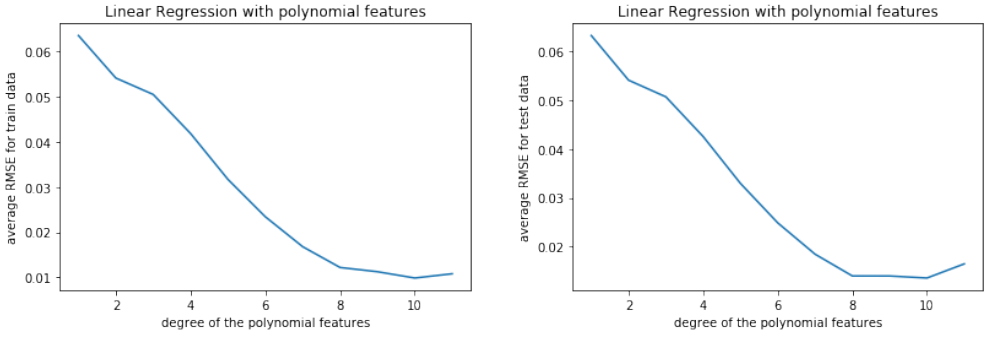
Workflow ID	Train RMSE	Test RMSE
0	0.0358	0.0358
1	0.1488	0.1487
2	0.0429	0.0429
3	0.00724	0.00721
4	0.0859	0.0858

Next we tried to extend our input features by polynomial function and do the parametric study of the degree of the polynomial function. The train and test RMSE are shown in figure 12 and table 7. In fact, when the degree is larger than ten, the time of generating the polynomial features become really long so that our parametric study only did on degree one to eleven. However, we still can observe that the testing RMSE does not decrease as training RMSE but increase at large degree, which indicated overfitting effect occurred. Cross validation here help us to capture the non consistent of training RMSE and testing RMSE and indicate the best degree of polynomial may be nine or ten.

Table 7: RMSE for linear regression with polynomial features

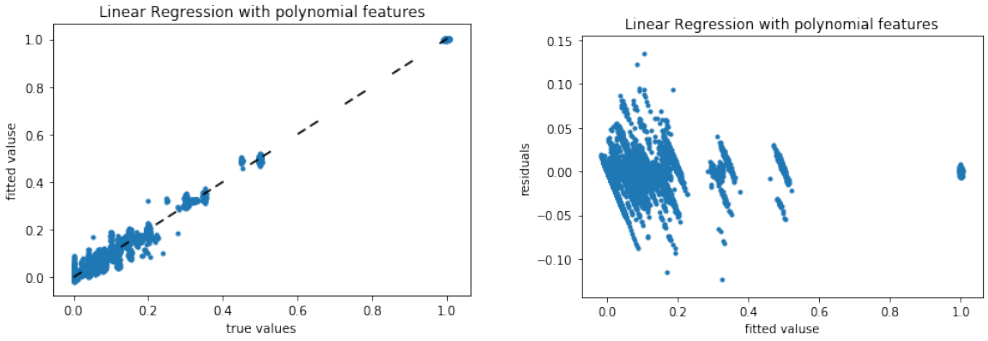
degree	Training RMSE	Testing RMSE
1	0.0635	0.0633
2	0.0541	0.0541
3	0.0505	0.0507
4	0.0418	0.0426
5	0.0317	0.0330
6	0.0234	0.0248
7	0.0168	0.0184
8	0.0122	0.0140
9	0.0113	0.0140
10	0.0099	0.0135
11	0.0108	0.0164

In figure 13, we plotted the prediction of backup size versus true values, and the residual value of them versus true value by choosing the degree of ten from the minimum testing RMSE in table 7. Here, the predictions are more accurate in large backup size. The residuals are also smaller than the predictions fitted by non-separating-workflowID dataset, which shows a significant improvement of polynomial transformation of input features and the splitting dataset.



(a) (b)

Figure 12: Polynomial features (a)train RMSE (b)test RMSE



(a) (b)

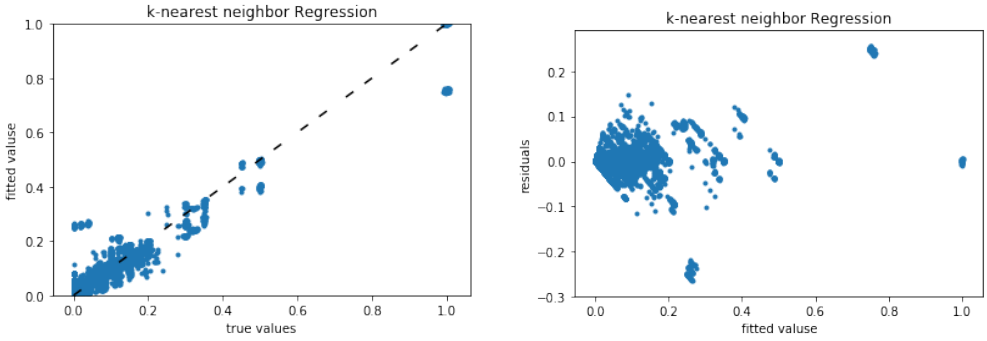
Figure 13: Scatter points of (a)fitted values vs true values and (b)residual vs fitted values

2.(e) K-nearest neighbours

In this part a k-NN regression model is built to predict network backup size. Features with strings are removed, and scalar encoding is used to transform week days to numbers 1-7. Training and testing RMSE from 10-fold cross validation are reported in table 8, scatter points of fitted values vs true values and residual vs fitted values are reported in figure 14. Dashed lines in figure 3 indicates the ideal fitting while scatter points show the actual fitting. From the plot we can see that k-NN model performs better than simple linear regression, but is not as good as random forest and neural network.

Table 8: RMSE for k-NN

Training RMSE	Testing RMSE
0.0279	0.0337



(a) (b)

Figure 14: Scatter points of (a)fitted values vs true values and (b)residual vs fitted values

Summary for dataset 1

We applied 4 different kinds of regression models on the backup dataset. Testing errors summarized from table 1 to table 6 are put together in table 9 for clear observation. The best prediction is achieved by random forest, with slight advantage over neural network. Simple linear regression does not give good prediction results, linear regression with polynomial features improve the prediction by some amount but still not very well. Performance of k-NN is better than linear regression but not as good as the best two models.

Table 9: RMSE for all models

Models	Testing RMSE
simple linear regression	0.1034
random forest with optimal parameter	0.0127
linear regression with polynomial features	0.0633
neural network	0.0184
k-NN	0.0337

Dataset 2: Boston housing dataset

1. Load the dataset

The Boston housing dataset include 14 variables and around 500 data points. We are interested in explaining the target variable MEDV (Median value of owner-occupied homes) using all the other 13 features.

2. Linear regression

We run a linear regression of MEDV on all the other attributes as the features. The OLS penalty function is

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

The p-values of F-statisitics for hypothesis testing $H_0 : \beta_i = 0,\ vs\ \beta_i \neq 0$ of each variable $(i = 1, ..., 13)$ are reported in the following table 10.

Table 10: P-value of significance test

Features	P-value
CRIM	0.001
ZN	0.001
INDUS	0.738
CHAS	0.002
NOX	0.000
RM	0.000
AGE	0.958
DIS	0.000
RAD	0.000
TAX	0.001
PTRATIP	0.000
B	0.001
LSTAT	0.000

From table 10 we know, INDUS and AGE are not significantally different from 0. All the other 11 features are significant for explaining target variable MEDV at the level of 0.1%.

We also performed a 10-folded cross validation for the OLS model. The averaged RMSE for the training data and testing data are reported below:

$$\begin{aligned} \text{RMSE for training data} &= 4.67045 \\ \text{RMSE for testing data} &= 4.79299 \end{aligned}$$

As expected, the RMSE for the testing data is larger than the training data due to the predition error.

For the whole dataset, we plotted of the fitted values against true values in figure 15. Almost all the scattered points in figure 15 are distributed around the 45^0 line which indicates the OLS fitting performance is good in general. But for the highvalue homes ($MEDV \approx 50$), the prediction seems to be biased downwards.

We also plotted the residuals versus fitted values in figure 16. The residuals are almost randomly distributed at around the zero horizontal line except for few outliers which is the consistency with the results from figure 15.

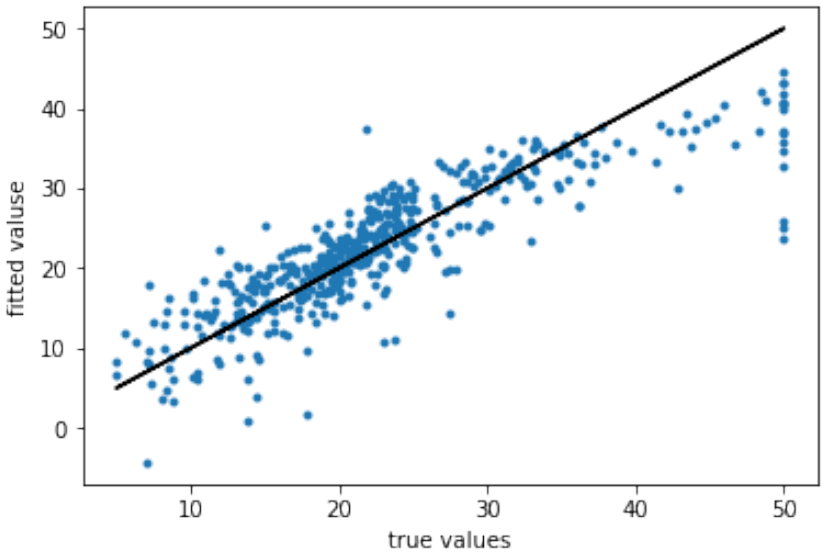


Figure 15: Fitted value against true value

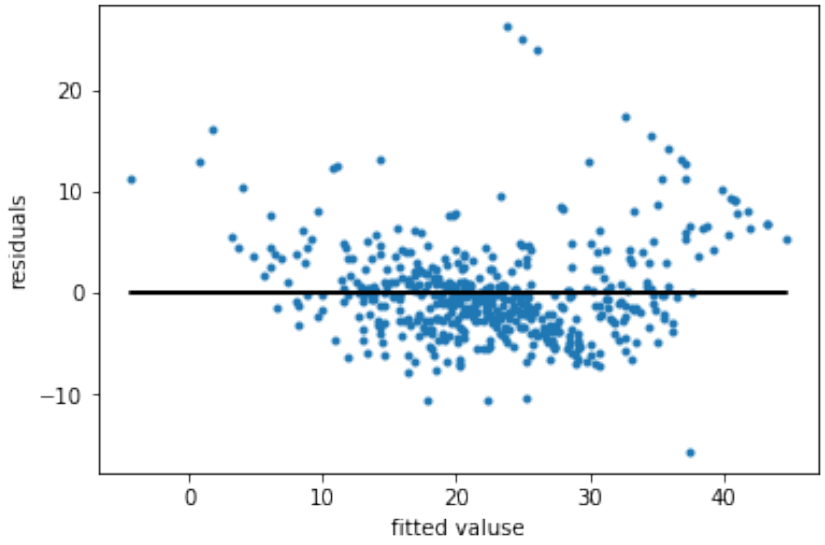


Figure 16: Residuals against fitted value

3. Explore regularization

In this part, we applied three regularization methods to control the possible overfitting problem, including the Ridge, Lasso and Elastic Net Regularizer.

For these regularized models, we choose the regularization parameters via 10-fold cross validation and report them in table 11 along with the best RMSE. We also compare the values of the estimated coefficients for these regularized good models, with the unregularized model. In table 11, we only present the coefficients for CRIM, but the conclusion holds for all the other features.

Table 11: P-value of significance test

	Ridge	LASSO	Elastic Net	Unregularized
Regu. Para.	0.01	0.001	0.001 & 1	0
testing data RMSE	4.797931	4.79299	4.79308	4.79299
Coefficient for CRIM	-0.1079	-0.1078	-0.10177	-0.1080

From table 11 we know, the optimal regularization parameters are small for all the cases. The reason is that we only have 13 features but around 500 data points. Actually, there is almost no overfitting problem in the original unregularized model.

For the values of the coefficients of variable CRIM, the regularized regression models shrink all the coefficient toward 0, thus the absolute value is a bit smaller compared with the unregularized model.

3 Dataset 3: Car insurance dataset

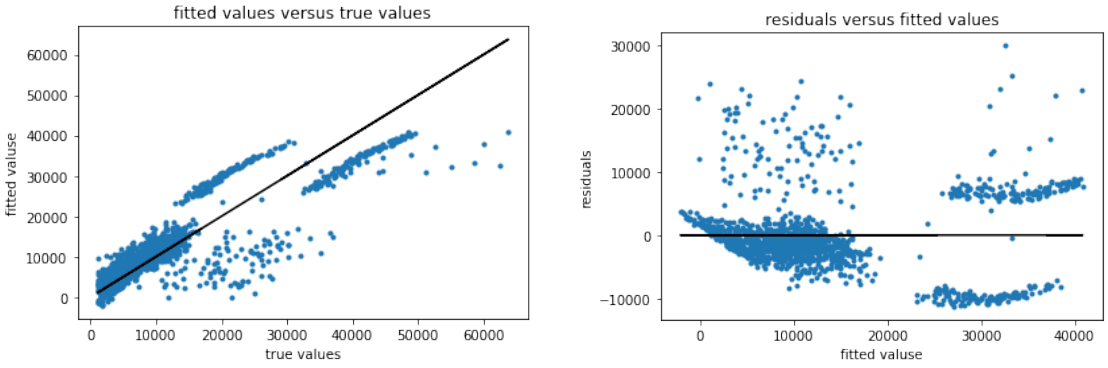
1. Feature Preprocessing

Three feature preprocessing methods are experimented on the car insurance dataset before the linear regression: (a) one-hot encoding on ft4, ft5, ft6; (b) standardization on ft1, ft2, ft3 and one-hot encoding on ft4, ft5, ft6; (c) dividing ft1 into 3 ranges with values 1, 2, 3 and keep

others in (b). The averaged train and test RMSE via 10-fold cross validation are given in table 12. Figures 17 - 19 show the scatter plots of fitted values versus true values and residuals versus fitted values for these 3 preprocessing methods. The results show that different preprocessing leads to very small difference. Preprocessing a and b have almost identical results and diving ft1 into 3 ranges in preprocessing c slightly increases the train and test RMSE.

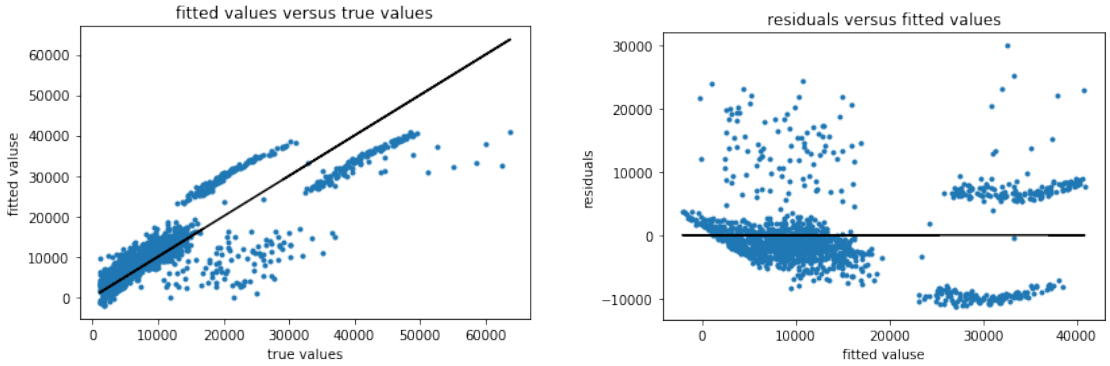
Table 12: RMSE for linear regression with different feature preprocessing

	Training RMSE	Testing RMSE
Preprocessing a	6039.34	6063.64
Preprocessing b	6039.74	6063.72
Preprocessing c	6199.25	6220.99



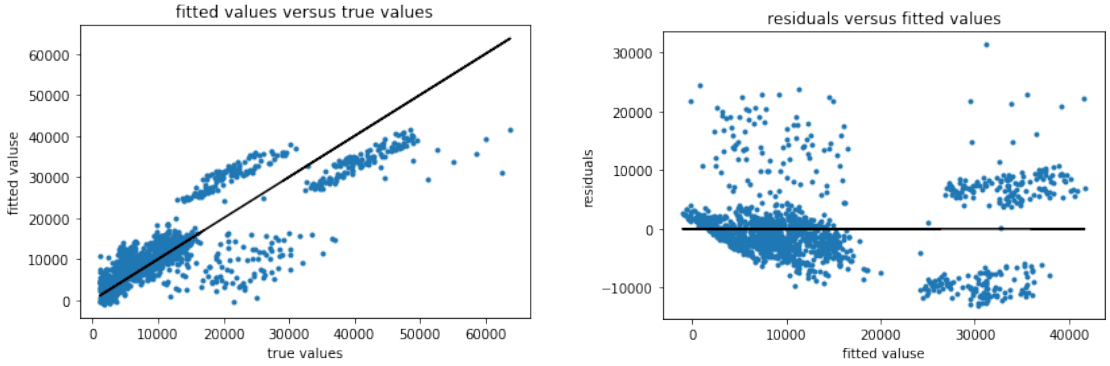
(a) (b)

Figure 17: Scatter points of preprocessing a: one-hot-encoding for ft4,ft5,ft6



(a) (b)

Figure 18: Scatter points of preprocessing b: standardization for ft1,ft2,ft3 and one-hot-encoding for ft4,ft5,ft6



(a) (b)

Figure 19: Scatter points of preprocessing c: dividing ft1; standardization for ft2,ft3; one-hot-encoding for ft4,ft5,ft6

2. Correlation exploration

In this section, the categorical features ft4, ft5, ft6 are converted to numerical features with scalar encoding. Then the f-regression and mutual information regression are performed on 6 features. Figure 20 shows the dependency of the charge against each feature; the normalized values of univariate F-tests statistics, mutual information, and p-value are also given on the top of each plot. The F-test captures the linear dependency and mutual information captures

any form of dependency. P-value measures the probability of observing the distribution in the dataset assuming the null hypothesis is true and is very useful when measuring the importance of a feature. In general, F-test score and mutual information should be large and P-value should be small for a important feature.

Following this criterion, we found ft1 and ft5 are obviously the 2 most important features. The charge has strong linear dependency on ft5 and strong nonlinear dependency on ft1. In addition, the P-value shows that ft2 are also relatively important.

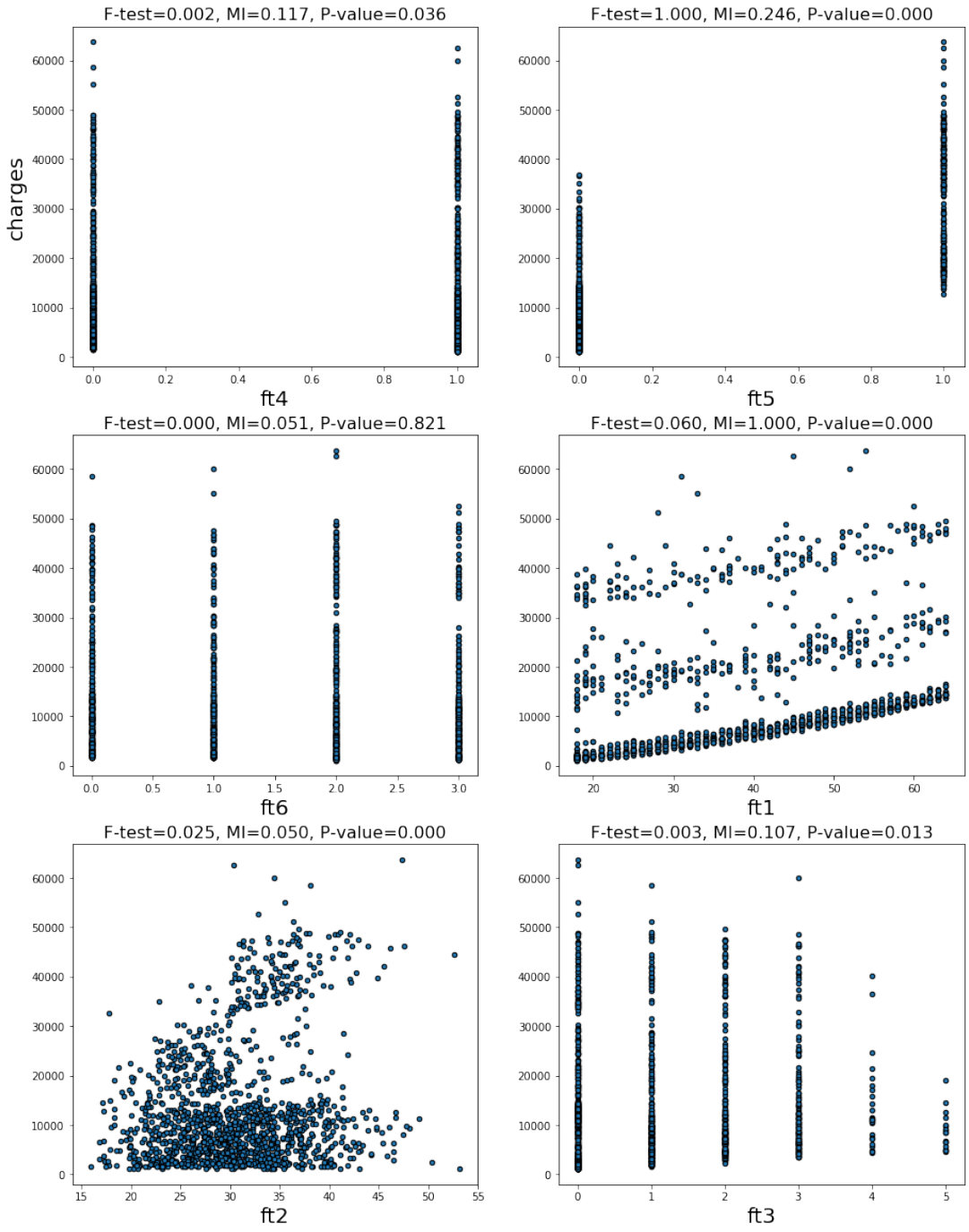


Figure 20: Correlation exploration of features

Figure 21 shows the dependency of charges of ft1 and ft2 and the points are colored based on ft5. When ft1 has large value or when ft5 is "Yes", the charges increases obviously. On the other hand, the charge seems to increase with ft2 only when ft5 is "Yes". When ft5 is "No", ft2

doesn't have obvious effect on the charge. In other words, ft5 seems to act as a switch on ft2.

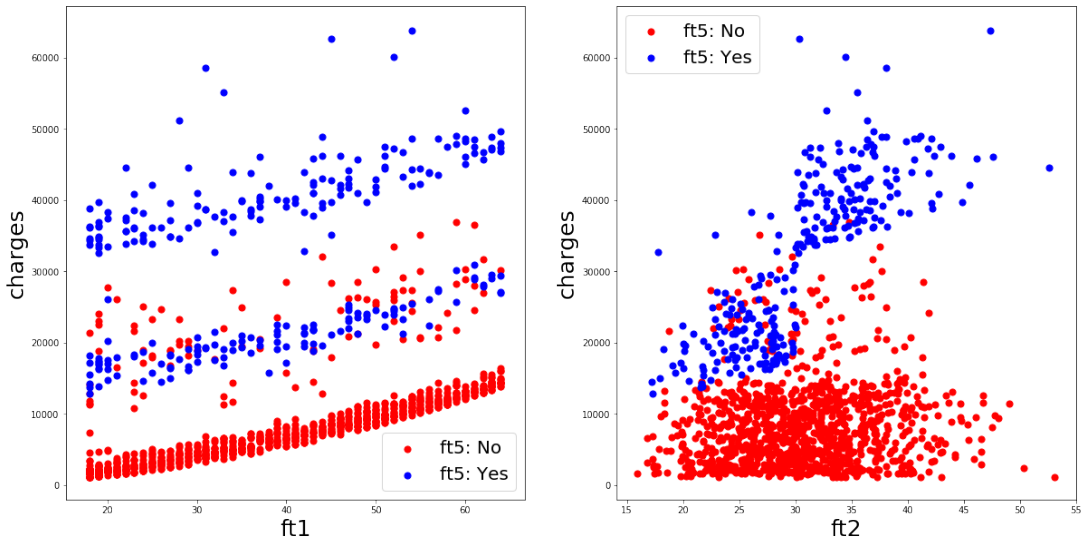
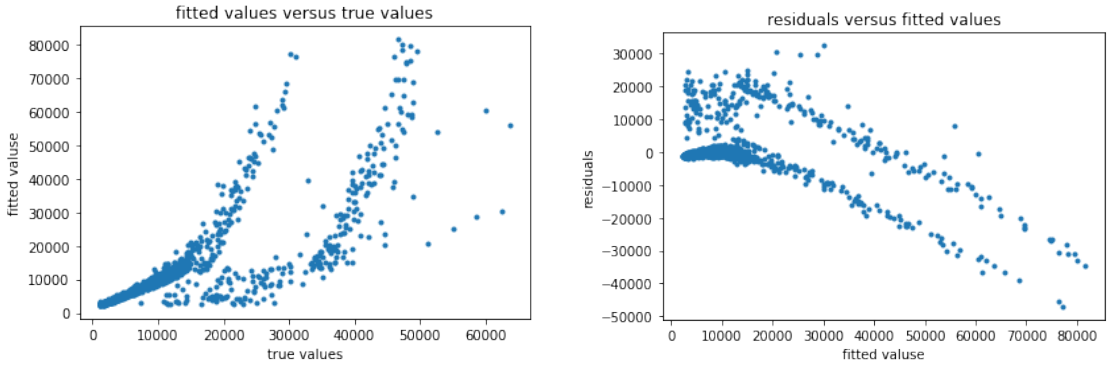


Figure 21: Scatter plot of charges vs ft1 and ft2 (color based on ft5)

3. Modify the target variable

In this section, we performed the linear regression on the logarithm transformed target variable $log(y)$. The feature preprocessing b in part 1 is selected. When calculating the averaged train and test RMSE or visualizing the regression results, the predicted values $log(y)_{predict}$ is transformed to $exp(log(y)_{predict})$. Figure 22 shows the scatter plots of fitted values versus true values and residuals versus fitted values. The fitting is improved for small target values(< 15000), however, the overall accuracy is worse. Table 13 compared the train and test RMSE from the regression on y and $log(y)$, both uses the feature preprocessing in part1(b). It is obvious the RMSE increases when using the modified target variable.



(a) (b)

Figure 22: Scatter points of linear regression on modified target variable

Table 13: RMSE for linear regression on modified target variable

	Training RMSE	Testing RMSE
target: y	6039.74	6063.72
target: $log(y)$	8355.00	8368.93

The correlation exploration is repeated for the linear regression on the new target. Figure 23 shows that ft1 and ft5 are still the 2 most important features, then follows ft2 and ft3. Figure 24 shows the dependency of the new target on ft1, ft2 and ft5. Similar patterns as in figure 21 are observed.

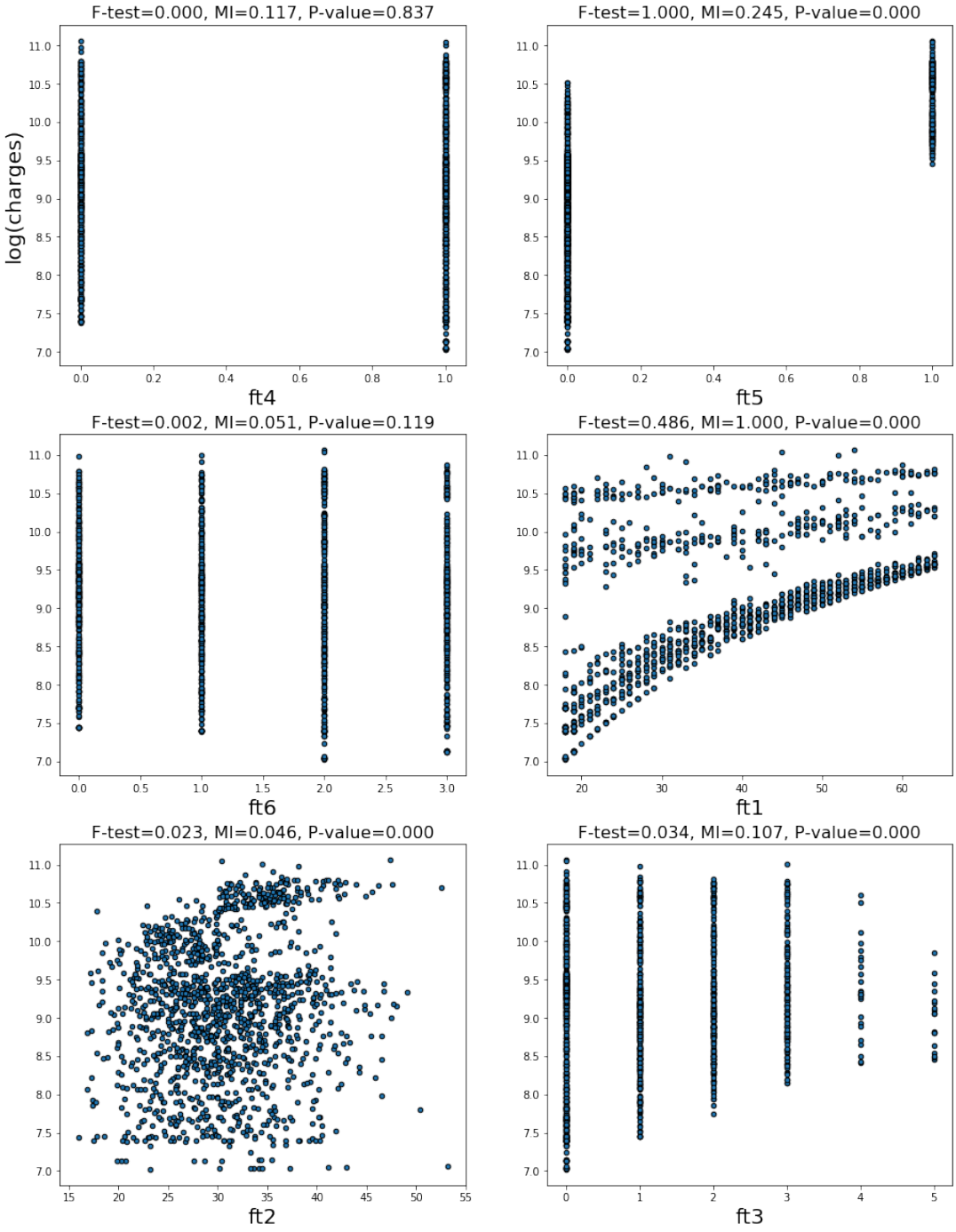


Figure 23: Correlation exploration of features (regression on modified variable)

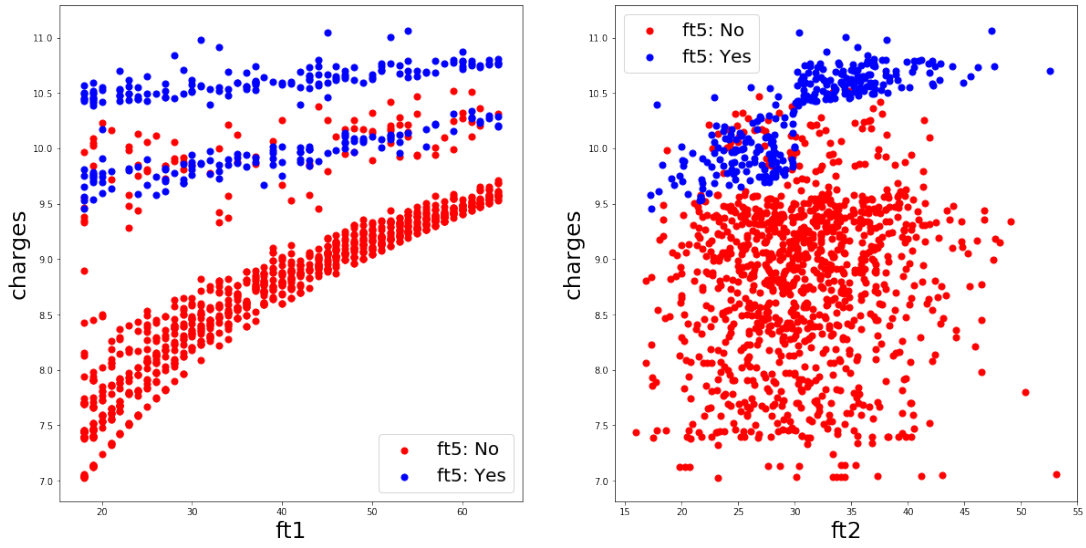
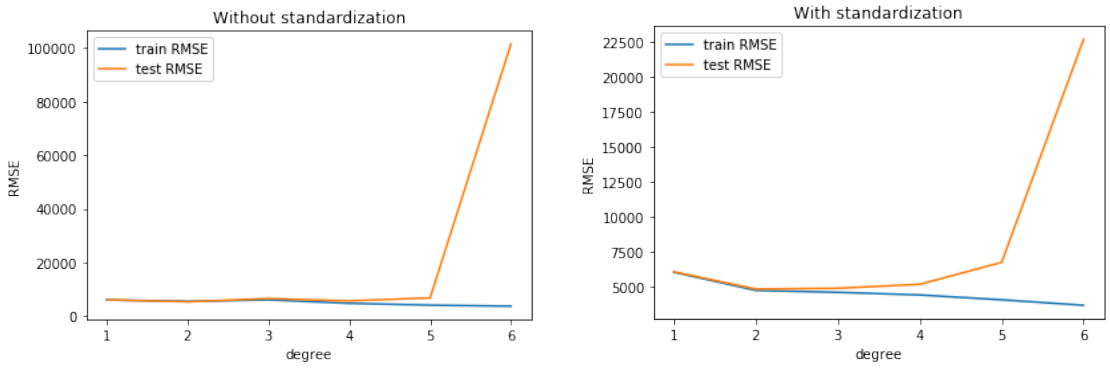


Figure 24: Scatter plot of charges vs ft1 and ft2 (color based on ft5)(regression on modified variable)

4. Bonus questions

(a) Linear regression with polynomial features

The current results show that the linear regression model has poor performance on the car insurance dataset. In this section, we introduced polynomial features with different degrees combined with the feature preprocessing in part 1. The polynomial degree varies from 1 (only adding bias term) to 6. Two different preprocessing method are used: (a) one-hot encoding on ft4,ft5,ft6; (b) standardization on ft1,ft2,ft3 and one-hot encoding on ft4,ft5,ft6. Again, 10-fold cross validation is performed to evaluate the accuracy of polynomial regression.The evolution of train RMSE and test RMSE with polynomial degrees are shown in figure 25. It shows that using standardization on numerical features(ft1,ft2,ft3) before the polynomial transformation leads to better performance, and the best degree is 2. Table 14 and figure 26 show that using the quadratic features can significantly improve the performance of linear regression on the car insurance dataset.

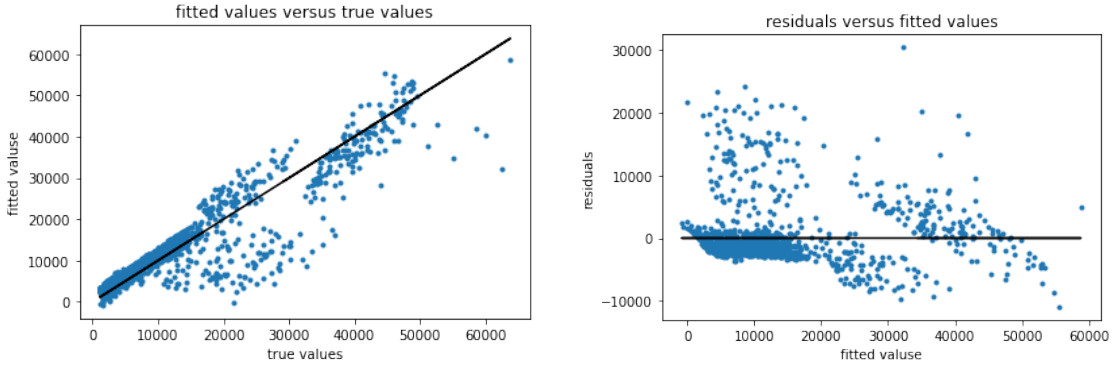


(a) (b)

Figure 25: RMSE vs degree of polynomial features

Table 14: RMSE for linear and polynomial(degree 2) regression with feature preprocessing b

	Training RMSE	Testing RMSE
linear regression	6039.74	6063.72
polynomial regression(degree 2)	4736.32	4832.40



(a) (b)

Figure 26: Scatter points of linear regression with polynomial features (degree = 2)

(b) Other models for regression

In this section, we tried other models on the car insurance dataset: random forest, neural network, and gradient boosting tree. For each class of models, the grid search for several key hyper-parameters are performed. For each individual model, the average test RMSE from 10-fold cross validation are used as the criterion of performance evaluation.

i. Random Forest

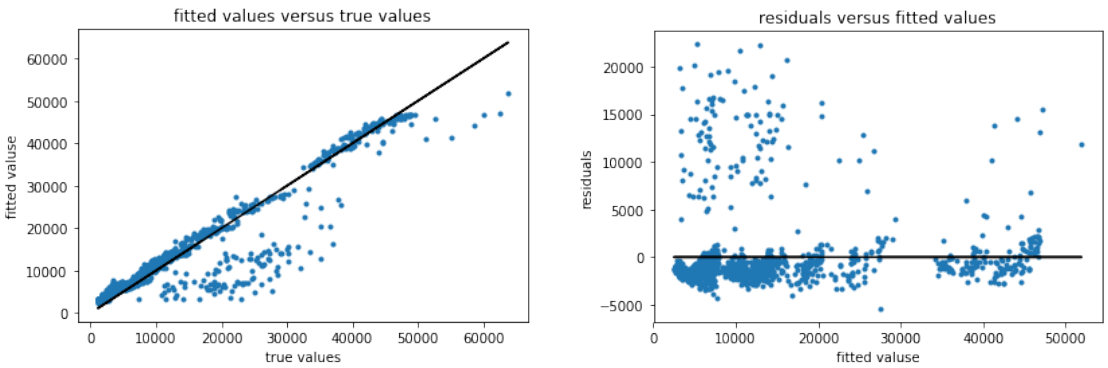
"bootstrap" is always used in random forest, and the hyper-parameters are:

- Depth of each tree: [2, 3, 4, 5, 6]
- Maximum number of features: [2, 3, 4, 5, 6]
- Number of trees: [10, 20, 50, 100, 150, 200]

Table 15 shows the top 10 best random forests based on testing RMSE from the grid search, and the scatter plot of the best one is presented in figure 27. Compared with all linear regression based models(including polynomial regression), the performance is significantly improved according to both the RMSE values and the scatter plots.

Table 15: Top 10 best models from hyper-parameter search for random forest

Rank	Depth of each tree	Max. number of features	Number of trees	Testing RMSE	Training RMSE
1	5	4	150	4494.40	4075.75
2	5	4	200	4495.70	4074.84
3	5	4	100	4502.32	4078.17
4	5	5	150	4510.08	4038.95
5	5	5	200	4513.90	4036.48
6	5	5	100	4514.27	4039.12
7	5	4	50	4516.67	4087.74
8	4	5	200	4518.72	4264.01
9	5	5	20	4519.80	4053.02
10	6	4	200	4520.70	3778.48



(a) (b)

Figure 27: Scatter points of best random forest

ii. Neural Network

"Adam" algorithm is used to train the neural network and the maximum iterations are 200. The hyper-parameters for neural network are:

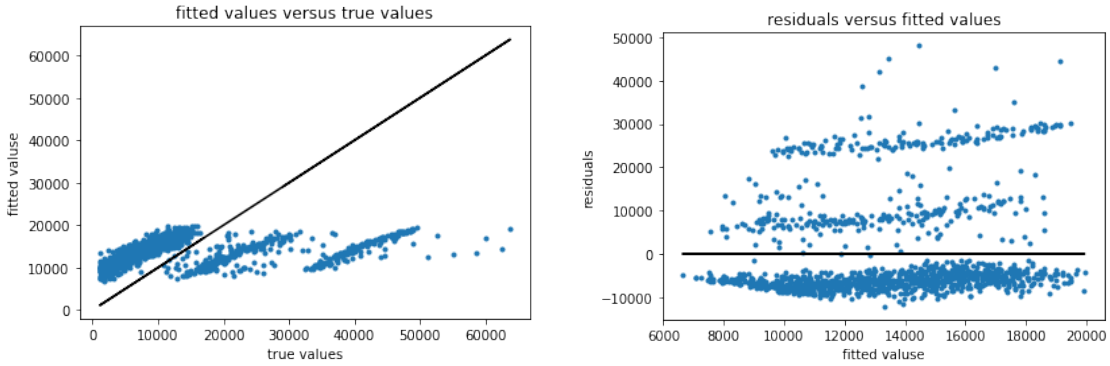
- Number of hidden units: [10, 50, 100, 150, 200]

- Activation function: [logistic, tanh, relu]
- L_2 penalty parameter: [1e-5, 3e-5, 1e-4, 3e-4, 1e-3]

Table 16 shows the top 10 best neural networks based on testing RMSE from the grid search, and the scatter plot of the best one is presented in figure 28. The neural network has poor performance on this dataset. This is because only one hidden layer is applied so the network is way too shallow to fit a nonlinear function. A deeper neural network with more hidden layers should improve the fitting significantly.

Table 16: Top 10 best models from hyper-parameter search for neural network

Rank	Number of hidden units	Activation function	L_2 penalty parameter	Testing RMSE	Training RMSE
1	200	relu	0.0001	11360.75	11351.01
2	200	relu	0.0003	11362.02	11352.00
3	200	relu	0.001	11362.20	11351.81
4	200	relu	3.00E-05	11363.73	11353.57
5	200	relu	1.00E-05	11367.88	11358.46
6	150	relu	0.0001	11414.67	11409.82
7	150	relu	0.0003	11415.97	11411.61
8	150	relu	0.001	11417.19	11411.96
9	150	relu	3.00E-05	11418.63	11414.30
10	150	relu	1.00E-05	11421.11	11416.53



(a) (b)

Figure 28: Scatter points of best neural network

iii. Gradient Boosting Tree

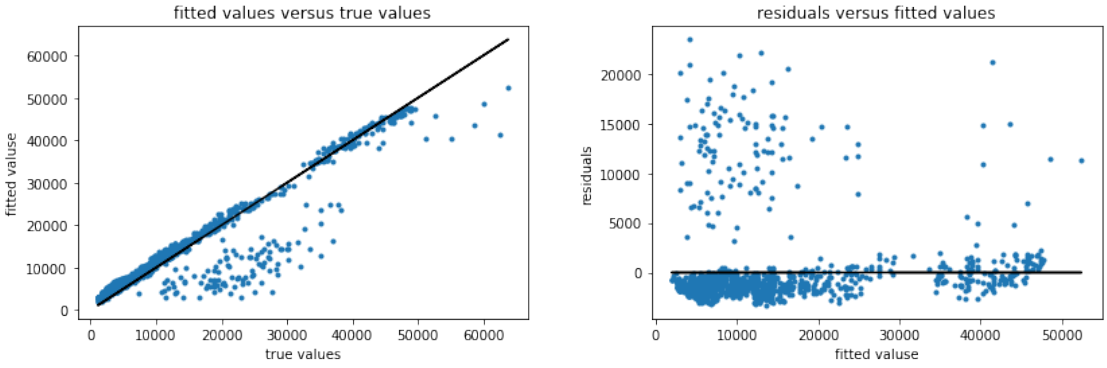
The hyper-parameters for gradient boosting tree are:

- Learning rate: [1e-2, 3e-2, 1e-1, 3e-1, 1]
- Depth of each tree: [2, 3, 4, 5, 6]
- Maximum number of features: [2, 3, 4, 5, 6]
- Number of trees: [20, 50, 100, 150, 200]

Table 17 shows the top 10 best gradient boosting trees based on testing RMSE from the grid search, and the scatter plot of the best one is presented in figure 29. The gradient boosting tree achieves slightly better performance according to the testing RMSE than the random forest. Both random forest and gradient boosting tree are ensemble methods. The difference is that each tree in random forest is trained independently and only a faction of samples is used through bootstrap. Therefore, random forest will never overfit. Gradient boosting tree, on the other hand, is trained on the whole dataset, and each tree is built sequentially based on the residual in the last step. Therefore, it can overfit the dataset and we need to adjust the learning rate and limit the total number of trees to prevent the overfitting.

Table 17: Top 10 best models from hyper-parameter search for gradient boosting tree

Rank	Learning rate	Depth of each tree	Max. number of features	Number of trees	Testing RMSE	Training RMSE
1	0.03	3	5	150	4474.39	4132.84
2	0.1	3	5	50	4479.75	4086.98
3	0.03	3	5	200	4483.11	4029.03
4	0.1	3	4	50	4483.81	4105.69
5	0.1	3	6	50	4486.23	4076.65
6	0.03	3	6	150	4486.38	4123.85
7	0.03	3	6	200	4492.18	4013.65
8	0.03	3	4	150	4495.04	4157.37
9	0.03	3	4	200	4497.10	4056.95
10	0.3	3	6	20	4506.11	3994.75



(a) (b)

Figure 29: Scatter points of best gradient boosting tree

Summary for dataset 3

We applied different feature preprocessing methods and different models on the car insurance dataset. The preprocessing alone doesn't help improving the performance of linear regression. Polynomial features introduces non-linearity into the linear regression and thus can significantly

improve fitting. A one-layer neural network barely works on this dataset. The random forest and gradient boosting tree have the best performance on this dataset. Table 18 summaries the testing RMSE of different models on this dataset.

Table 18: RMSE for all models

Models	Testing RMSE
simple linear regression	6063.72
linear regression on new target	8368.93
linear regression with polynomial features	4832.40
random forest	4494.40
neural network	11360.75
gradient boosting tree	4474.39