EE 219 Project 4
Regression Analysis
Winter 2018

Jui Chang
Wenyang Zhu
Xiaohan Wang
Yang Tang

Introduction

In this project, we made regression analysis on Network Backup Dataset, which contains simulated traffic data on a backup system over a network. Our task is to make prediction of the backup size based on the attributes utilizing regression analysis.

Regression analysis is a statistical process in data mining for estimating the relationships among variables. In this project, we studied and used four prediction models, including Linear Regression Model, Random Forest, Neural Network Regression Model and Polynomial Regression Model, to predict the backup size and used cross-validation to evaluate the performances.

Question 1 Load the dataset

After loading the dataset, first we need to encode the columns of "day of week", "work flow" and "file name" into integers. Then, we calculate the included weeks and days and group the relevant data into a data frame. Finally, we can plot the backup size versus days figures. Below are our results:

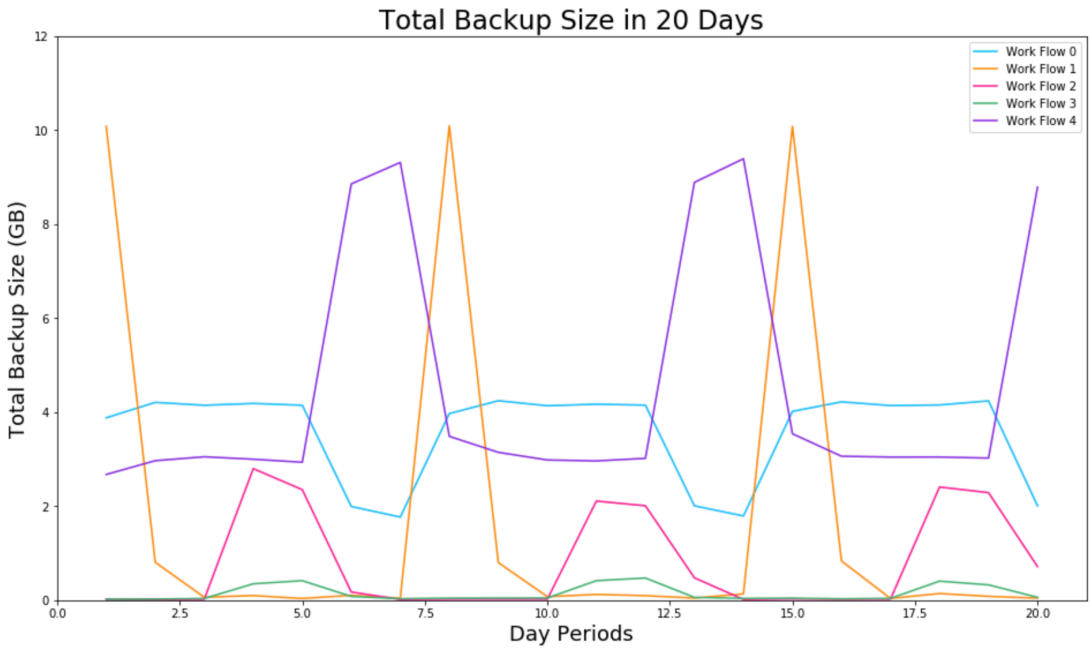


Figure 1. total backup size v.s. 20 days

From figure 1, it's easily observed that there is a periodic repeating pattern. For each work flow, the wave crest and through of the total size appear alternately and a complete period is about 7 days. For example, for work flow 0, on the first day, the total size is about 3.93 GB, and then the sizes are 4.18 GB, 4.15 GB, 4.16 GB, 4.1 GB, 2 GB and 1.9 GB from the second day to the 7^{th} day. And on the 8^{th} day, the total size is about 4 GB again, and the change pattern in the second week is similar to that of the first week. Other work flows act the same trend.

To be specific, if we observe the 5 work flows respectively, we can find their own change pattern:

1) Work Flow 0: the crests of backup size appear on the first to 5^{th} days in each week, and on weekend, the backup sizes are relatively small. So we may deduce that the work flow 0 represents the roads in some business district where there are larger traffic flow on weekdays, just like CBD.
2) Work Flow 1: the crest of backup size appears only on Monday and on other days, the backup sizes are very small.
3) Work Flow 2: the large sizes appear on Thursday and Friday, and on other days, the data flows are very small.

4) Work Flow 3: in this flow, the traffic data is relatively much smaller than other flows. The backup size only increases a little on Thursday and Friday, and on other days, the size is too small to count.
5) Work Flow 4: the trend of this flow is completely opposite to work flow 0. The crests appear on weekend and the sizes are relative small and static on weekdays. This flow may represent some shopping plaza or entertainment area.

Figure 1 only shows 20 days data, and it may not be that representative. So we extend the day period to 105 days and get the figure 2 below. From this figure, we can see that the change pattern in 105 days is the same as that in 20 days, which verifies the rules we find above.

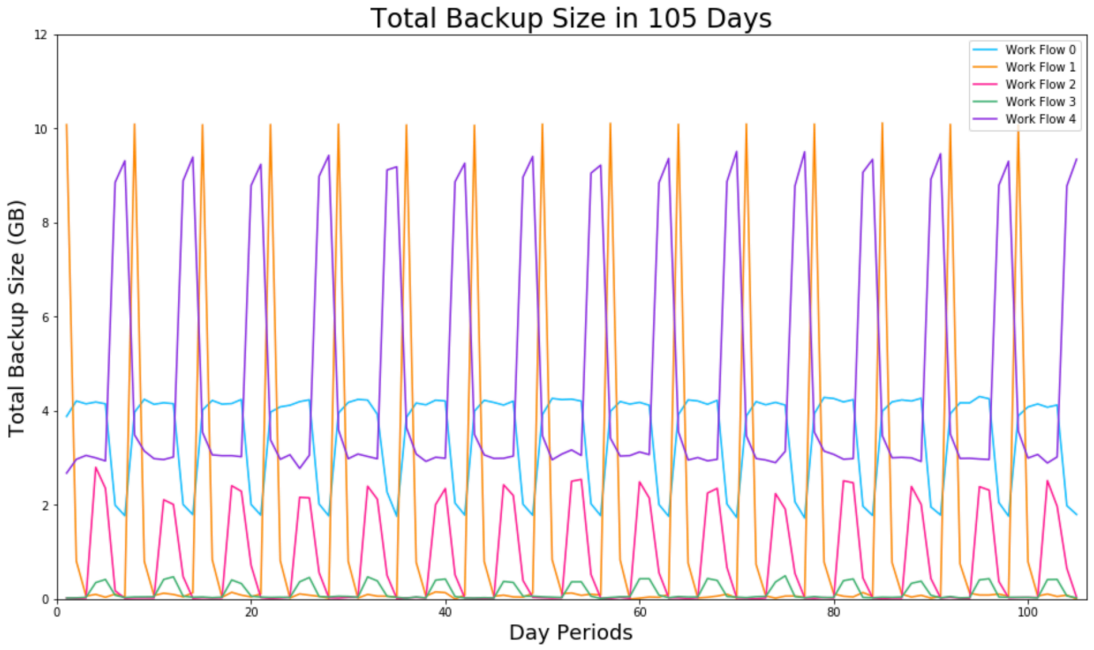


Figure 2. total backup size v.s. 105 days

Question 2 Predict

There are seven attributes in the original traffic dataset, and our task is to predict the backup size based on the training model. The “backup size” is what we need to predict, so we can exclude it. And for “backup time”, it’s directly correlated to backup size and it’s actually the result attribute, so we can also exclude it. Therefore, in this part, we use five features (day of week, hour of the day, work flow number, file type and week number) as candidate features to predict backup size.

In the following part, we trained three different models (Linear Regression Model, Random Forest, and Neural Network Regression Model) to fit the data and plot the fitted figures, and then use 10-fold cross-validation to evaluate the model performance (by training RMSE and test RMSE), and then try other methods including encoding combinations, feature selections, standardization and regularization to improve the regression models.

Question 2(a) Linear regression model

We use ordinary least square as the penalty function.

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

where the minimization is on the coefficient vector β.

Question 2(a) – i Scalar Encoding Only

Some of the five candidate features are string values, so we need to convert them into numerical values to calculate for regression models. In this part, we use straightforward scalar encoding schema to convert each categorical feature into one dimensional numerical values:

- Monday – 1, Tuesday – 2, Wednesday – 3, Thursday – 4, Friday – 5, Saturday – 6, Sunday – 7
- Work flow IDs are mapped to 0 to 4 accordingly
- File names are mapped to 0 to 29 accordingly

Then we use the dataset to train and fit a basic linear regression model and use 10-fold cross validation to evaluate the performance and got the corresponding training and test RMSE as below:

Table 1. training RMSE & test RMSE for scalar encoding

fold	train_rmse	test_rmse
0	0.103243	0.106718
1	0.103967	0.100185
2	0.103226	0.106850
3	0.103946	0.100367
4	0.103195	0.107116
5	0.103938	0.100445
6	0.103203	0.107050
7	0.103936	0.100467
8	0.103201	0.107074
9	0.103992	0.099947

From the table, we can see that test RMSEs are similar to training RMSEs, and they are distributed at around 0.1.

Next, we picked up the best parameter of the linear regression model among those 10 folds and used it to fit the whole dataset. Finally, we plotted the fitted values v.s. true values figure and residuals v.s. fitted values figure as below:

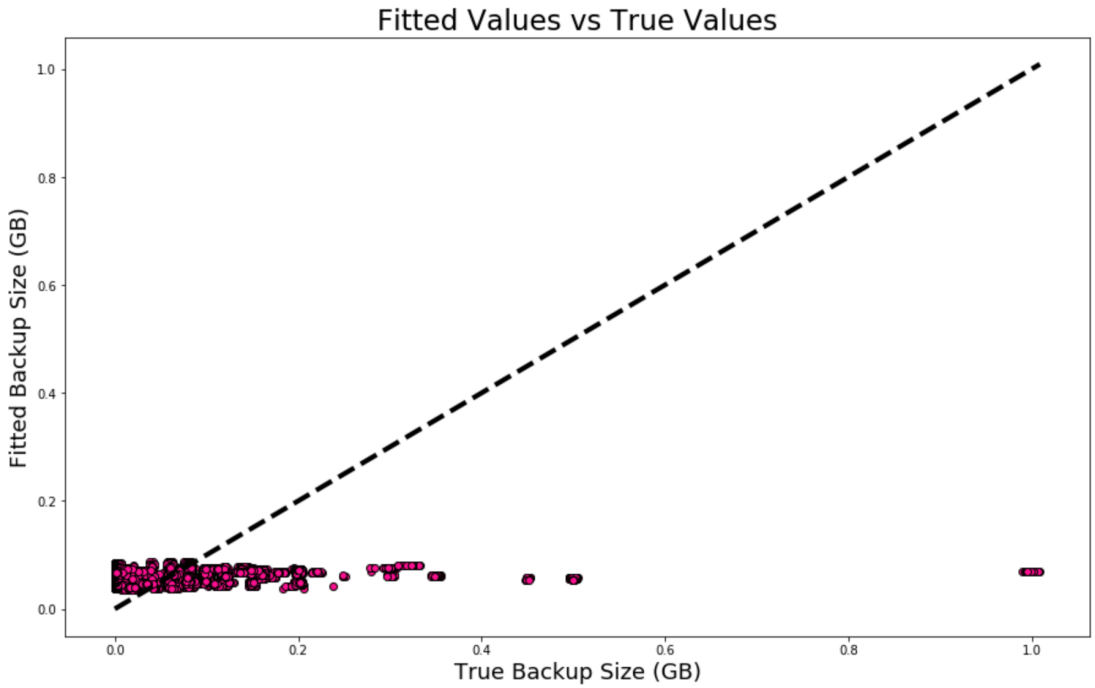


Figure 3. fitted values v.s. true values for scalar encoding

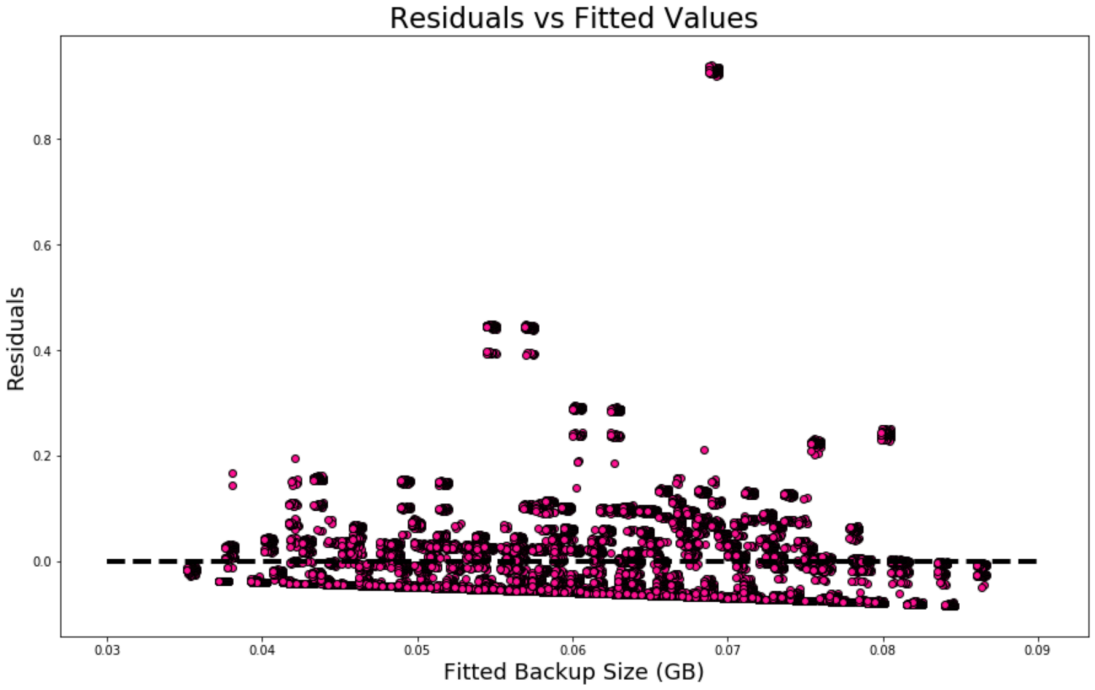


Figure 4. residuals v.s. fitted values for scalar encoding

From the above figures, we can find that the linear regression model using scalar encoding approach is not ideal. The fitted values have large difference to the true values, and they are distributed from 0.03 GB to 0.09 GB, compared to the true values from 0

GB to 1 GB. Besides, the residuals scattered around 0, and the largest residual is about 0.9 GB which is much larger than the fitted value (different for an order of magnitude).

Question 2(a) – ii Data Preprocessing

In this part, we standardize all five features and the real sizes into standard normally distributed data. Below are training RMSE and test RMSE, and two plots:

Table 2. training RMSE & test RMSE (standardized)

fold	train_rmse	test_rmse
0	0.990916	1.024267
1	0.997861	0.961560
2	0.990749	1.025531
3	0.997666	0.963311
4	0.990454	1.028085
5	0.997588	0.964062
6	0.990527	1.027456
7	0.997569	0.964267
8	0.990511	1.027685
9	0.998099	0.959281

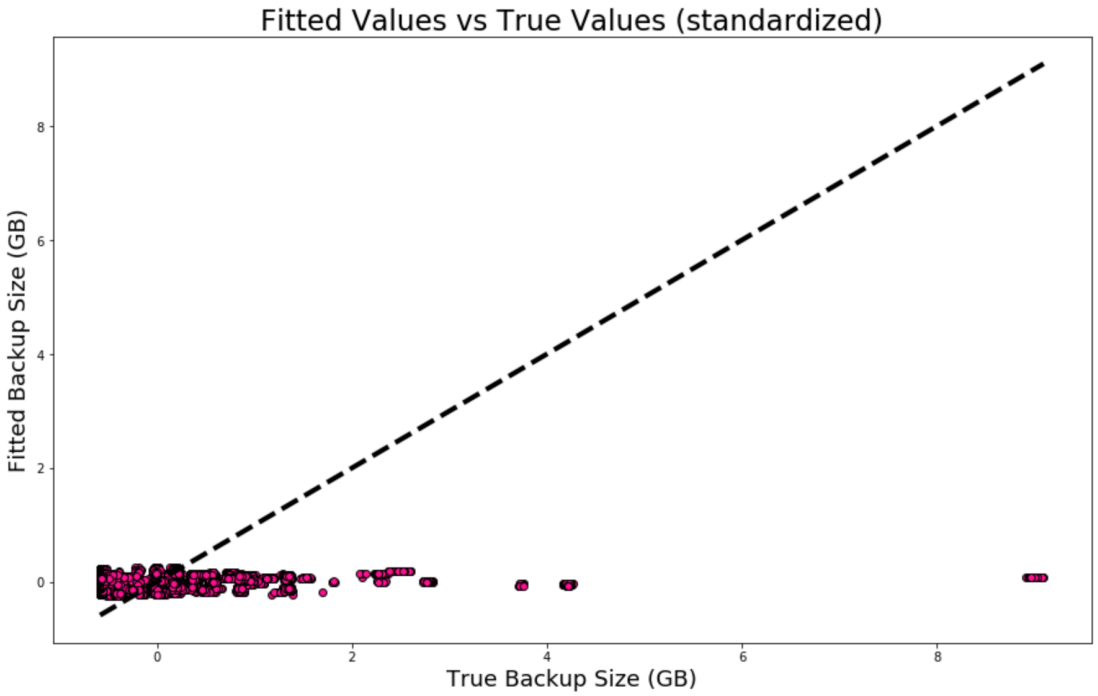


Figure 5. fitted values v.s. true values (standardized)

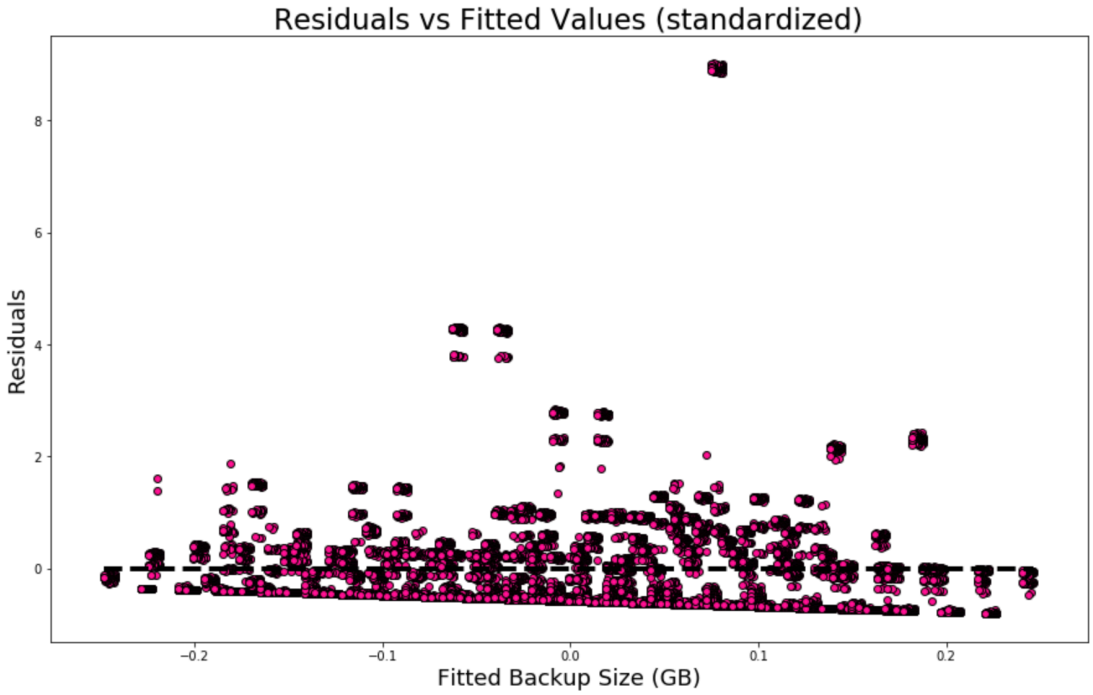


Figure 6. residuals v.s. fitted values (standardized)

From the RMSE results, we can find that the training RMSE and test RMSE after standardization are average 0.1 smaller than those without standardization and it's not much different. While for two plots above, the distributed shape is similar to those

without standardization, however, the scale of the data changed a lot. First, before standardization, the fitted sizes are distributed in the range from 0.03 to 0.09, and the true size distributes from 0 to 1. After standardization, the fitted sizes centered around 0, from -0.25 to 0.25, and the true sizes standardized from about -1 to 9. In other words, they both scale 10 times. Second, the residuals also scale 10 times as the fitted data and true data changed. After standardization, the residuals are in the range of -0.7 to 9, which are about 10 times to those before standardization for about -0.07 to 0.9.

Question 2(a) – iii Feature Selection

Intuitively, the feature "week" is not very important to the backup size since from the figures in Question 1, we can find an apparent repeating pattern in 7 days period. So in this part, we will use two more accurate method f_regression and mutual_info_regression to select three important features and then train a new linear regression model.

1) Use f_regression measure

   Integrating f_regression measure, we select three most important features, and their indices corresponding to the features are:

   1 – day of week

   2 – start time

   3 – work flow ID

   Use those three most important features to train a new linear model, and we can get the training RMSE and test RMSE as below:

Table 3. training RMSE & test RMSE (standardized + f_regression)

fold	train_rmse	test_rmse
0	0.990921	1.024132
1	0.997869	0.961420
2	0.990750	1.025524
3	0.997668	0.963290
4	0.990455	1.028083
5	0.997589	0.964064
6	0.990529	1.027430
7	0.997570	0.964248
8	0.990517	1.027589
9	0.998100	0.959228

The fitted values vs true values and residuals vs fitted values plots are as below:

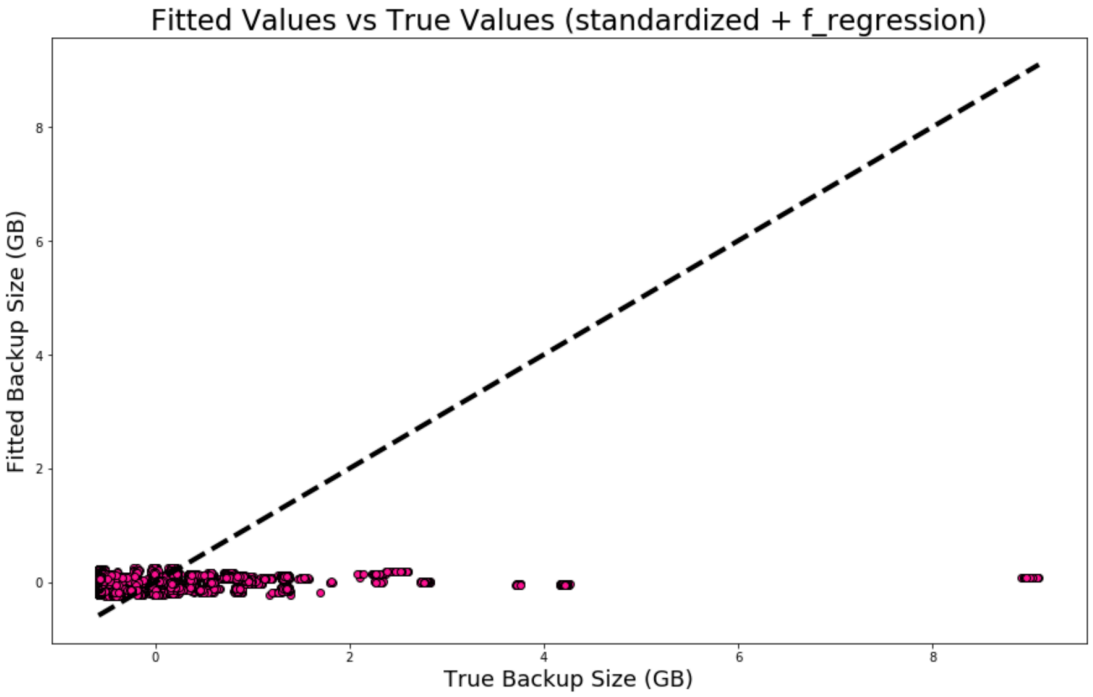


Figure 7. fitted values v.s. true values (standardized + f_regression)

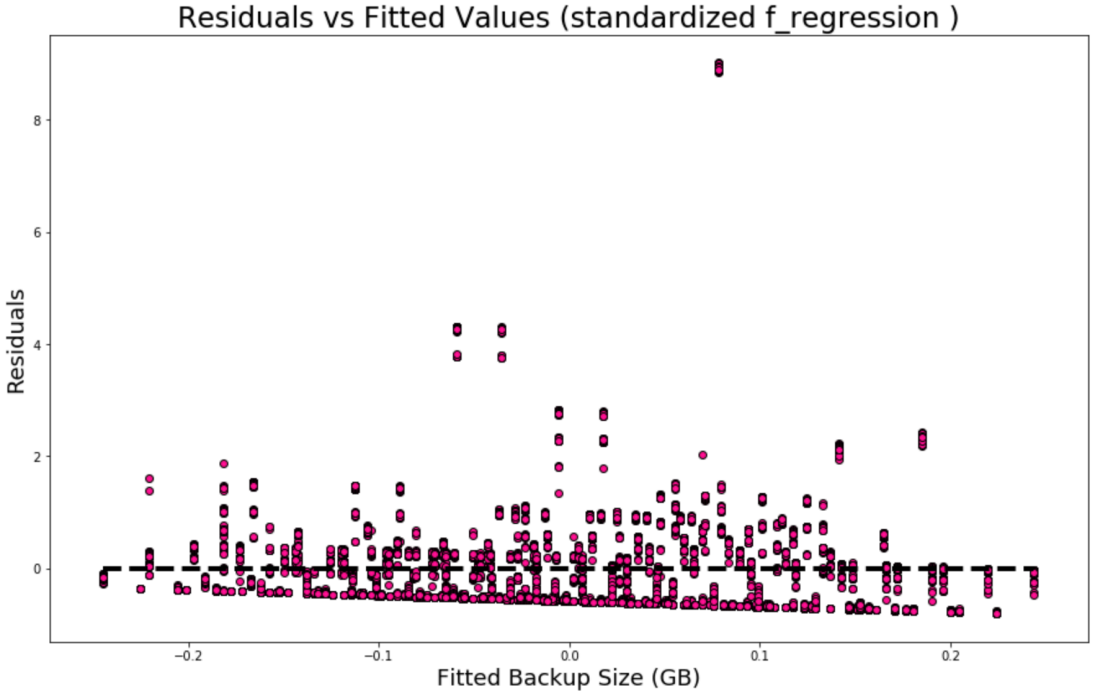


Figure 8. residuals v.s. fitted values (standardized + f_regression)

From the figures and table above, we can see that the performance didn't improve much compared to that without using f_regression. The fitted values distributed in the same range as question ii with standardization, and the residuals are also similar. This indicates that the three features f_regression measure selected are the real true significant and weighted features in the process of training the model, and the dropped features "week" and "file name" are not dominating factors to backup size.

2) Use mutual_info_regression measure

Integrating mutual_info_regression measure, we select three most important features, and their indices corresponding to the features are:

2 – start time

3 – work flow ID

4 – file name

Use those three most important features to train a new linear model, and we can get the training RMSE and test RMSE as below:

Table 4. training RMSE & test RMSE (standardized + mutual_info_regression)

fold	train_rmse	test_rmse
0	0.991888	1.025784
1	0.999000	0.961580
2	0.991707	1.027262
3	0.998791	0.963534
4	0.991408	1.029864
5	0.998724	0.964186
6	0.991500	1.029055
7	0.998700	0.964421
8	0.991464	1.029416
9	0.999230	0.959415

The fitted values vs true values and residuals vs fitted values plots are as below:

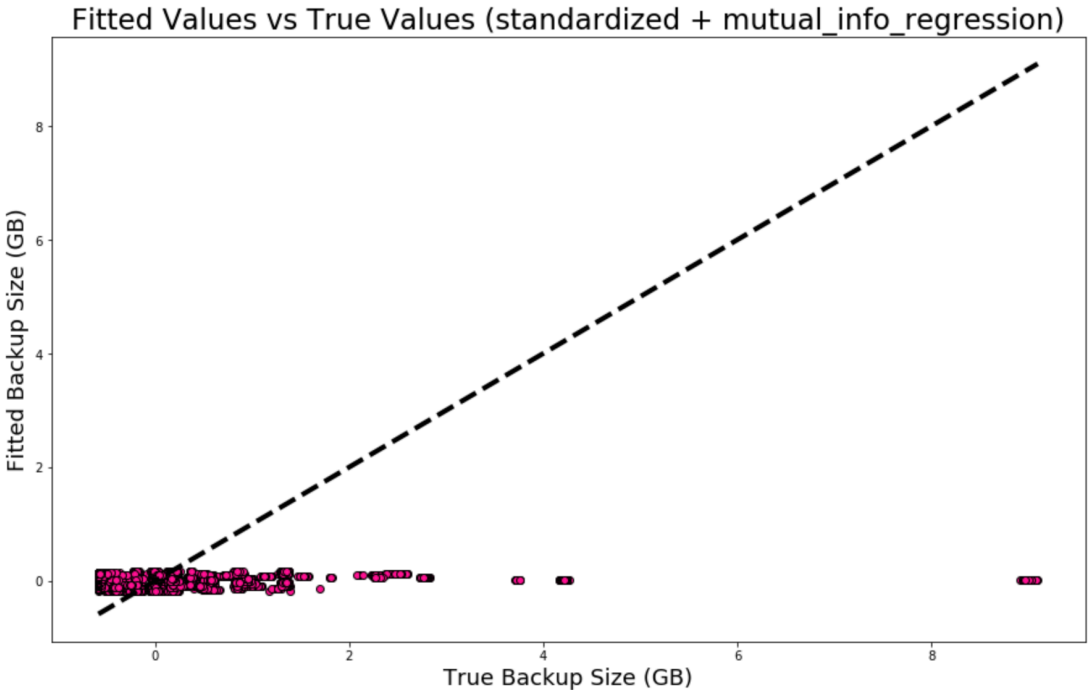


Figure 9. fitted values v.s. true values (standardized + mutual_info_regression)

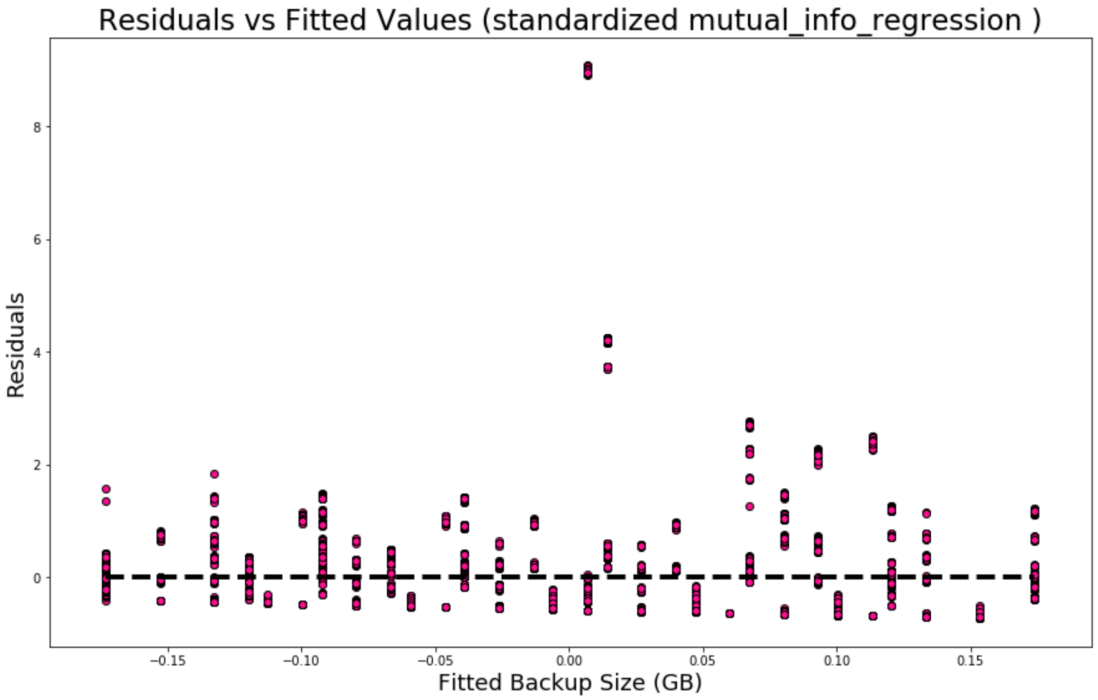


Figure 10. residuals v.s. fitted values (standardized + mutual_info_regression)

From the figures and table above, the performance after using mutual_info_regression is not improved much. The training RMSE and test RMSE are similar to that using f_regression and that without feature selection. However, after using mutual_info_regression, the distributed range of fitted values decreased to -0.15 to 0.15 compared to that using f_regression ranged from -0.2 to 0.2. Besides, in the figure of residuals, the distributed is more sparse and clear. The whole dataset spread into 30 group (vertical bar) which corresponding to 30 files. This indicates that among those three features, "file name" is more weighted.

With the observation of the results of those two feature selection measures, we can deduce that except the first feature "week", the other four features are the dominating features for the linear regression model. Among those four features, "backup start time" and "work flow" are the two most primary and important features.

Question 2(a) – iv Feature Encoding

In the process of linear regression, we use 5 features to train the model. In the previous

part, we only use scalar encoding method to convert those 5 features into numerical values as non-negative integers. In this part, we add One-Hot-Encoding method to those 5 features and calculate the performances on those 32 combinations. To efficiently record the performances, we use bit operations here. First, we initialize a 5-bit number to represent 32 combinations (reverse bit order for feature order). “0” represents using scalar encoding method, and “1” represents using one-hot-encoding method. For example, “combination 9 = 01001” represents “week” – one-hot-encoding, “day of week” – scalar encoding, “start time” – scalar encoding, “work flow” – one-hot-encoding, “file name” – scalar encoding.

The mean training RMSE and mean test RMSE changed as below:

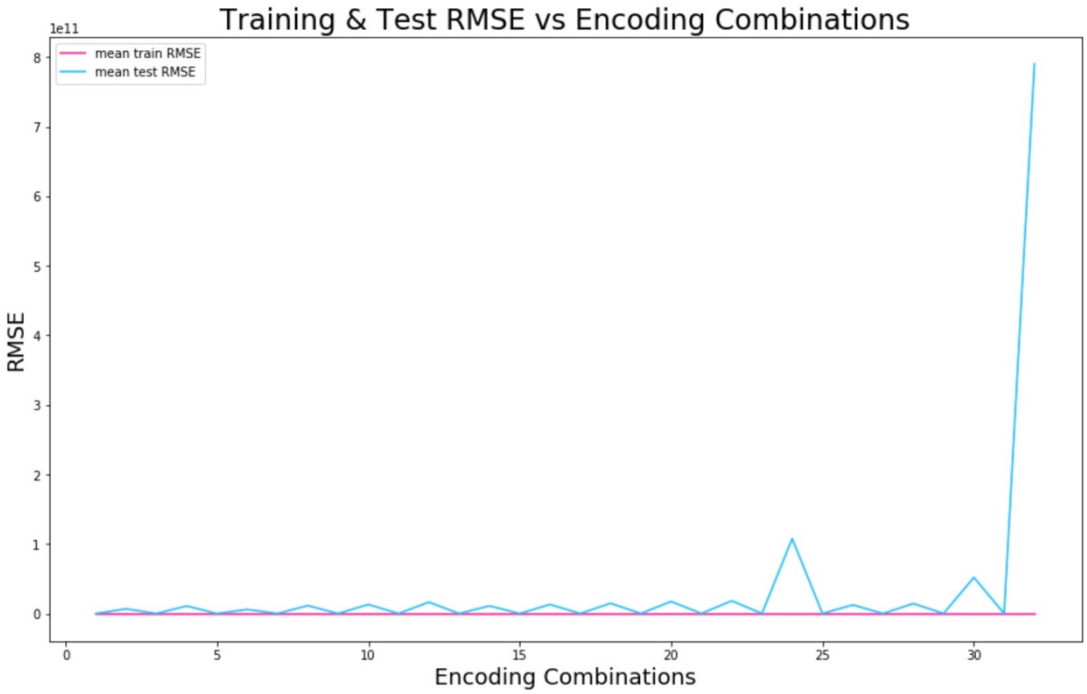


Figure 11. training & test RMSE v.s. encoding combinations

From this figure, we can find that the performances on those combinations vary much. The training RMSEs are almost static from 0.088 to 0.1, while the test RMSEs for even combinations are very close to that of training RMSEs, and for odd combinations, the test RMSEs are very large.

From the mean training RMSEs and test RMSEs of those 32 combinations, we can find the best three combinations with better performances:

```
The best combination is: 22
The best mean_train_rmse is: 0.0883364063497, and the best mean_test_rmse is: 0.0883701294703
The second best combination is: 14
The second best mean_train_rmse is: 0.0883385487681, and the second best mean_test_rmse is: 0.0883706233827
The third best combination is: 30
The third best mean_train_rmse is: 0.0883369846271, and the third best mean_test_rmse is: 0.0883742439185
```

Figure 12. best three combinations

These three combinations have close performances. The meanings of the combinations are as below:

Table 5. the meaning of best three combinations

Combination	Bit order	Encoding method (0 – scalar encoding, 1 – one-hot-encoding)				
		"week"	"day of week"	"start time"	"work flow"	"file name"
22	10110	0	1	1	0	1
14	01110	0	1	1	1	0
30	11110	0	1	1	1	1

From the above table and the figure of training and test RMSE, we can find that the combinations with "week" encoded as scalar can achieve much better performance. Intuitively, the backup size changed in repeating pattern for 7-day period, and has not much dependence on the "week" feature. Therefore, when training and fitting the linear regression model, this feature should have 0 coefficient. If we encode it as scalar (only one number), then it's easier to find that it has no impact on backup size and can be assigned with 0 coefficient. While if we encode it as the vector, then we spread the impact into several parts, and it's possible to take it into consideration, which causes much deviation. Thus, for "week" feature, it should be with scalar encoding method.

While for the other four features, they are all important features to train the regression model. Besides, the specific values should be treated equally. In other words, for example, "day of week" feature has 7 values Monday to Sunday, though they can be mapped to 1 to 7, the numerical value only represents one day, not the ordinal sequence. "start time", "work flow" and "file name" the same. With these intuitive thought, the four features should be encoded as vector. From the table, we can find that within the tolerance of error, the 4 features with one-hot-encoding achieve better performances,

which verifies our guess.

Question 2(a) – v Controlling Ill-conditioning and Over-fitting

From the figure 11, we observe that the curve is serrated and vary much. For even value combinations, the test RMSEs are close to training RMSEs, while for odd value combinations, the test RMSEs are very large. It's because when the combinations are even numbers, the last bit is 0, which indicates that the feature "week" encoded with scalar numbers. This feature is not important to backup size, and encoded with only on scalar numbers, it's easier to find its irrelevance and set the coefficient to 0. However, for odd numbers, the last bit is 1, which indicates that the feature "week" encoded with vector. This will spread its impact into several parts and each part is either 1 or 0, which will make it difficult to find its irrelevance to the backup size. Thus, the trained regression model will produce deviate results and large RMSEs.

Next, we use three regularization methods Ridge Regularizer, Lasso Regularizer and Elastic Net Regularizer to control ill-conditioning and over-fitting problems. Below are our results:

```
using Ridge Regularizer
Optimal Combination:
use_one_hot: ['day_of_week', 'start_time', 'work_flow']
use_scalar: ['week', 'file_name']
Optimal Alpha: 5
Optimal Test Rmse: 0.0883677380035
estimated coefficients:
[  3.92371018e-02  -1.28385492e-02  -2.02296713e-02  -5.23861235e-03
  -5.69522387e-03   3.27107576e-03   1.49387908e-03  -2.01731751e-02
  -2.10260133e-02   7.77904203e-03   3.34016216e-02  -1.98635056e-03
   2.00487535e-03   3.88772567e-02  -1.36944804e-02  -4.01480968e-02
  -5.71822376e-02   7.21475581e-02   1.12198388e-05   5.87765230e-05]
----------------------------------------
```

Figure 13. Regularization with Ridge Regularizer

```
using Lasso Regularizer
Optimal Combination:
use_one_hot: ['week', 'day_of_week', 'start_time', 'work_flow', 'file_name']
use_scalar: []
Optimal Alpha: 0.001
Optimal Test Rmse: 0.0888033150298
estimated coefficients:
[ -7.89378832e-04  -3.97987629e-05   3.50273462e-04  -3.50408399e-04
   3.30577856e-04  -2.13416268e-04   8.20825549e-06   8.23510457e-04
   1.26465661e-04   1.23745592e-04  -5.69967202e-04   7.22425669e-04
  -1.84632908e-04  -1.14632874e-05  -3.26141889e-04   3.93117111e-02
  -1.28639774e-02  -2.02670034e-02  -5.24818883e-03  -5.70655632e-03
   3.27725480e-03   1.49676768e-03  -2.02050649e-02  -2.10615321e-02
   7.79151188e-03   3.34547540e-02  -1.98706723e-03   2.00741659e-03
   3.27638144e-02  -1.20563131e-02  -3.44588984e-02  -4.87757697e-02
   6.25272312e-02   4.74142587e-03   5.99085364e-03   5.66125491e-03
   5.63475837e-03   5.24421656e-03   5.49130147e-03  -1.62341162e-03
  -2.10708489e-03  -1.89836804e-03  -2.18193696e-03  -2.28671168e-03
  -1.95879803e-03  -6.48654465e-03  -6.07490112e-03  -5.60116924e-03
  -6.66748266e-03  -4.14978568e-03  -5.47899716e-03  -8.28995617e-03
  -7.96287188e-03  -7.90399935e-03  -8.26363303e-03  -8.31223878e-03
  -8.04309515e-03   1.01612981e-02   1.07247021e-02   1.09141982e-02
   9.84795292e-03   1.04940610e-02   1.03849672e-02]
----------------------------------------
```

Figure 14. Regularization with Lasso Regularizer

```
using Elastic Net Regularizer
Optimal Combination:
use_one_hot: ['week', 'day_of_week', 'start_time', 'work_flow', 'file_name']
use_scalar: []
Optimal Alpha1: 1e-05
Optimal Alpha2: 0.00999
Optimal Test Rmse: 0.0889835690943
estimated coefficients:
[ -5.47613163e-04  -0.00000000e+00   1.85837480e-04  -1.62756136e-04
   1.63718609e-04  -4.64811103e-05   0.00000000e+00   5.97510939e-04
   0.00000000e+00   0.00000000e+00  -3.52168422e-04   5.07015899e-04
  -1.99188348e-05   0.00000000e+00  -1.38659992e-04   3.67874154e-02
  -1.17706627e-02  -1.87687608e-02  -4.71642638e-03  -5.11965214e-03
   3.13152974e-03   1.46559163e-03  -1.90361249e-02  -1.97663581e-02
   7.29075431e-03   3.15254988e-02  -1.90049483e-03   1.85158202e-03
   3.23900707e-02  -1.18769001e-02  -3.33756988e-02  -4.71304450e-02
   6.09671235e-02   4.01513708e-03   4.97271621e-03   4.72163425e-03
   4.70127791e-03   4.40130394e-03   4.58776338e-03  -8.49238118e-04
  -1.21859984e-03  -1.05921285e-03  -1.27576343e-03  -1.35577652e-03
  -1.10536401e-03  -5.30775494e-03  -4.97963328e-03  -4.61440285e-03
  -5.43854606e-03  -3.50087906e-03  -4.52619630e-03  -7.14509952e-03
  -6.89247276e-03  -6.84699925e-03  -7.12475884e-03  -7.16229616e-03
  -6.95441990e-03   9.12569157e-03   9.56084119e-03   9.70720324e-03
   8.88369399e-03   9.38272003e-03   9.29846630e-03]
----------------------------------------
```

Figure 15. Regularization with Elastic Net Regularizer

```
using un-regularized best model
Optimal Combination:
use_one_hot: ['day_of_week', 'start_time', 'file_name']
use_scalar: ['week', 'work_flow']
Optimal Test Rmse: 0.0883701294703
estimated coefficients:
[  1.27121601e+11   1.27121601e+11   1.27121601e+11   1.27121601e+11
   1.27121601e+11   1.27121601e+11   1.27121601e+11   5.28985666e+10
   5.28985666e+10   5.28985666e+10   5.28985666e+10   5.28985666e+10
   5.28985666e+10  -5.61826018e+09  -5.61826018e+09  -5.61826018e+09
  -5.61826018e+09  -5.61826018e+09  -5.61826018e+09  -3.60363106e+09
  -3.60363106e+09  -3.60363106e+09  -3.60363106e+09  -3.60363106e+09
  -3.60363106e+09  -1.58900195e+09  -1.58900195e+09  -1.58900195e+09
  -1.58900195e+09  -1.58900195e+09  -1.58900195e+09   4.25627173e+08
   4.25627173e+08   4.25627173e+08   4.25627173e+08   4.25627173e+08
   4.25627173e+08   2.44025629e+09   2.44025629e+09   2.44025629e+09
   2.44025629e+09   2.44025629e+09   2.44025629e+09   1.23977661e-05
  -2.01462912e+09]
----------------------------------------
```

Figure 16. without regularization

From the figures above, we can find that the number of coefficient depends on the total features after encoding. For example, with Ridge Regularizer, the total features are 1+7+6+5+1=20, so we got 20 coefficients. With Lasso Regularizer, the total features are 15+7+6+5+30=63, so we got 63 coefficients. Next, by observation of the values of the coefficient, the order of the magnitude after regularization is from 10^{-5} to 10^{-2}, and the order of the magnitude before regularization is from 10^{8} to 10^{11}. This indicates that regularization can greatly decrease the estimated coefficient and make them more practical and reliable.

Question 2(b) Predict with random forest regression model

In this part, we use a random forest regression model for the regression task of Network Backup Dataset. We use mean square variance as loss function for this regression task. Since we use bootstrapping, in the random forest regression model, we can use Out of Bag (OOB) error to detect the model performance. In scikit-learn, the default metric of OOB error is out of bag R^2 score. Hence, we use both root of mean square error (RMSE) and OOB error to evaluate our random forest regression model performance.

. Question 2(b) – i RMSE and Out Of Bag error for initial model

Firstly, we build a random forest regression model based on the following initial values:

- Number of trees (n_estimators): 20
- Maximum number of features (max_features): 5
- Depth of each tree (max_depth): 4
- Bootstrap (bootstrap): True
- Out of Bag error (oob_score): True

Using 10 folds cross validation and calculate the average training average RMSE and test average RMSE. Average RMSE is calculated by summing up each fold's square error, dividing it by total number of folds, and then obtaining its square root. We obtain the average training and test RMSE, and OOB scores as the following values:

- Average Training RMSE: 0.06033022846142782
- Average Test RMSE: 0.06051692020455181
- Out of Bag error: 0.663879023186

For the details of training RMSE and test RMS of 10 folds cross validation, we have the following table.

Fold	Training RMSE	Test RMSE
1	0.06013267	0.0676306
2	0.06162003	0.05316943
3	0.06014079	0.06746134
4	0.06097094	0.05272497
5	0.05952577	0.06670115
6	0.06157289	0.05446148
7	0.0601267	0.06774232
8	0.05956976	0.05142071
9	0.05938959	0.06648393
10	0.06080396	0.05277848

Table 6. Training RMSE and Test RMSE of 10 folds cross validation

We also plot the graph of fitted values versus true values scattered over the dataset and the one of residuals against fitted values scattered over the dataset.

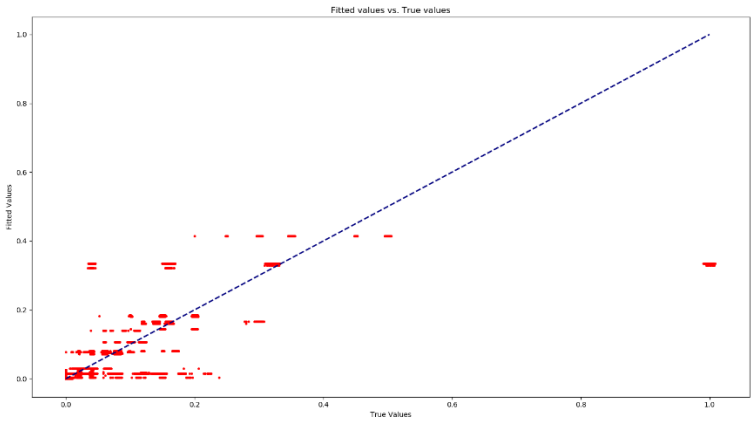


Figure 17. Fitted values vs. True values

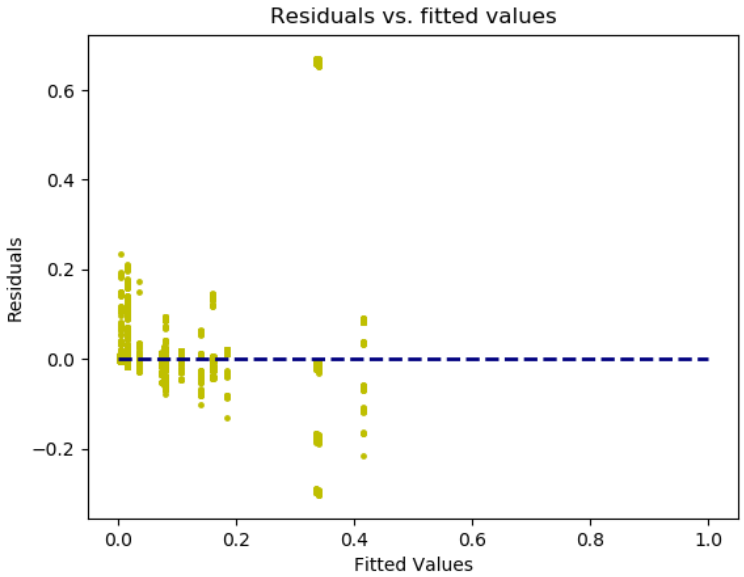


Figure 18. Residuals vs. Fitted values

. Question 2(b) – ii Optimal number of trees and maximum number of features

We sweep the number of trees from 1 to 200 and the maximum number of features from 1 to 5 to find the optimal number of trees and maximum number of features. We also plot the graph of OOB error versus number of trees and the one of average Test RMSE against number of trees.

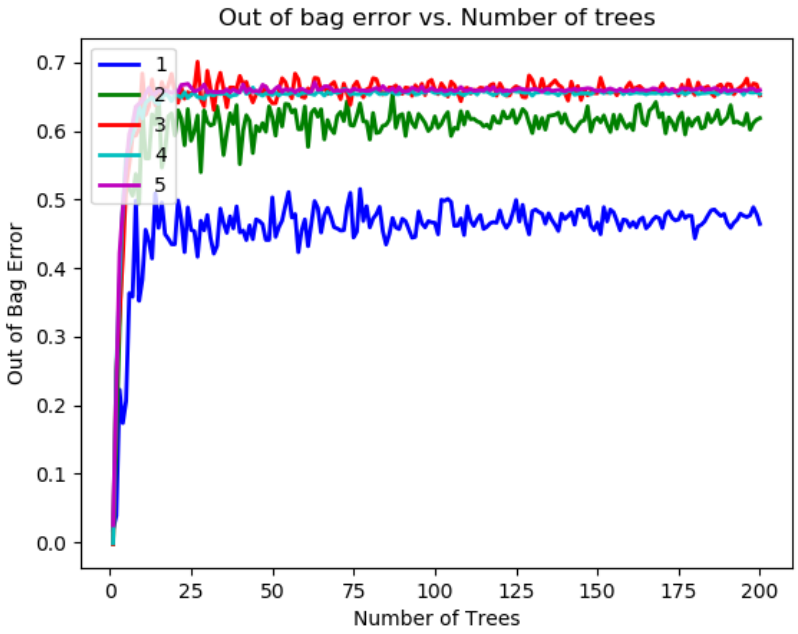


Figure 19. Out of bag error vs. Number of trees

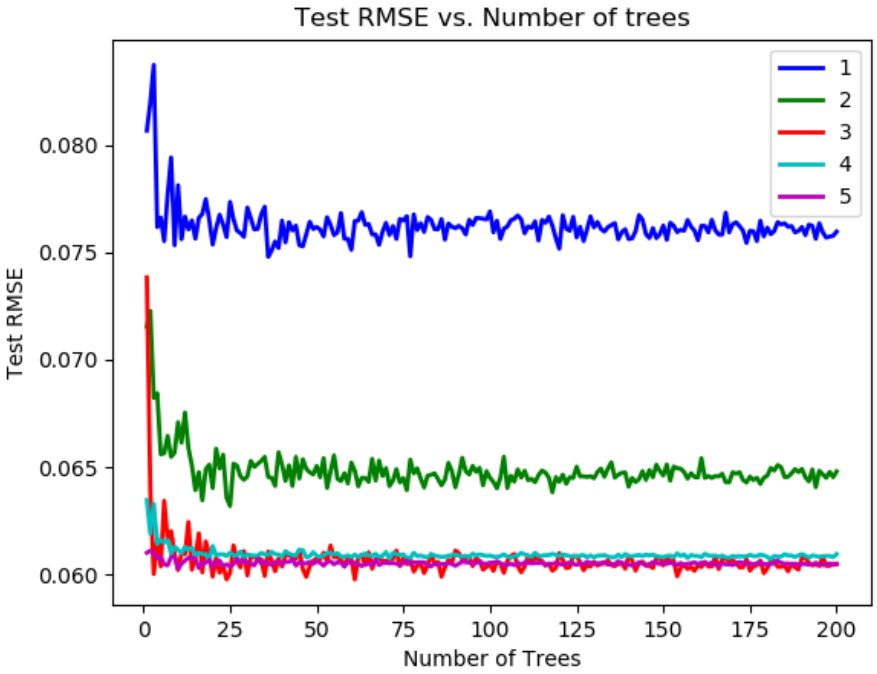


Figure 20. Test RMSE vs. Number of trees

The optimal hyper-parameters obtained by finding the maximum OOB errors:

- OOB errors: 0.700884771854

- Number of trees: 27
- Number of features: 3

The optimal hyper-parameters obtained by finding the minimum average test RMSE:

- Test RMSE: 0.0597642420802
- Number of trees: 24
- Number of features: 3

. Question 2(b) – iii Another Parameter

Besides max number of features, we also try the minimum impurity decrease. We sweep the minimum impurity decrease from 0 to 0.2 with the gap 0.04 and the number of trees from 1 to 200 as well. Similarly, to find the optimal minimum impurity decrease and the number of trees, we plot the graph of OOB error versus number of trees and the one of average Test RMSE against number of trees.

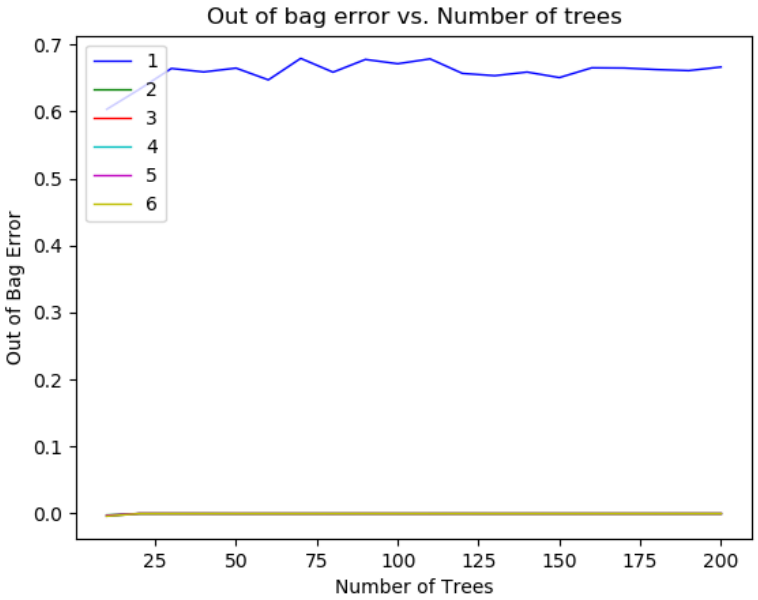


Figure 21. Out of bag error vs. Number of trees

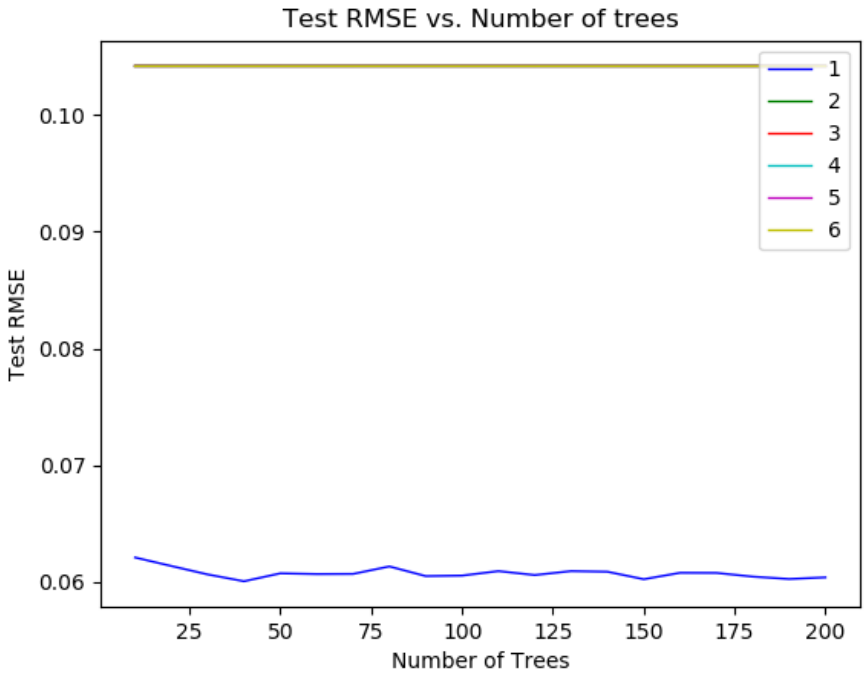


Figure 22. Test RMSE vs. Number of trees

The optimal hyper-parameters obtained by finding the maximum OOB errors:

- OOB errors: 0.679337058104
- Number of trees: 70
- Minimum impurity decrease: 0

The optimal hyper-parameters obtained by finding the minimum average test RMSE:

- Test RMSE: 0.060057567677
- Number of trees: 40
- Minimum impurity decrease: 0

. Question 2(b) – iv Optimal hyper-parameters

According the results gained from part ii) and part iii), we obtain the optimal hyper-parameters of the random forest regression model for this Network Backup Dataset.

- Number of trees: 27
- Number of features: 3
- Minimum impurity decrease: 0
- Depth of each tree: 4

- Bootstrap: True

. Question 2(b) – v Visualize decision trees

Using the optimal hyper-parameters displayed in part iv), we build the best random forest regression model and visualize one decision tree in the random forest. Based on this random forest regression model, we get the following RMSEs and OOB error:

- Training RMSE: 0.05972217990006018
- Test RMSE: 0.05963923047914849
- Out of Bag error: 0.651284637024

We also plot the graph of fitted values versus true values and the one of residuals against fitted values based on this regression model.

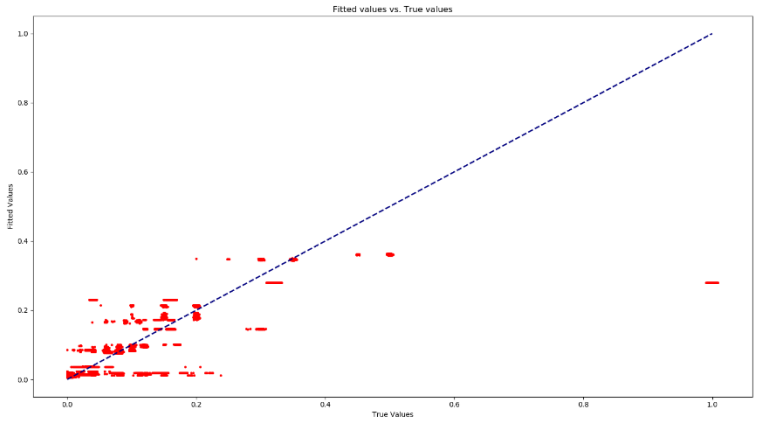


Figure 23. Fitted values vs. True values

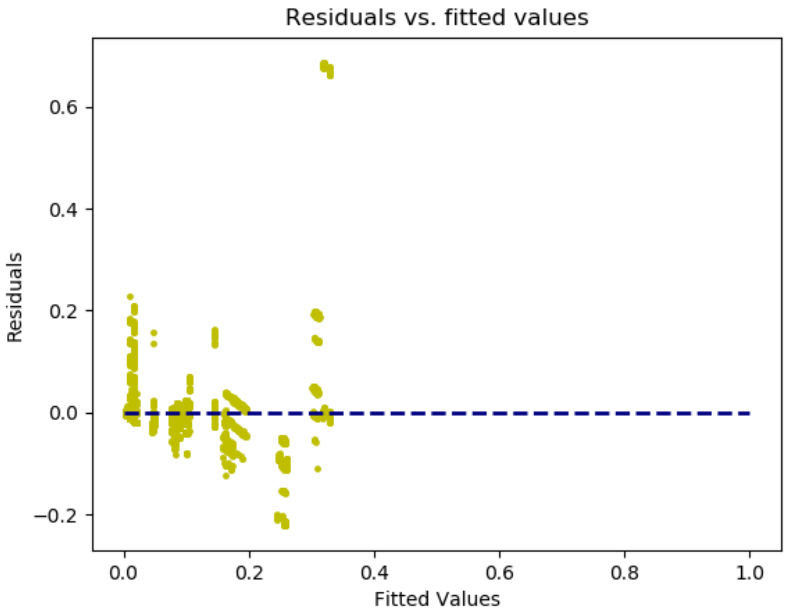


Figure 24. Residuals vs. Fitted values

We also plot the 3D graph of the true values and fitted values vs. two most important features among the all five ones, where orange dots represent true values and blue ones represent fitted values.

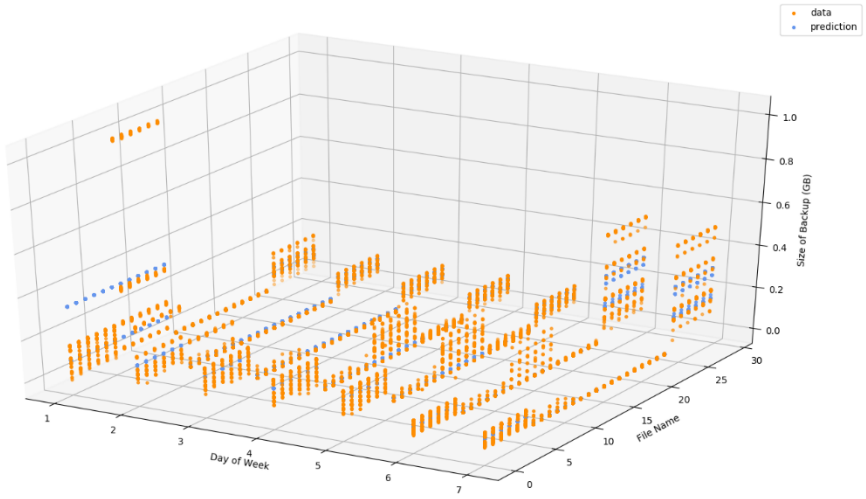


Figure 25. True/fitted values vs. two most important features

We get much lower RMSEs on both training and test dataset, which means this regression model is better than the one built in part i).

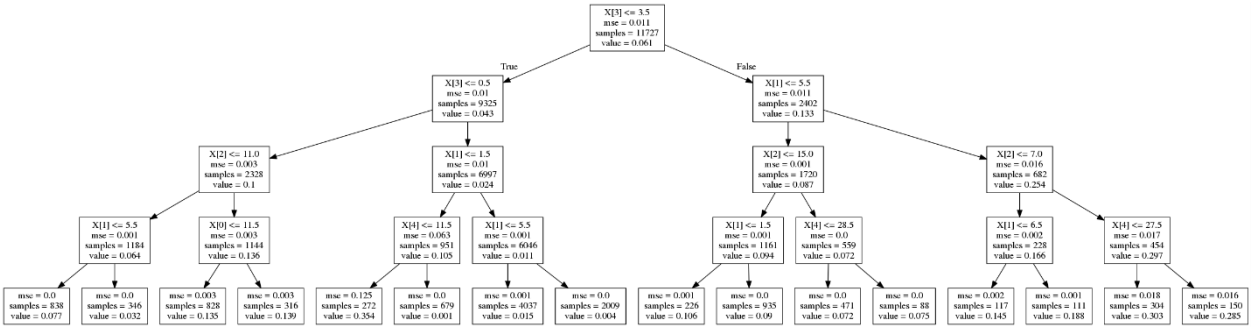


Figure 26. Structure of one decision tree from the random forest

The root node is based on the feature: Workflow ID. It is exactly the most important features with the highest feature importance: 0.318.

From both RMSE figures and 3D plot of true and fitted values, the random forest built with optimal parameters (number of trees = 27, number of features = 3, max depth of tree = 4, minimum impurity decrease = 0) has better performance the original one (number of trees = 20, number of features = 5, max depth of tree = 4, minimum impurity decrease = 0). We get lower training RMSE and test RMSE and from the 3D plot, we observe the fact that fitted values (blue dots) are very close to those true values (orange dots). Therefore, this random forest with optimal parameters has good regression result of the Network Backup dataset.

Question 2(c) Neural network regression model

In this part, we trained a neural network regression model with one hidden layer of the whole dataset encoded with one-hot-encoding method. Then we changed the number of hidden units from 10 to 200 in three activity functions (relu, logistic, tanh) respectively and plot the Test-RMSE vs number of hidden units figure as below:

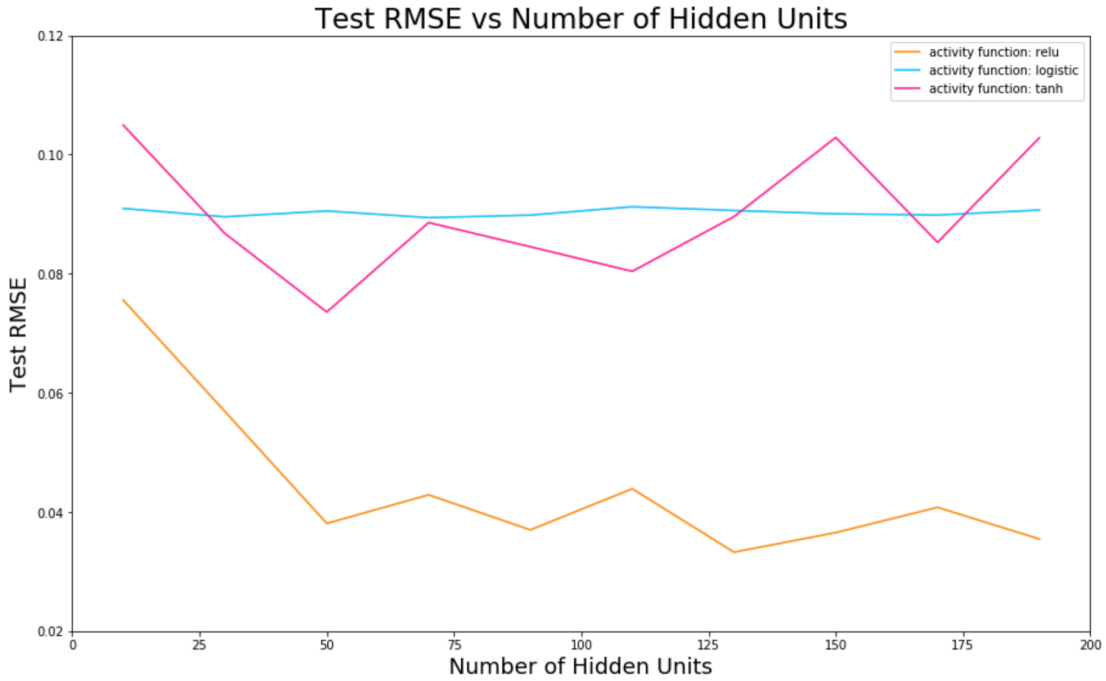


Figure 27. Test RMSE v.s. Number of Hidden Units

From the figure above, we find that for activity function relu and tanh, as the number of hidden units increases, the test RMSE becomes smaller. For activity function logistic, the test RMSE is relatively static at about 0.092. Besides, the activity function relu can achieve relatively smaller test RMSE than the other two functions. And the best combination is to use relu activity function with 130 hidden units.

Next, we use the best combination to plot fitted values vs true values figure and residuals vs fitted values figure. The fitted results are really good and the fitted values is very close to the true values. Below are our results:

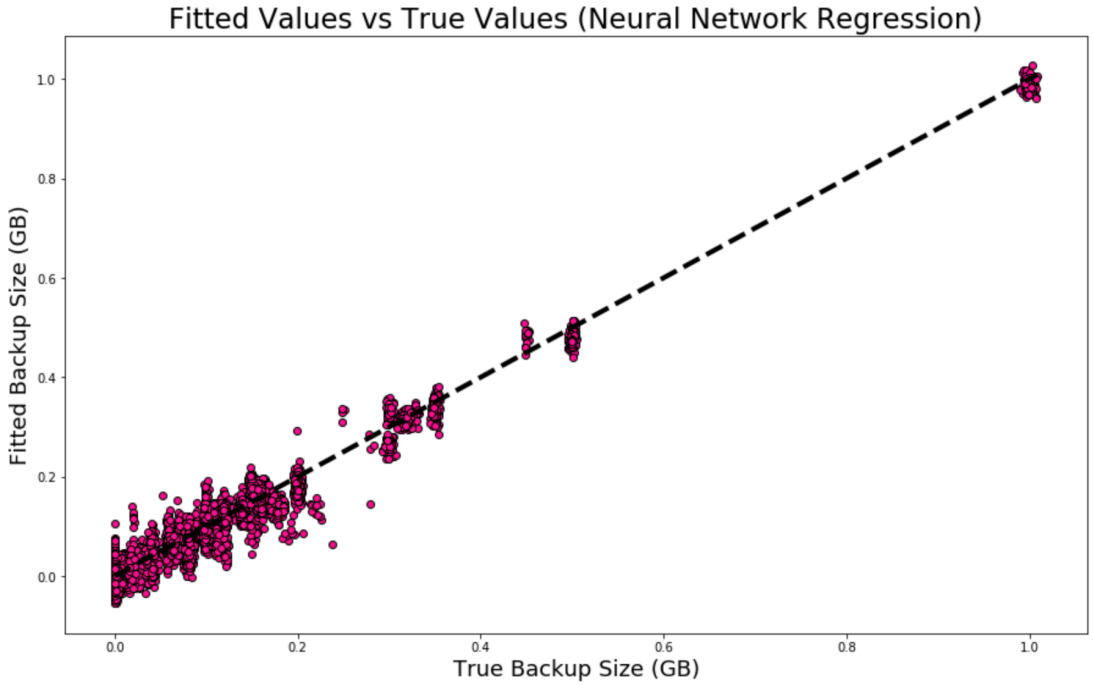


Figure 28. fitted values v.s. true values (neural network regression)

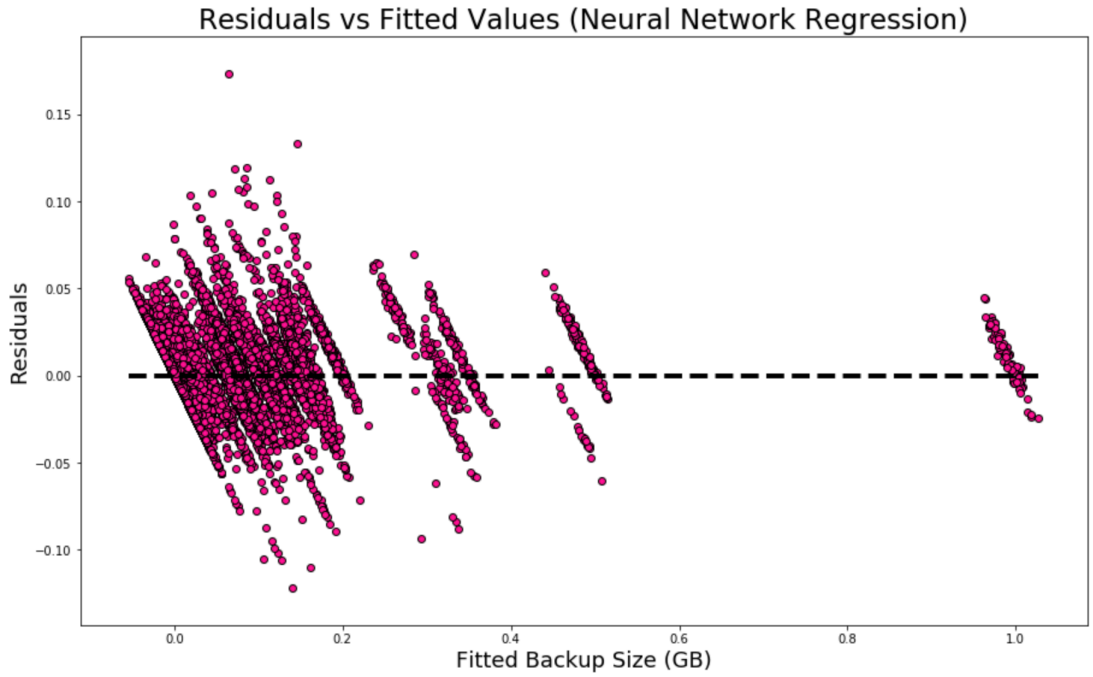


Figure 29. residuals v.s. fitted values (neural network regression)

Question 2(d) Predict the backup size

In this part, we use linear regression model and polynomial function model to predict

the backup size for each of the workflow separately, and then use 10-fold cross-validation to evaluate the performances and plot the RMSE figures.

Question 2(d) – i Predict with linear regression model

After splitting the whole dataset into five groups by work flow ID, we train and fit the linear regression model on each work flow respectively. Below are our results:

1) RMSE of each work flow

Table 7. RMSE of each work flow

Work Flow	RMSE
work flow 0	0.0255065911805
work flow 1	0.113405721099
work flow 2	0.0309727185747
work flow 3	0.00530162913819
work flow 4	0.0540565613482

2) Fitted values vs true values figure and residuals vs fitted values figure

   a) Work flow 0

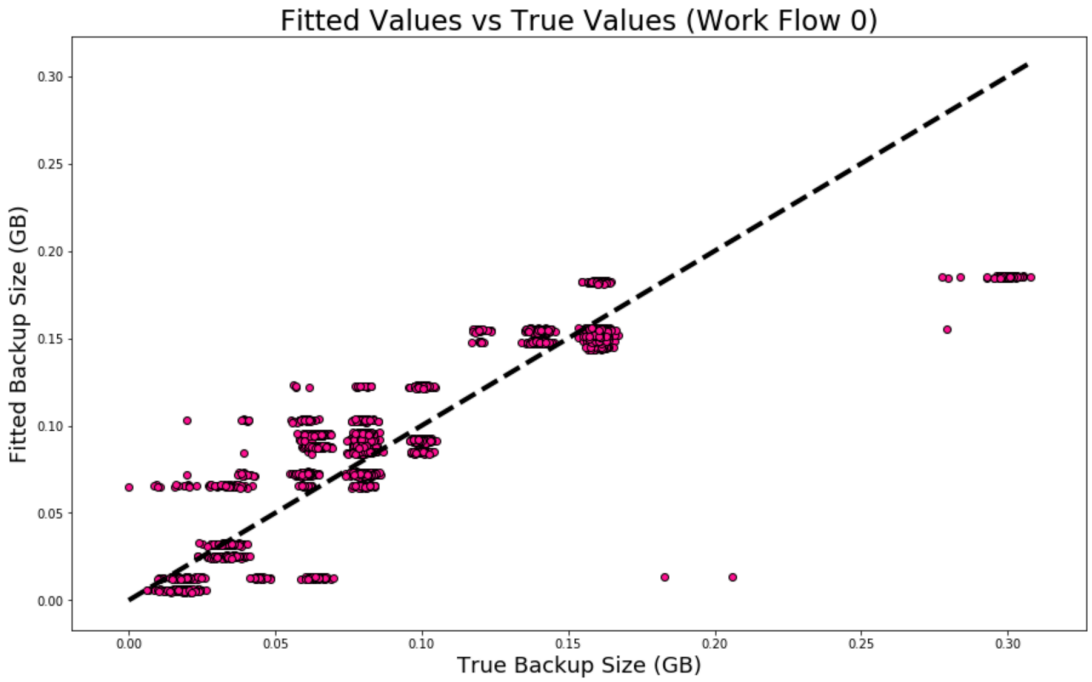


Figure 30. fitted values v.s. true values (work flow 0)

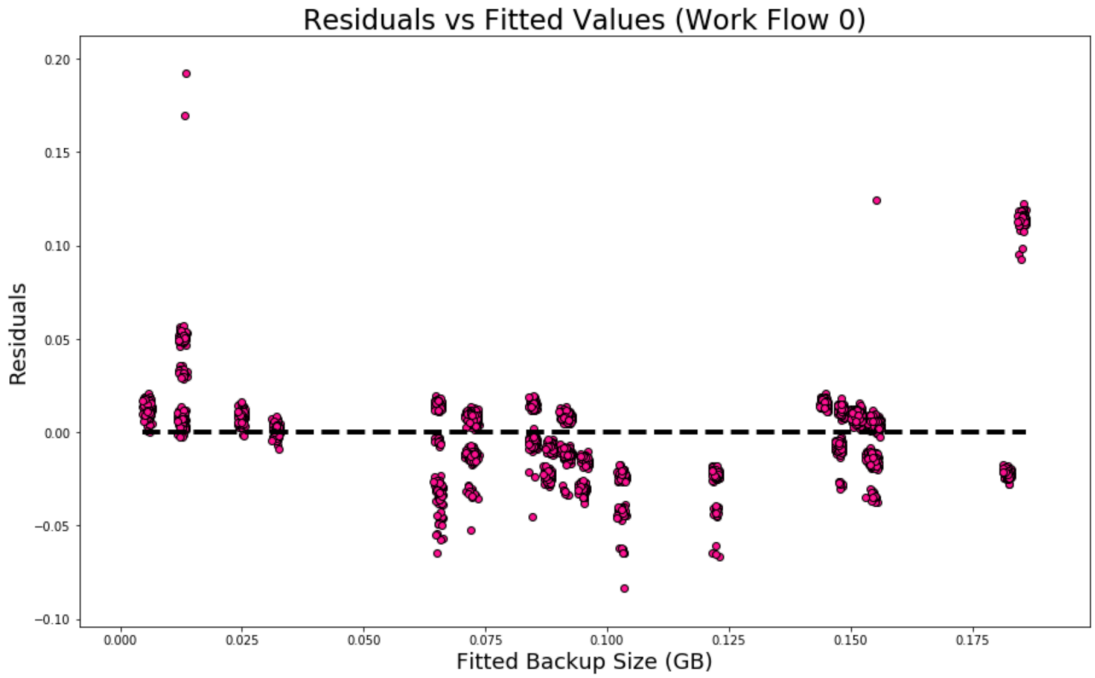


Figure 31. residuals v.s. fitted values (work flow 0)

b) Work flow 1

Figure 32. fitted values v.s. true values (work flow 1)

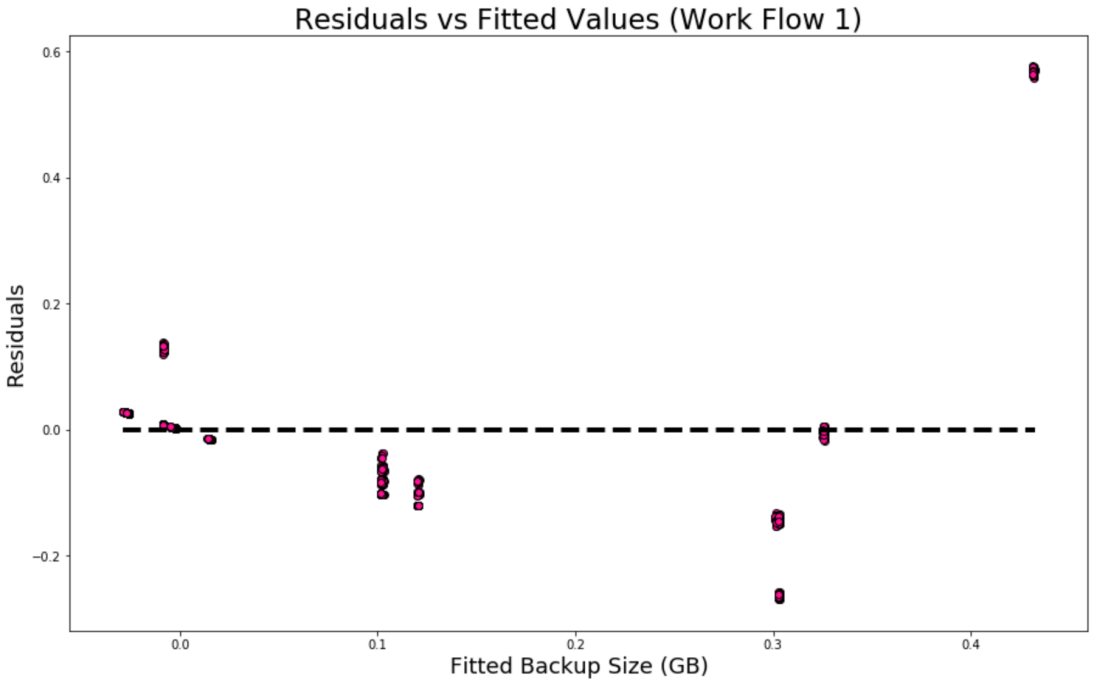


Figure 33. residuals v.s. fitted values (work flow 1)

c) Work flow 2

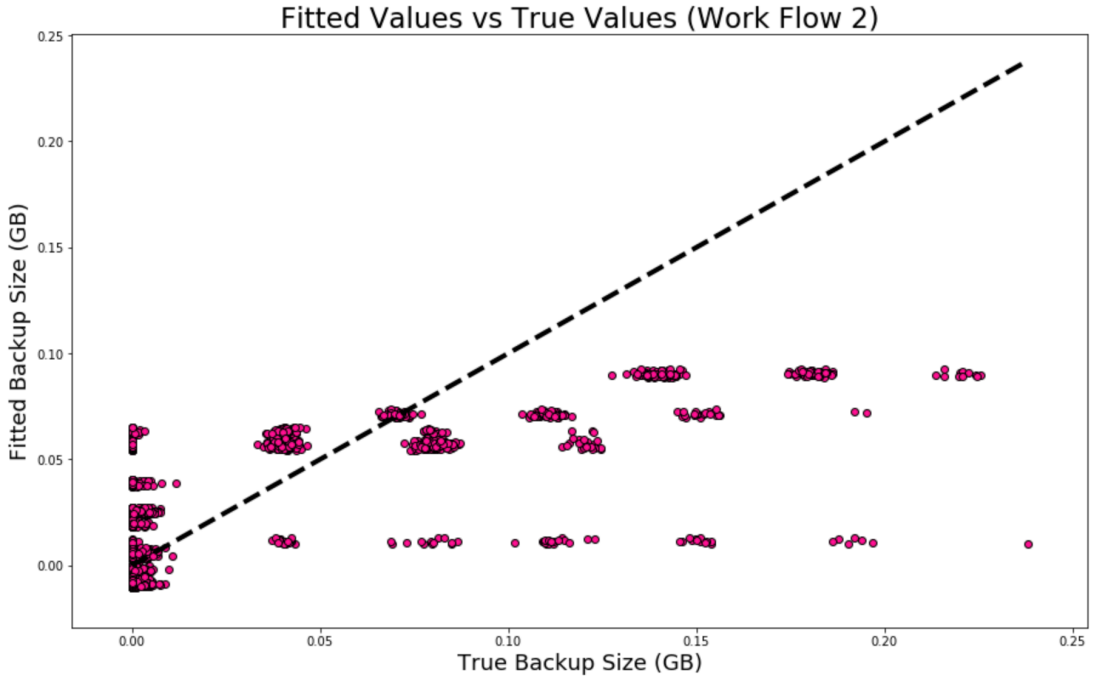


Figure 34. fitted values v.s. true values (work flow 2)

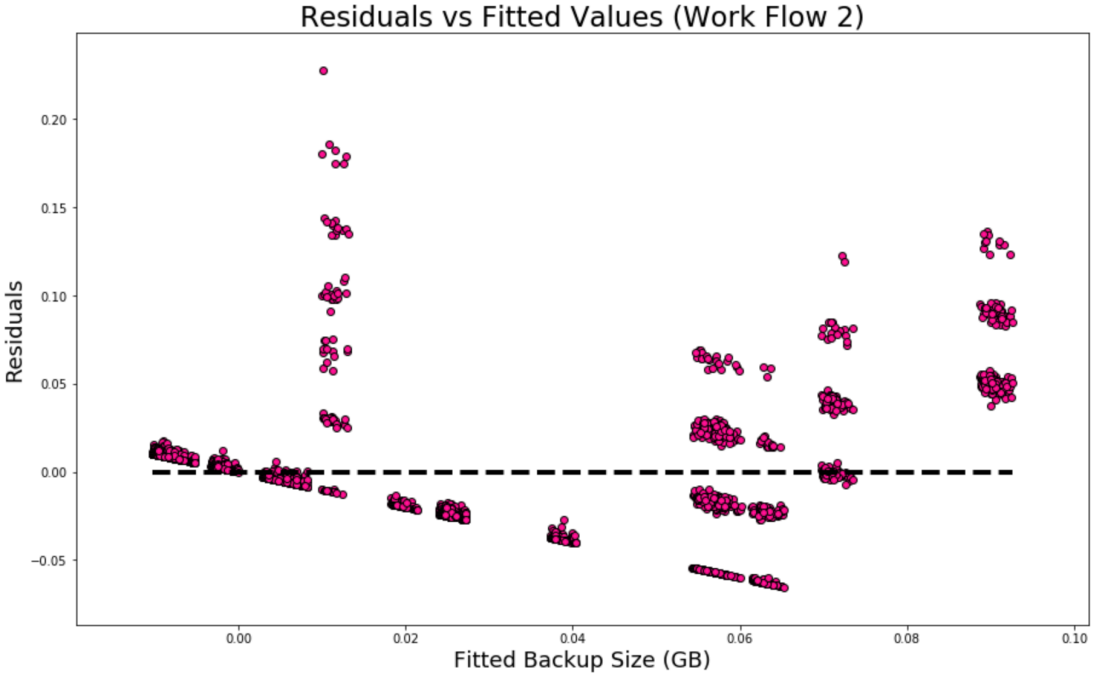


Figure 35. residuals v.s. fitted values (work flow 2)

d) Work flow 3

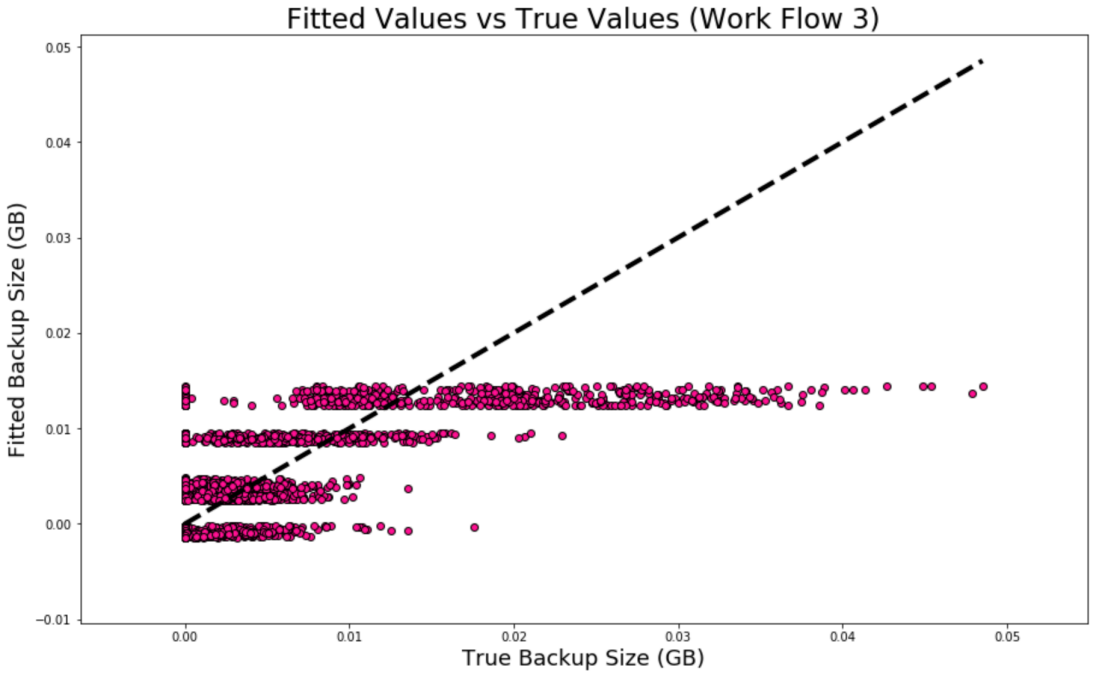


Figure 36. fitted values v.s. true values (work flow 3)

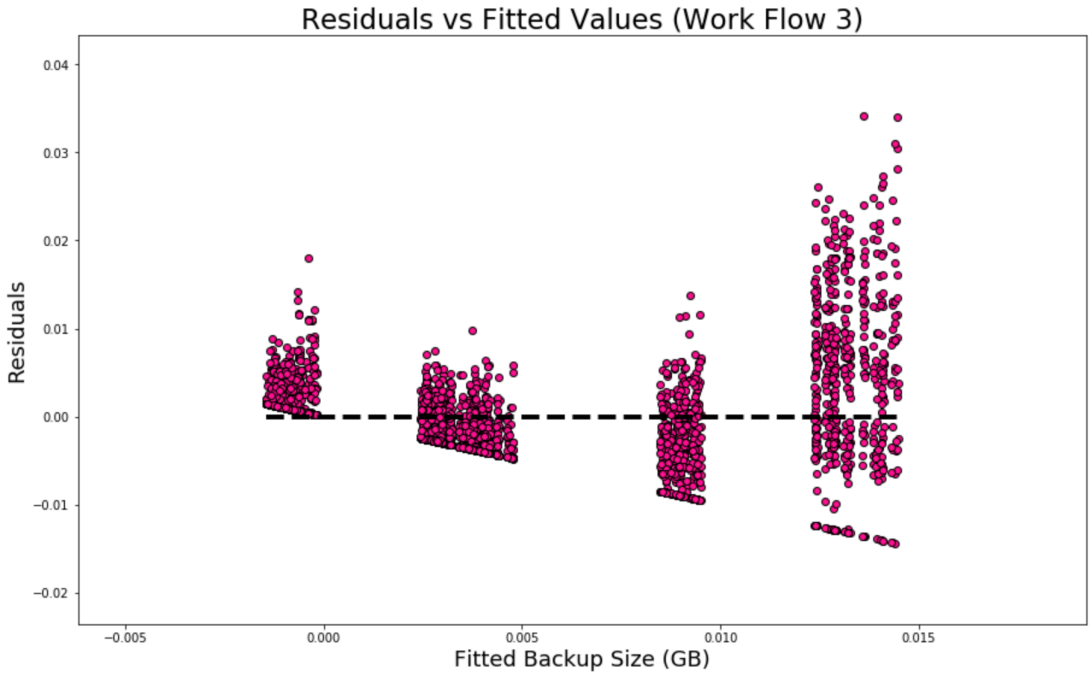


Figure 37. residuals v.s. fitted values (work flow 3)

e) Work flow 4

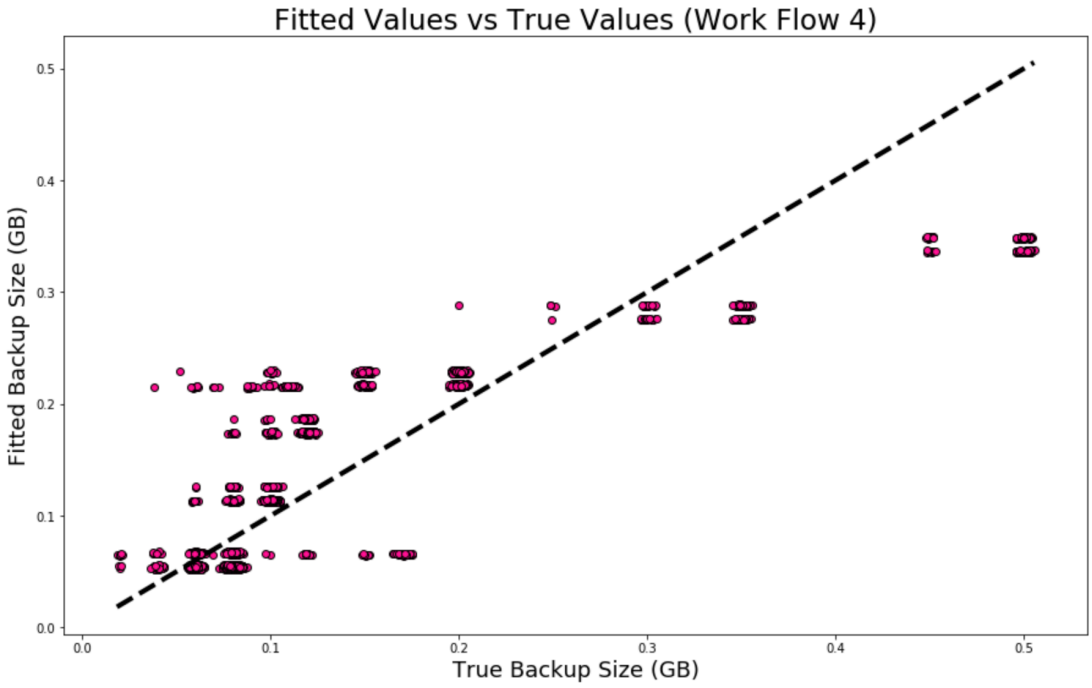


Figure 38. fitted values v.s. true values (work flow 4)

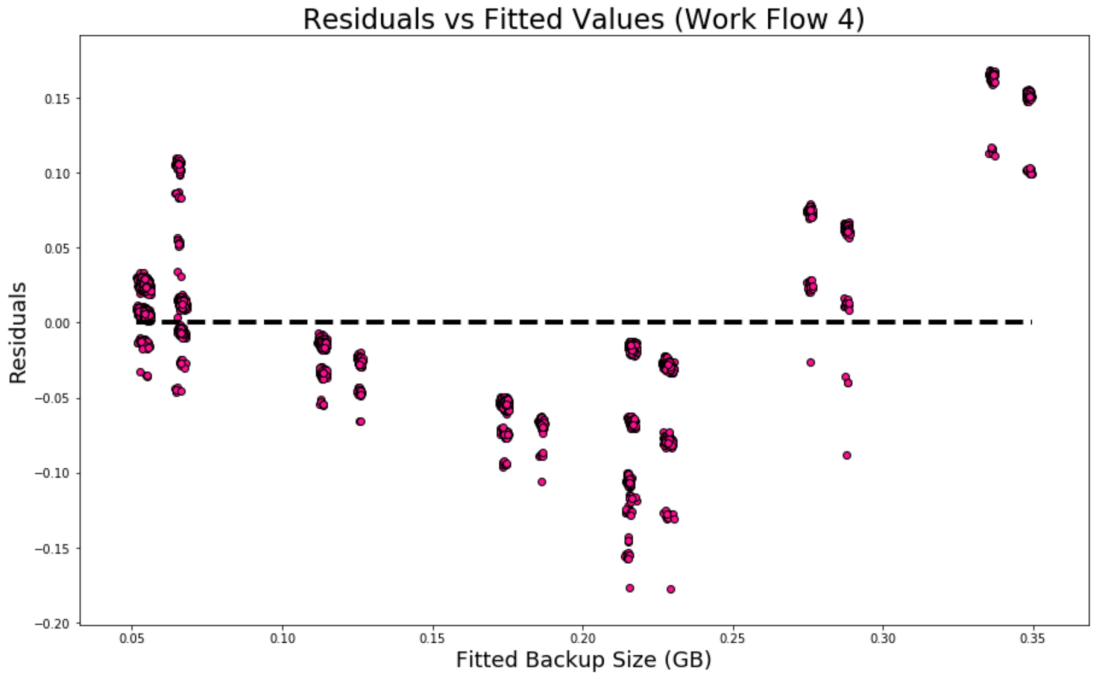


Figure 39. residuals v.s. fitted values (work flow 4)

From the table and figures above, we find that the performances are generally improved after using linear regression model on each work flow separately. Without separating them, the RMSE is about 0.1, while after separating, the RMSE of work flow 1 increases even larger than that before. For other work flows, the RMSEs decrease obviously, and for work flow 3, the fitting effect improved a lot and the RMSE was reduced to a very low level. And by observation of the figures, the size values are reduced to specific range according to specific work flow. We can observe that for work flow 1, the fitting results are obviously worse than the other work flow fitting results. In work flow 1 fitted value vs true values figure, the points do not follow the diagonal line obviously, but for the other work flow figure, we can observe the trend of those points follows the trend from bottom left to the upper right corner of the figure.

Question 2(d) – ii Predict with polynomial function

In this part, we use polynomial function regression to train and fit the data of each work flow separately. Below are our results:

1) RMSE of each work flow

Table 8. average training & test RMSE of each work flow

	workflows	degrees	train_rmse	test_rmse
0	0	1	0.089940	0.095090
1	0	2	0.076885	0.087282
2	0	3	0.069127	0.087041
3	0	4	0.063711	0.086856
4	0	5	0.060480	0.103205
5	1	1	0.099830	0.105873
6	1	2	0.088378	0.104592
7	1	3	0.078497	0.103653
8	1	4	0.070946	0.100697
9	1	5	0.067058	0.109269
10	2	1	0.081014	0.084952
11	2	2	0.065135	0.073449
12	2	3	0.057027	0.072203
13	2	4	0.047347	0.069511
14	2	5	0.040776	0.068304
15	3	1	0.054037	0.054196
16	3	2	0.017394	0.017970
17	3	3	0.016531	0.018530
18	3	4	0.015618	0.019415
19	3	5	0.014741	0.020946
20	4	1	0.054039	0.054199
21	4	2	0.017427	0.018043
22	4	3	0.016523	0.018526
23	4	4	0.015624	0.019441
24	4	5	0.014778	0.020944

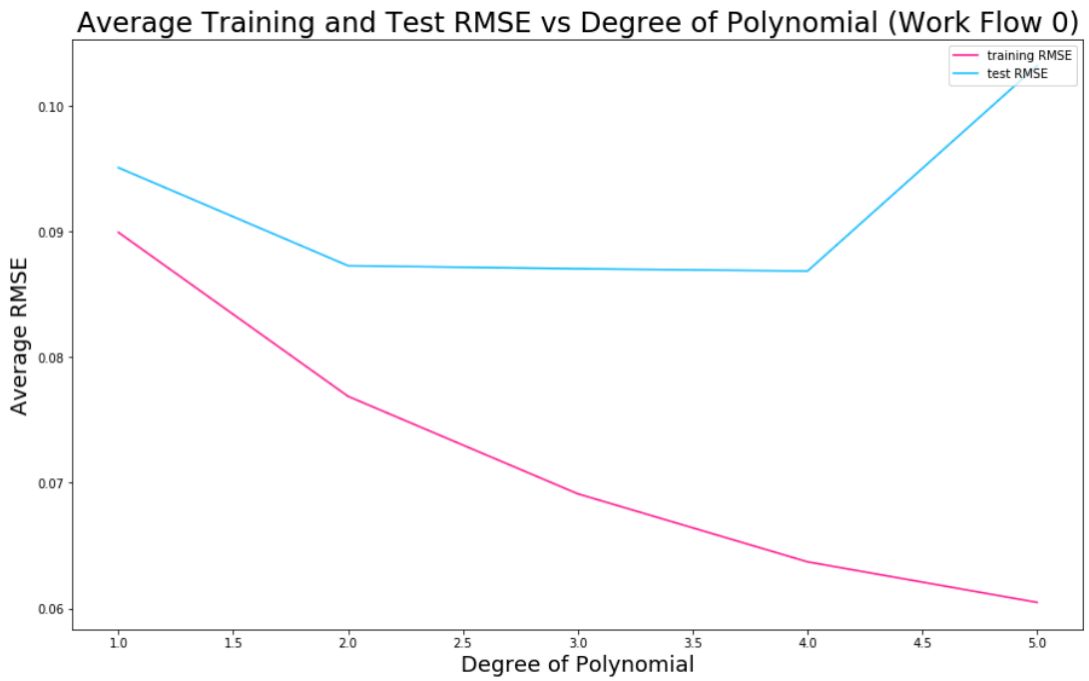


Figure 40. average training & test RMSE v.s. degree of polynomial (work flow 0)

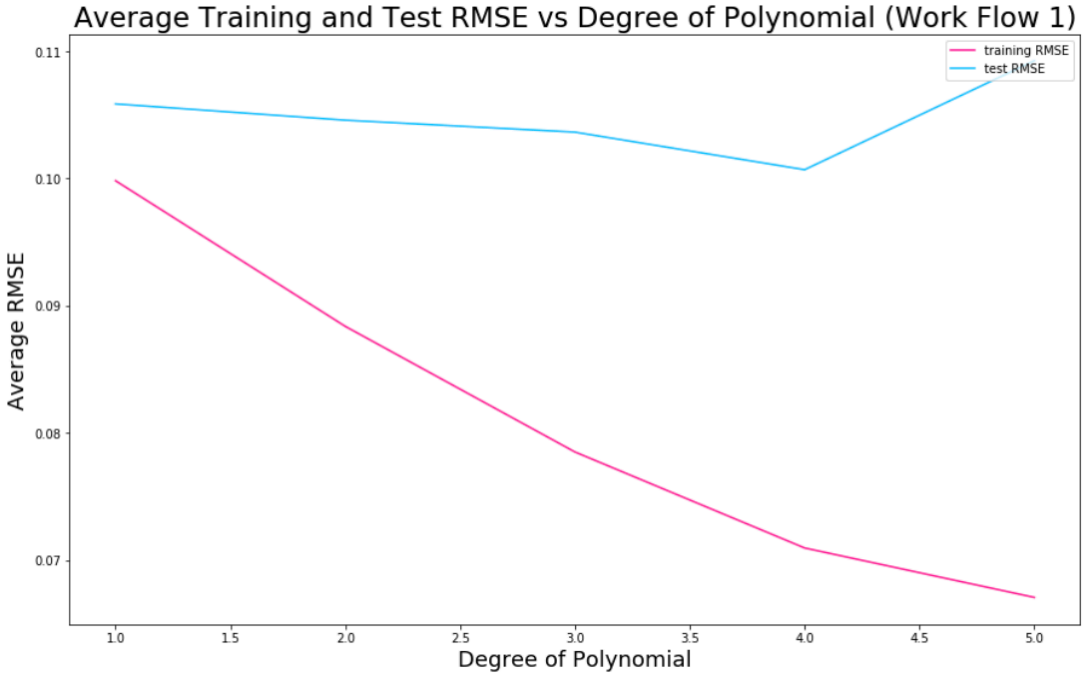


Figure 41. average training & test RMSE v.s. degree of polynomial (work flow 1)

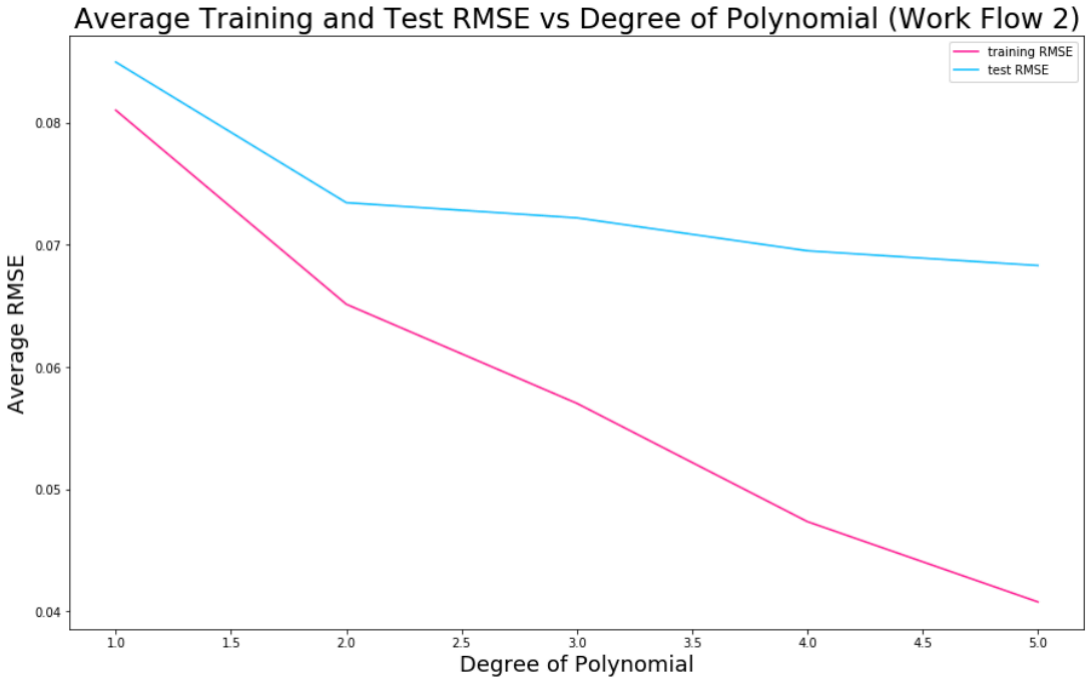


Figure 42. average training & test RMSE v.s. degree of polynomial (work flow 2)

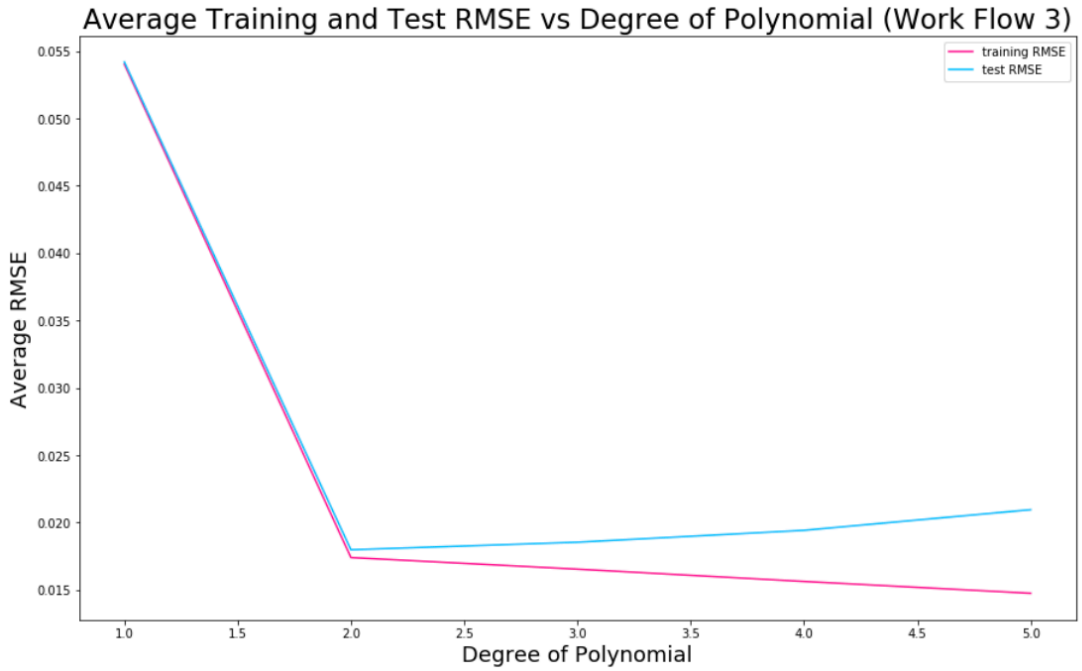


Figure 43. average training & test RMSE v.s. degree of polynomial (work flow 3)

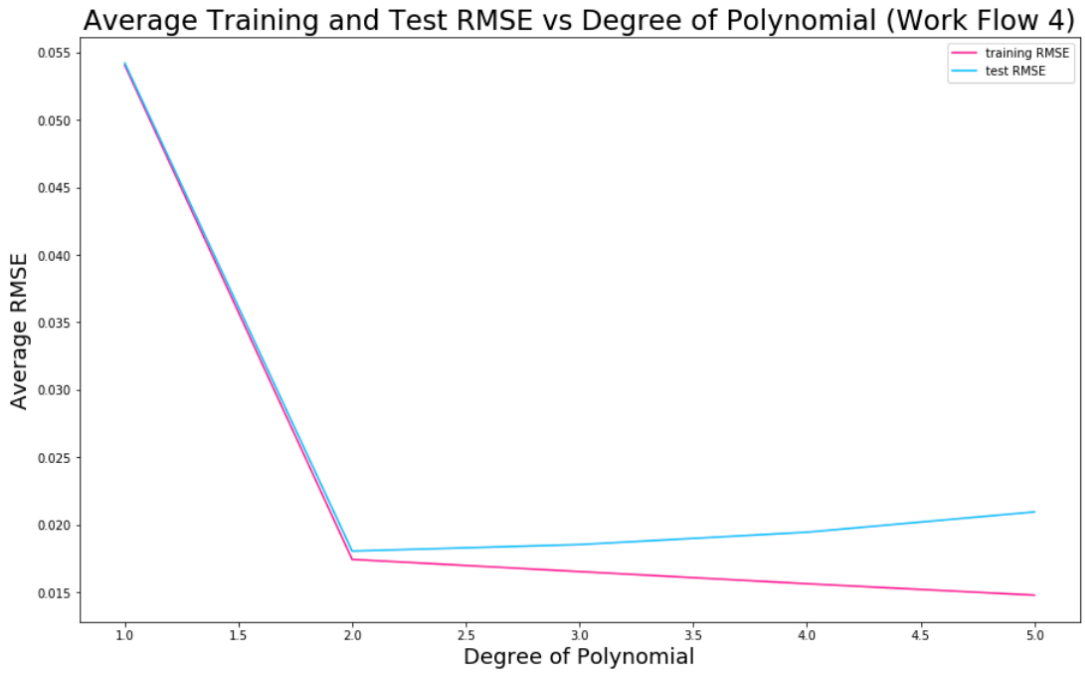


Figure 44. average training & test RMSE v.s. degree of polynomial (work flow 4)

2) Fitted values vs true values figure and residuals vs fitted values figure

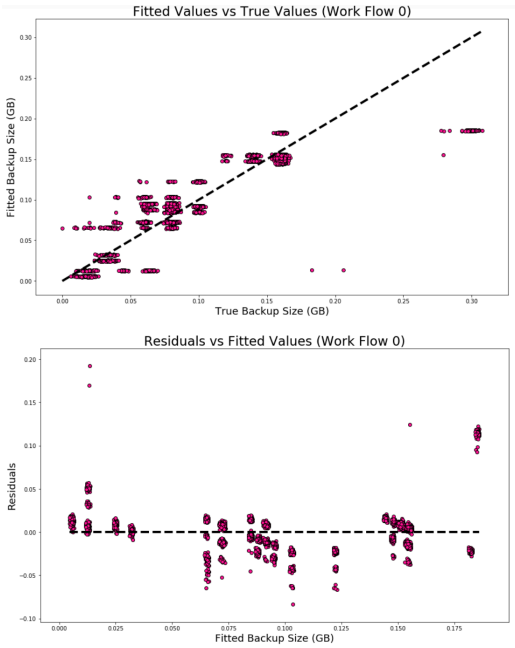


Figure 45-1. Degree 1

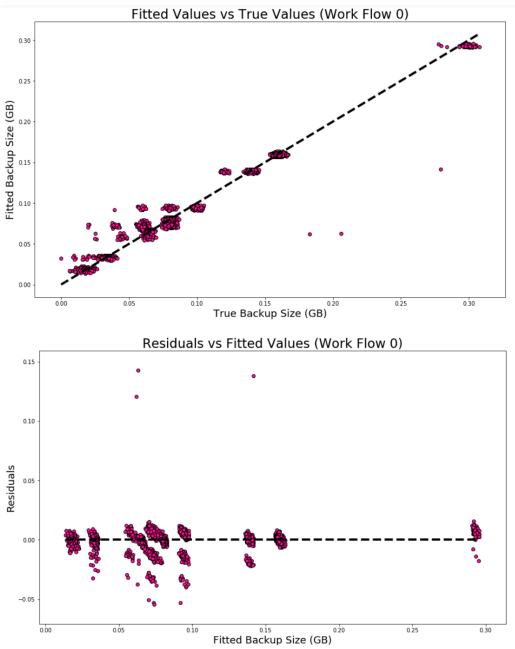


Figure 45-2. Degree 2

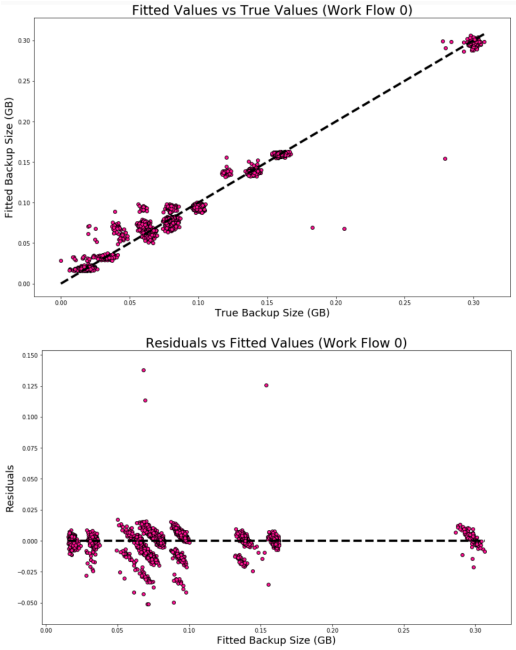


Figure 45-3. Degree 3

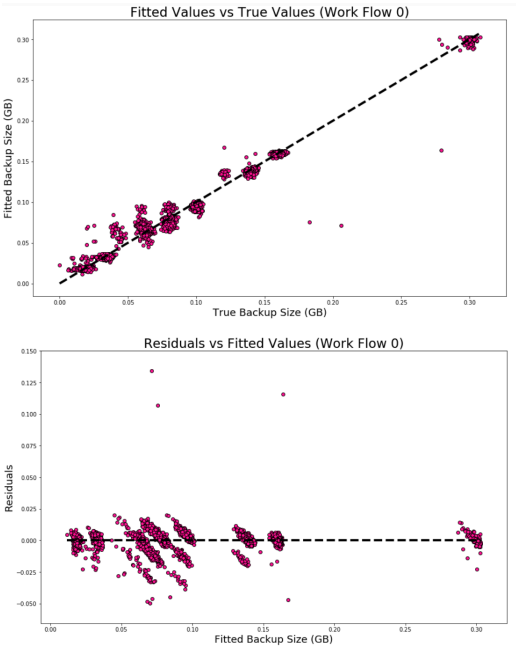
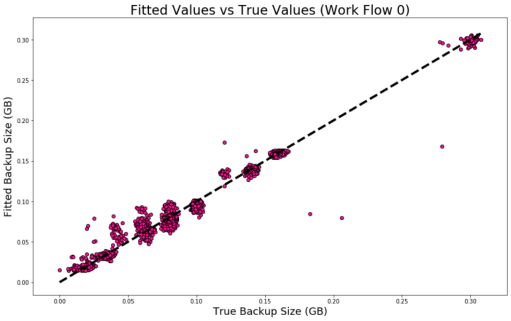


Figure 45-4. Degree 4



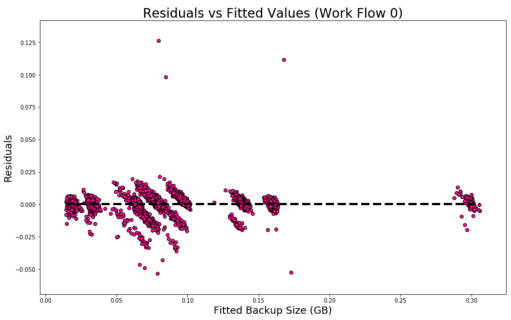


Figure 45-5. Degree 5

Figure 45. fitting model visualization (work flow 0)

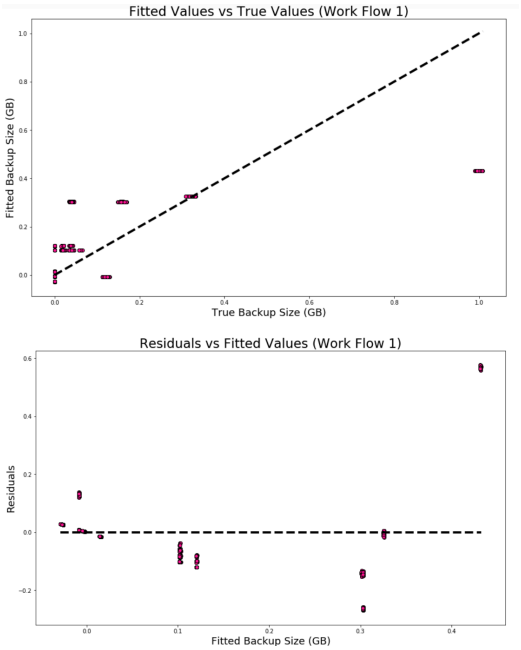


Figure 46-1. Degree 1

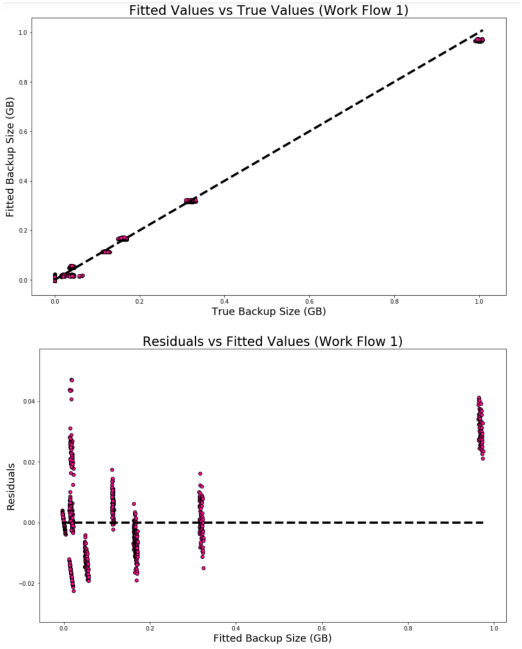


Figure 46-2. Degree 2

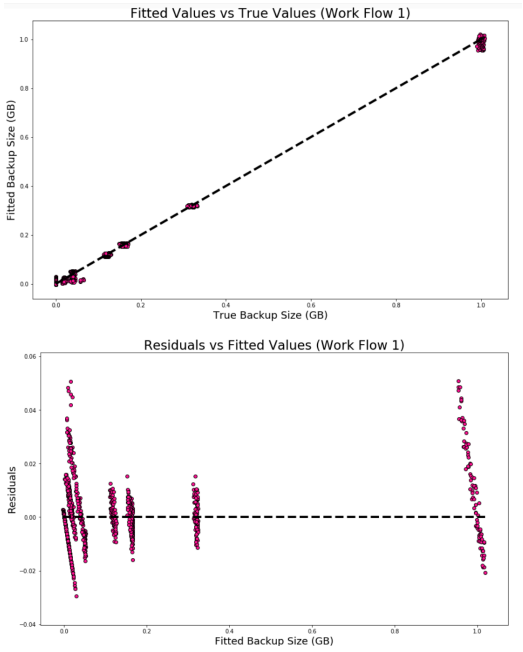


Figure 46-3. Degree 3

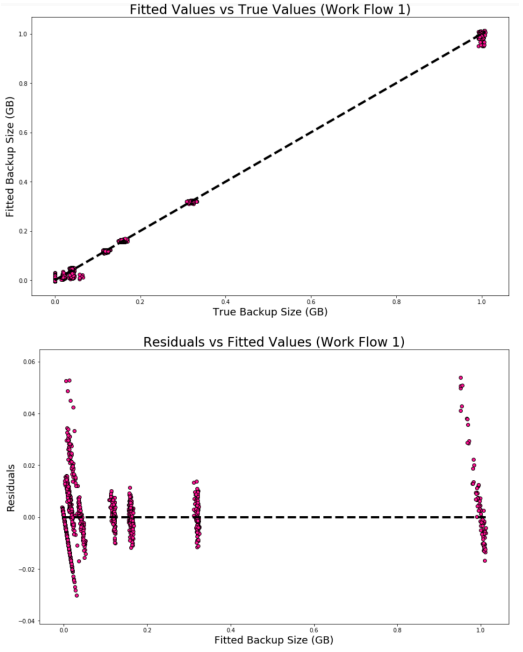


Figure 46-4. Degree 4

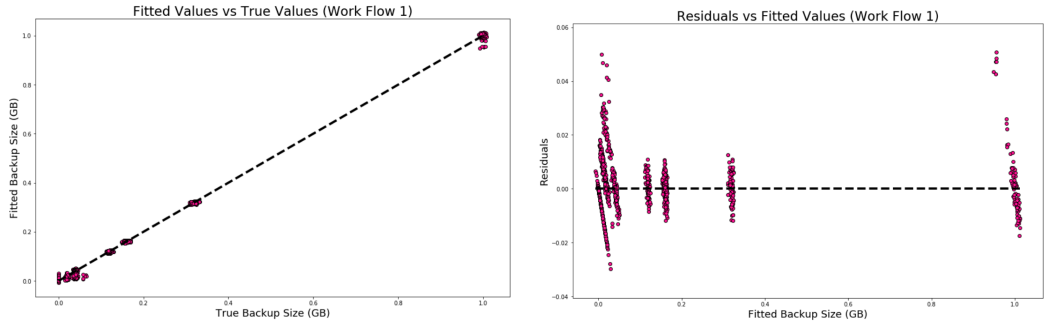


Figure 46-5. Degree 5

Figure 46. fitting model visualization (work flow 1)

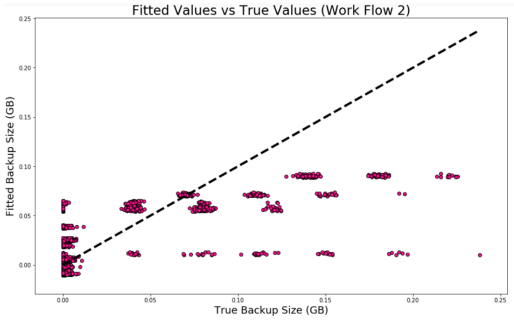


Figure 47-1. Degree 1

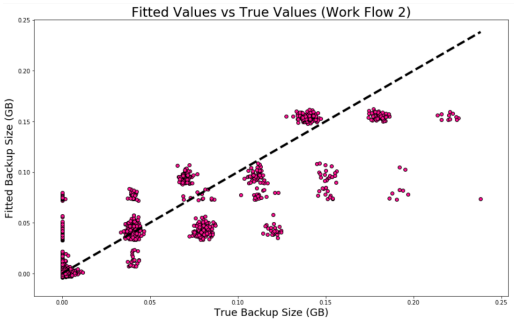


Figure 47-2. Degree 2

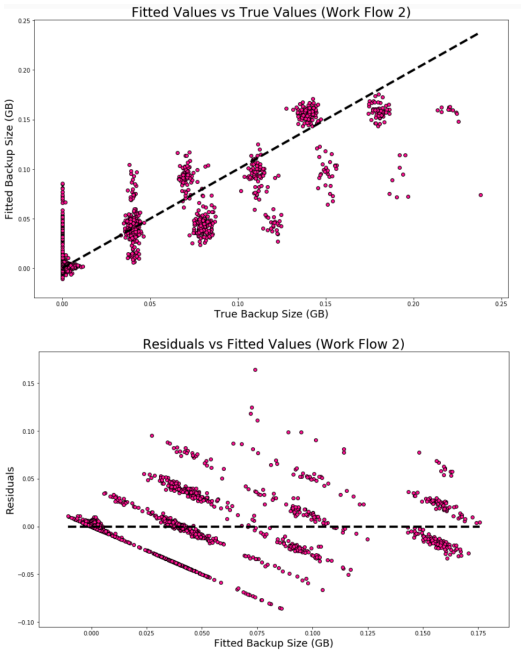


Figure 47-3. Degree 3

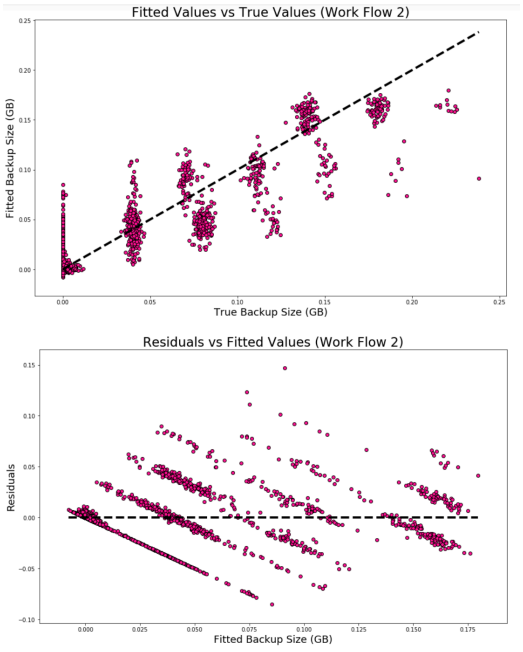


Figure 47-4. Degree 4

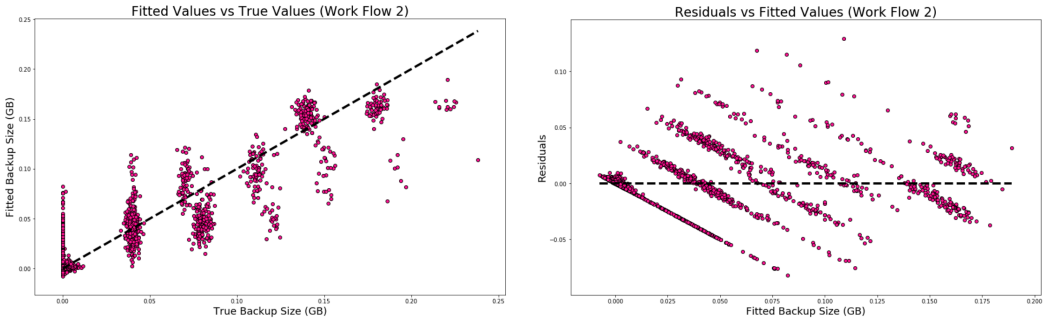


Figure 47-5. Degree 5

Figure 47. fitting model visualization (work flow 2)

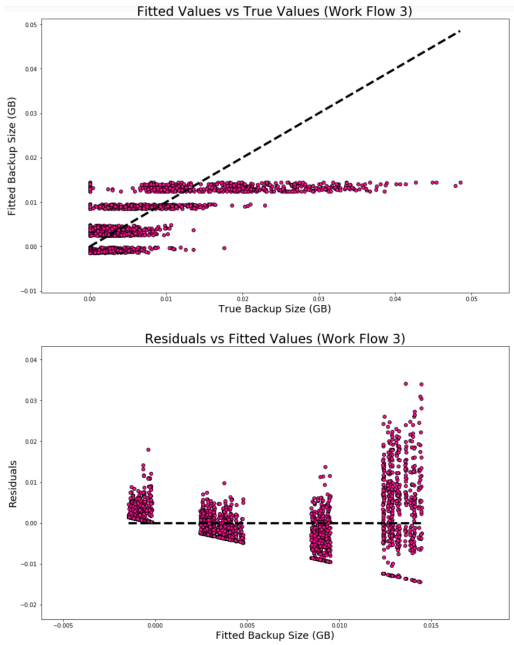


Figure 48-1. Degree 1

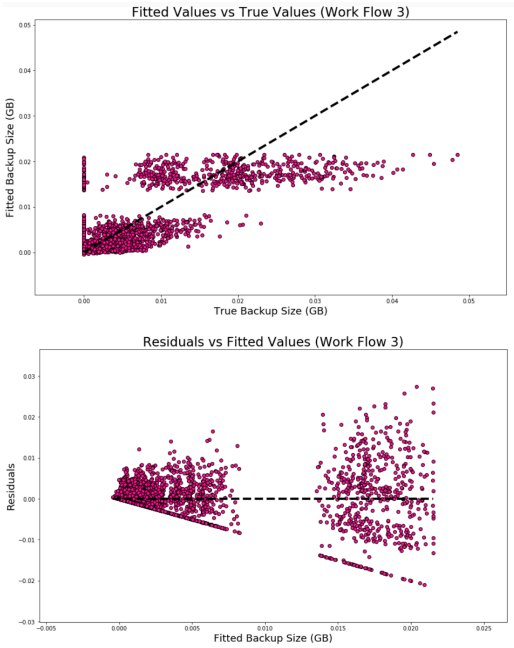


Figure 48-2. Degree 2

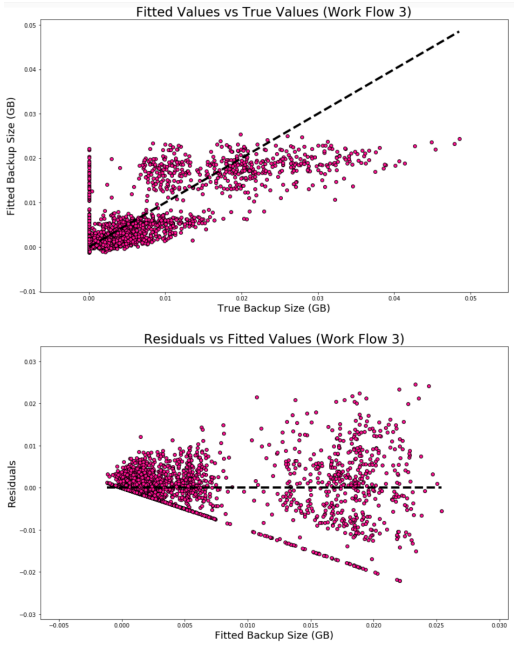


Figure 48-3. Degree 3

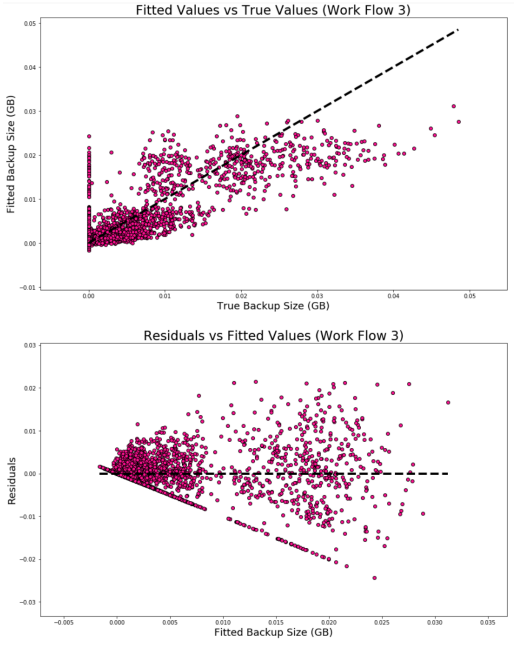
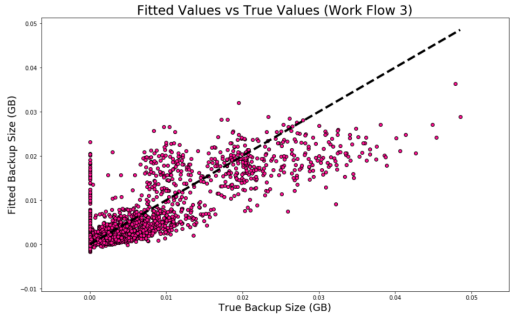


Figure 48-4. Degree 4



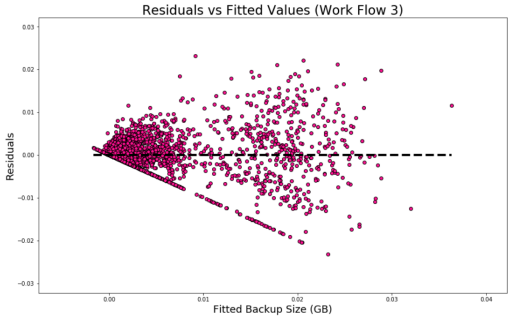


Figure 48-5. Degree 5

Figure 48. fitting model visualization (work flow 3)

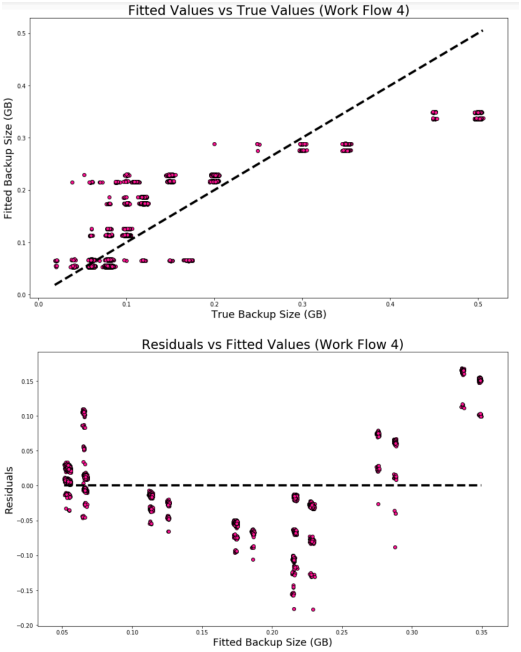


Figure 49-1. Degree 1

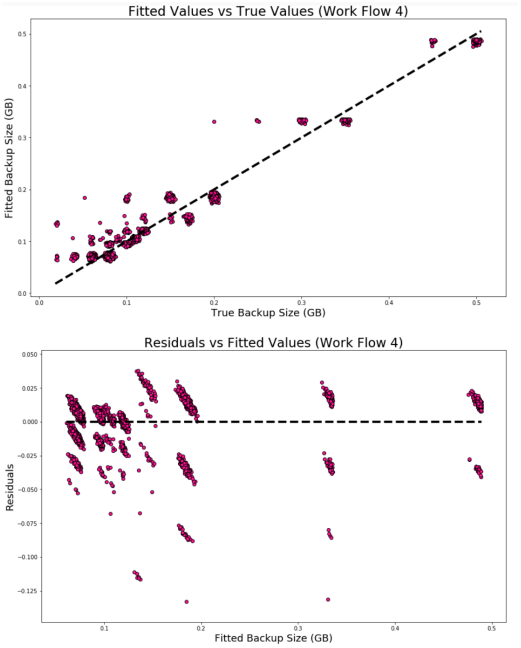


Figure 49-2. Degree 2

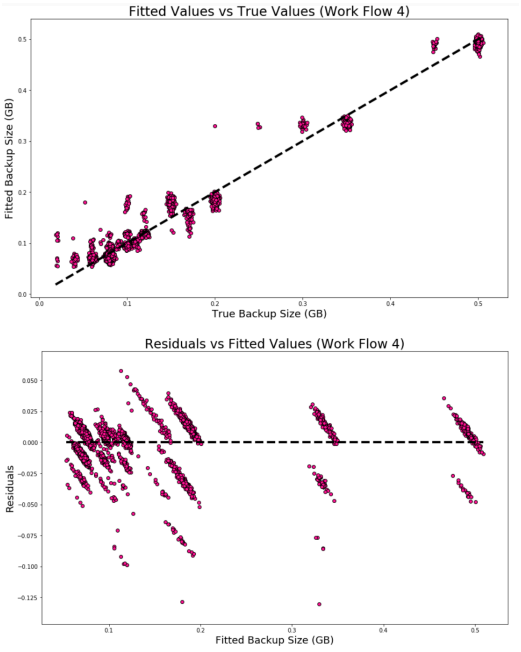


Figure 49-3. Degree 3

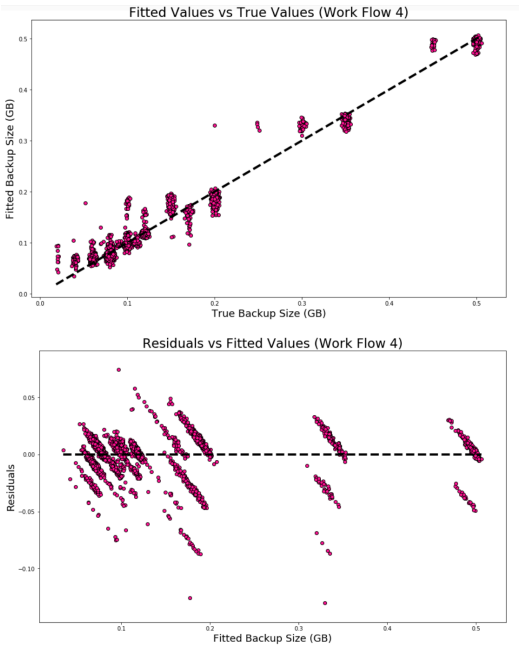


Figure 49-4. Degree 4

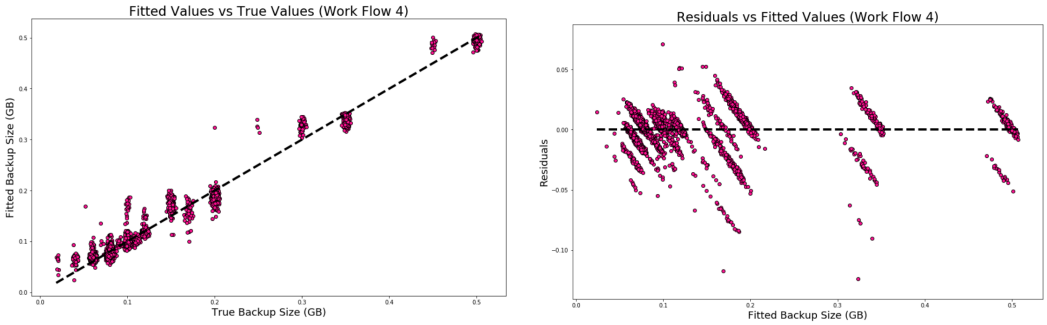


Figure 49-5. Degree 5

Figure 49. fitting model visualization (work flow 4)

From the table and figures above, we can find that as the degree of polynomial increases, the RMSEs decrease correspondingly. While for larger degree, the reduction of RMSEs become slow. So we may deduce that after increasing to some degree, the effectiveness will not be apparent and may even get worse at the threshold. From our figures, for work flow 0 and 1, after degree 4, the test RMSE becomes worse; for work flow 2, the test RMSE decreases continually; for work flow 3 and 4, after degree 2, the test RMSE becomes gradually worse.

Cross validation can help control the complexity of the model since it can make estimation of generalization more conservatively by separating model selection from testing. For over-fitting problem, training RMSEs are small and test RMSEs are large. So cross validation will use the parameters obtained from the training data to test other data. In this process, the complexity of the model can be controlled to some degree.

Question 2(e) k-nn Regression

In this part, we first standardize the feature and output to train k-nearest neighbor regression model on the dataset from k=1 to 30, and find the best parameter k. For each k, we do 10-fold cross-validation to get the train and test RMSE to prevent overfitting. Train and test RMSE vs k is shown below:

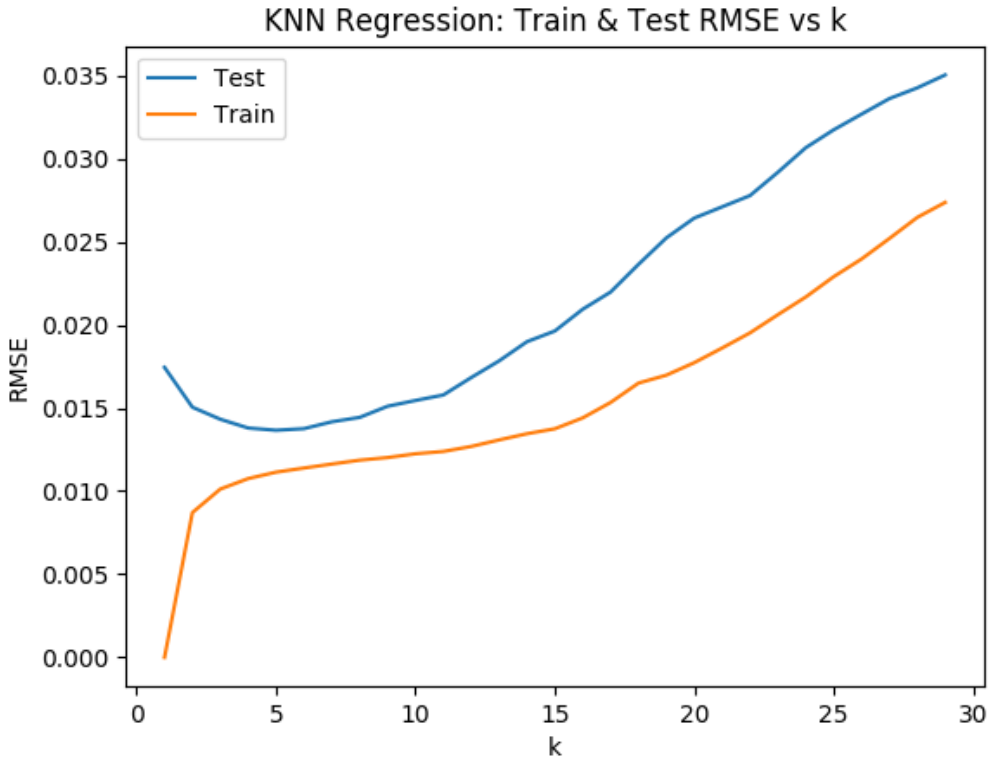


Figure 50. KNN Regression with Cross Validation: Train and Test RMSE vs k

From the figure above, we can observe that the test RMSE has min at k=5, average test RMSE is 0.01366, which means the most effective neighbor size is 5 for predicting test data with k-nn regression. When k smaller than 5, as k grows larger, the test RMSE becomes smaller because the number of "correct neighbors" increases, it improves the prediction. When k larger than 5, as k grows larger, the test RMSE becomes larger because the number of "incorrect neighbors" increases, the incorrect neighbor number gradually surmounts the correct neighbors number, so the prediction becomes poorer.

Next, after we get the optimal k of k-nn regression, we find out the 10-fold train and test RMSE, and it shows in below table:

Table 9. Training RMSE & Test RMSE with Optimized k (standardized feature)

fold	Train_rmse	Test_rmse
1	0.011213	0.014811
2	0.011041	0.014493
3	0.011284	0.011833

4	0.010894	0.015080
5	0.011290	0.011261
6	0.010972	0.015146
7	0.011256	0.012878
8	0.011050	0.014769
9	0.011286	0.012003
10	0.011123	0.014356

Next, we use the best combination to plot fitted values vs true value figure and residuals vs fitted figure, and the figures are shown below.

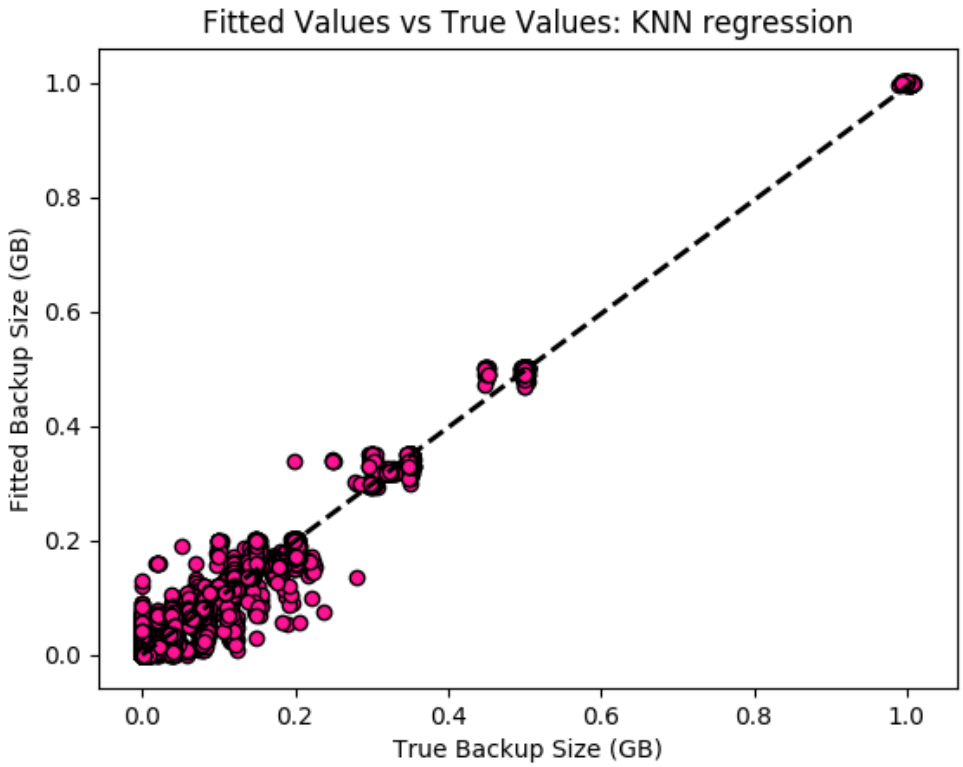


Figure 51. Fitted Values v.s. True Values with Optimized k (standardized feature)

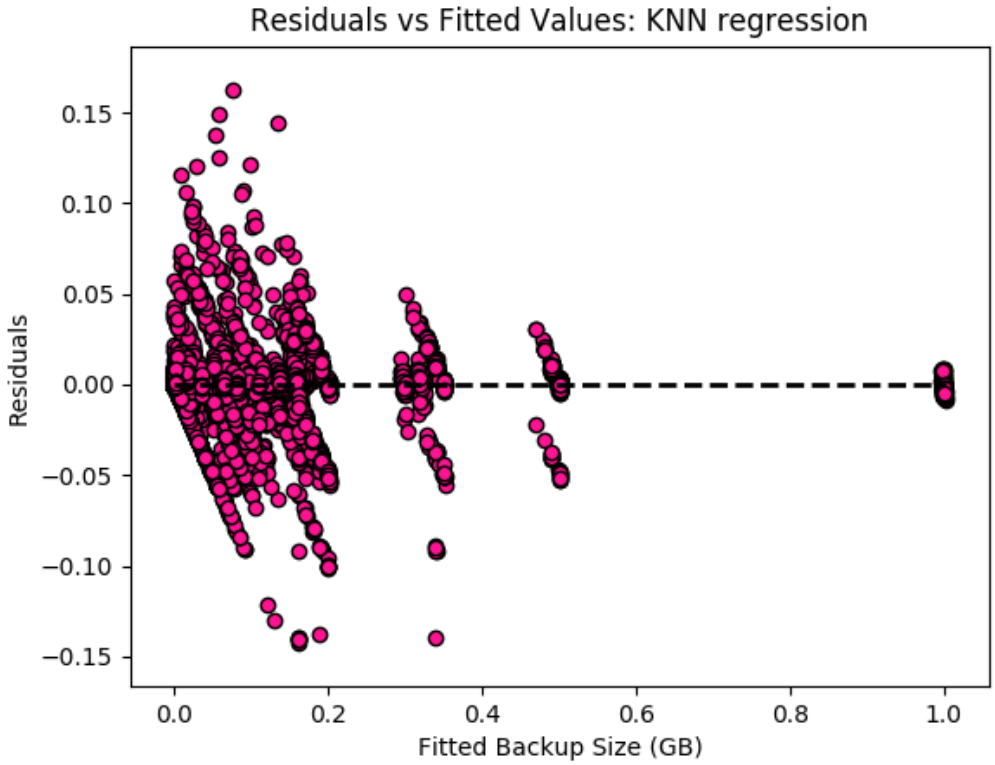


Figure 52. Residuals v.s. Fitted Values with Optimized k (standardized feature)

From above fitted values vs true value, we can observe the trend of those points follows the trend from bottom left to the upper right corner of the figure, which means the performance are good. We can also observe the points are dense at the bottom left corner and points are sparse at the top right corner, which means the fitted and true values are mostly small values. Additionally, because in the dense area the points have similar features, it is easier to predict those points with error, from residuals vs fitted values we can observe the condition and vice versa.

Question 3 Compare the Models

In this part, we compare the regression models and give out some comment. We discuss the model performance in two features, categorical features and sparse features, and also, the overall model performance.

Categorical features:

Table 10. RMSE for Categorical Feature with Regression Models

Regression Model	Linear regression (standardized)	Random Forest	KNN Regression (standardized)
Test RMSE	0.1028	0.05976	0.01366

Note:

For linear regression(standardized), I select linear regression (standardized) because it has close test RMSE to linear regression model with feature selection.

For Random Forest**,** the best combination: Number of trees=24, Number of features= 3

For KNN: I use optimal k=5 nearest neighbors.

The best model in categorical feature is k-nn regression, with optimal number of neighbors, it reaches the lowest test RMSE, which is 0.01366, much smaller than other two models. The linear regression has high test RMSE because our data's nonlinearity, so the model cannot perform well. However, for other models, like random forest, and k-nn, they do not have distribution assumption on the dataset, so it can fit complex better than linear model.

Sparse Feature:

Table 11. RMSE for Sparse Feature with Regression Models

Regression Model	Linear Regression with Optimal Feature Encoding	Linear Regression with Regularizer	NeuroNetwork Regression
Test RMSE	0.08837	0.08836	0.03500

Note:

For Linear Regression with Optimal Feature Encoding, the Optimal Combination is 22

For Linear Regression with Regularizer, I choose the best regularizer (Ridge) among Ridge, Lasso, Elastic Net regualrizer.

For Neuro Network Regression, I choose the best combination of hidden unit and activation: 130 hidden units and ReLU

The best model in sparse feature is neuro network regression, with optimal activation and hidden layer, it reaches the lowest test RMSE, which is 0.035, much smaller than other two models.

In Sparse feature, Linear Regression has test RMSE 0.08837 and with regularizer, the test RMSE improves to 0.08836, which means Ridge regularization can improve the performance of Linear regression model with feature encoding. In separate workflow regression case, work flow 2 has the best performance: in linear it has test RMSE =0.031 and in polynomial model degree = 2, it has test RMSE= 0.018. Test RMSE of work flow 3 for Linear and polynomial degree 2 model is smaller than the one of neuro network model, but for all workflow data, neuro network still perform better than Linear and polynomial model.

Overall Model Performance:

K-nn regression has lower test RMSE than neuro network regression, which means the k-nn model fits better on this data than neuro network does. However, in this project, we only use one hidden layer and hidden units smaller than 200, if we add more layer or unit and design correctly, it could perform even better than k-nn regression.