# Dataset1

## Question 1

First, we load the dataset. Figure 1 plots the back-up sizes for all workflows for first twenty-day period. Figure 2 plots for the first 105 period. From these two figures, we can see the pattern here that backup sizes for each workflow have the same pattern for every week. There is a repeating period of 7 days for all 5 workflows. The dataset does present an obvious weekly pattern on backup activities.

A close up of a map

Description automatically generated

Figure 1 Back-up sizes for all workflows for 20A close up of a logo

Description automatically generated

Figure 2 Back-up sizes for all workflows for 105

## Question 2

### Question 2(a)

Here we first fit a linear regression model. The training RMSE from 10-fold cross validation is 0.103585 and the testing RMSE 0.103676. Figure 3 and figure 4 plots fitted values against true values as scatter plots and residuals versus fitted values as scatter plots respectively.

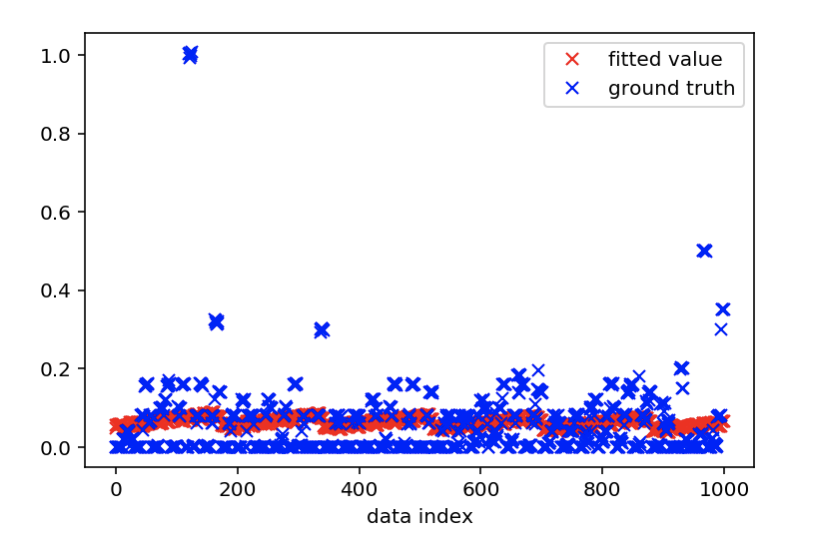


Figure 3 Fitted Values against True Values

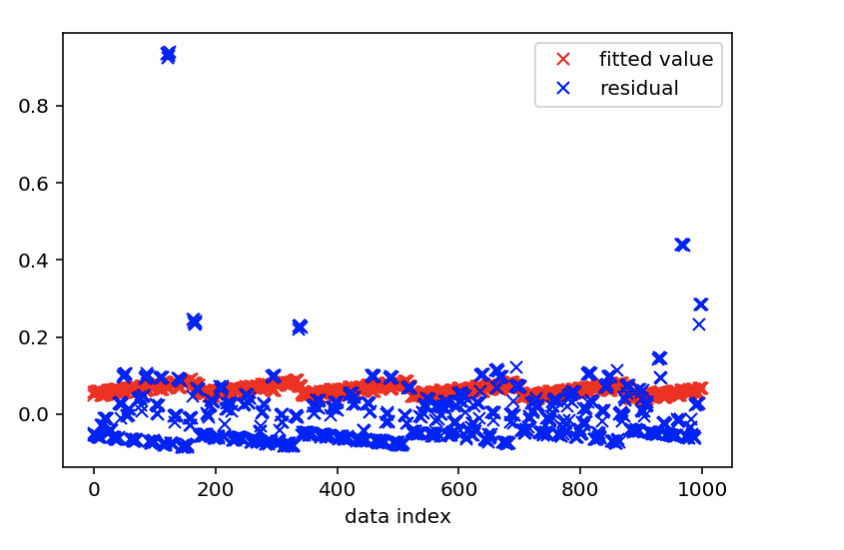


Figure 4 Residuals Versus Fitted values

### Question 2(b)

i). Now we work with random forest regression model. The training RMSE from 10-fold cross validation is 0.06069041002548773. The testing RMSE is 0.06076664492736625. The out of bag error is 0.6515227275747644.

ii). Then we sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5. Figure 5 shows the out-of-bag error (y axis) against number of trees (x axis). And figure 6 shows the average test-RMSE (y axis) against number of trees (x axis).

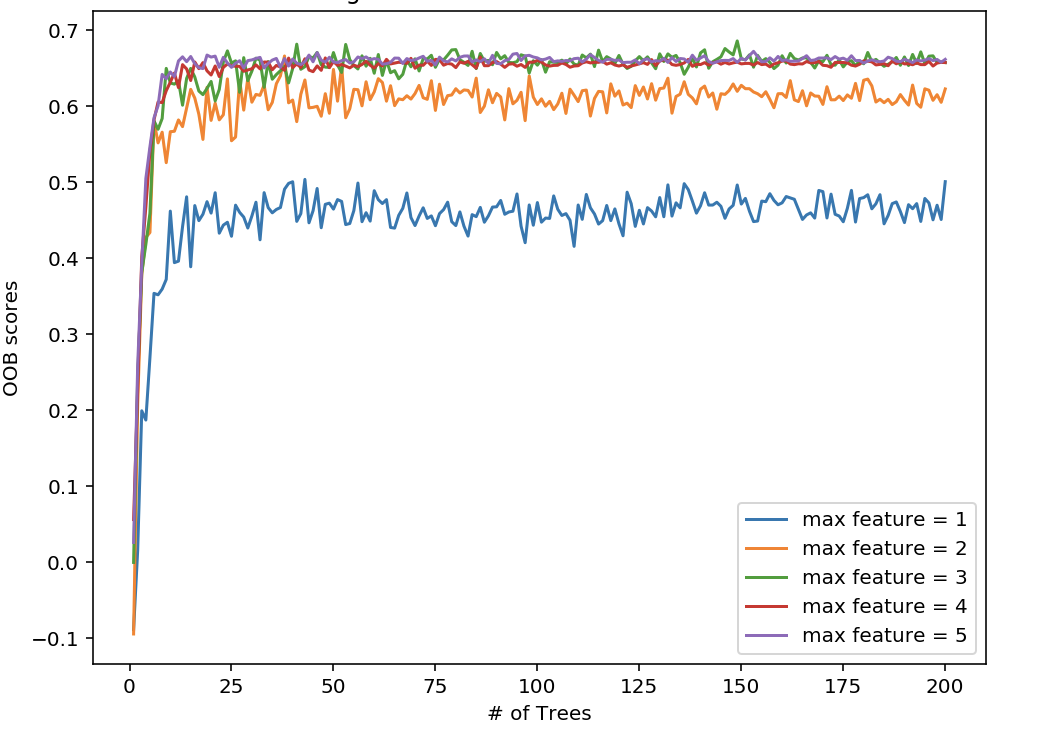


Figure 5 Out-of-bag Error against Number of Trees

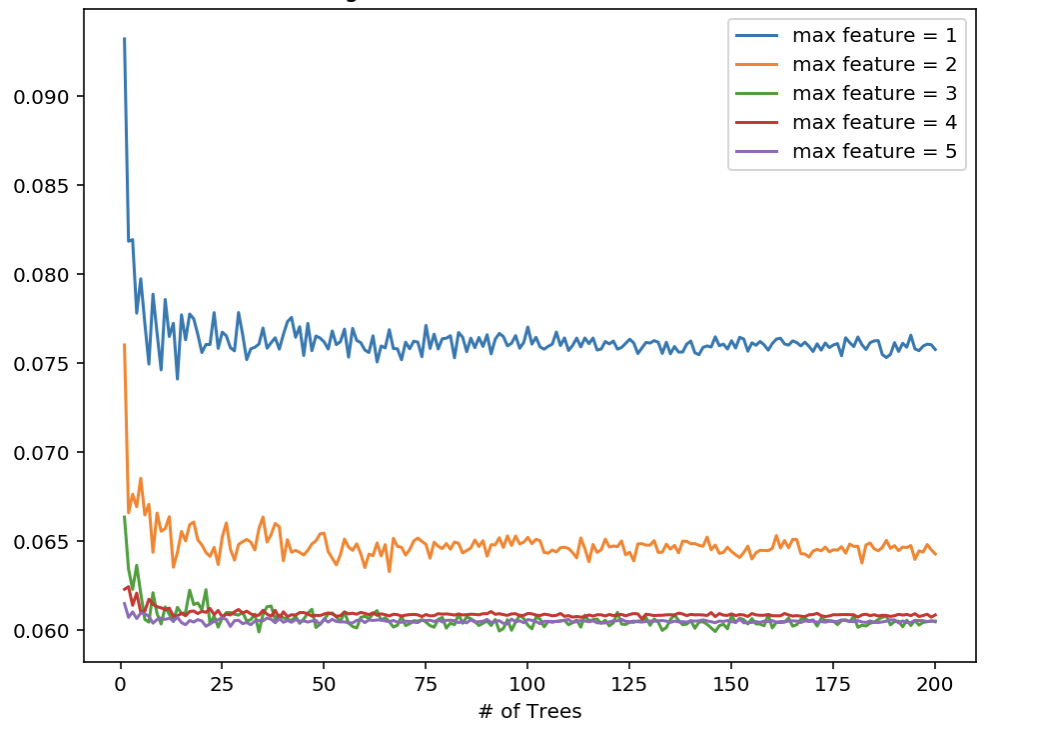


Figure 6 Average Test-RMSE against Number of Trees

iii) We reach maximal out of bag score 0.6859839618615451 at max feature = 3 and the number of trees is 149 and get the minimal test-RMSE 0.059910642274289556 at max feature = 3 and the number of trees is 34. We can conclude that the best performance happens at max feature = 3 constantly. Therefore, we set the max feature to 3 and sweep through different max tree depth value from 10 to 15 and number of trees from 10 to 100. Figure 7 shows the out-of-bag error (y axis) against number of trees (x axis). And figure 8 shows the average test-RMSE (y axis) against number of trees (x axis).

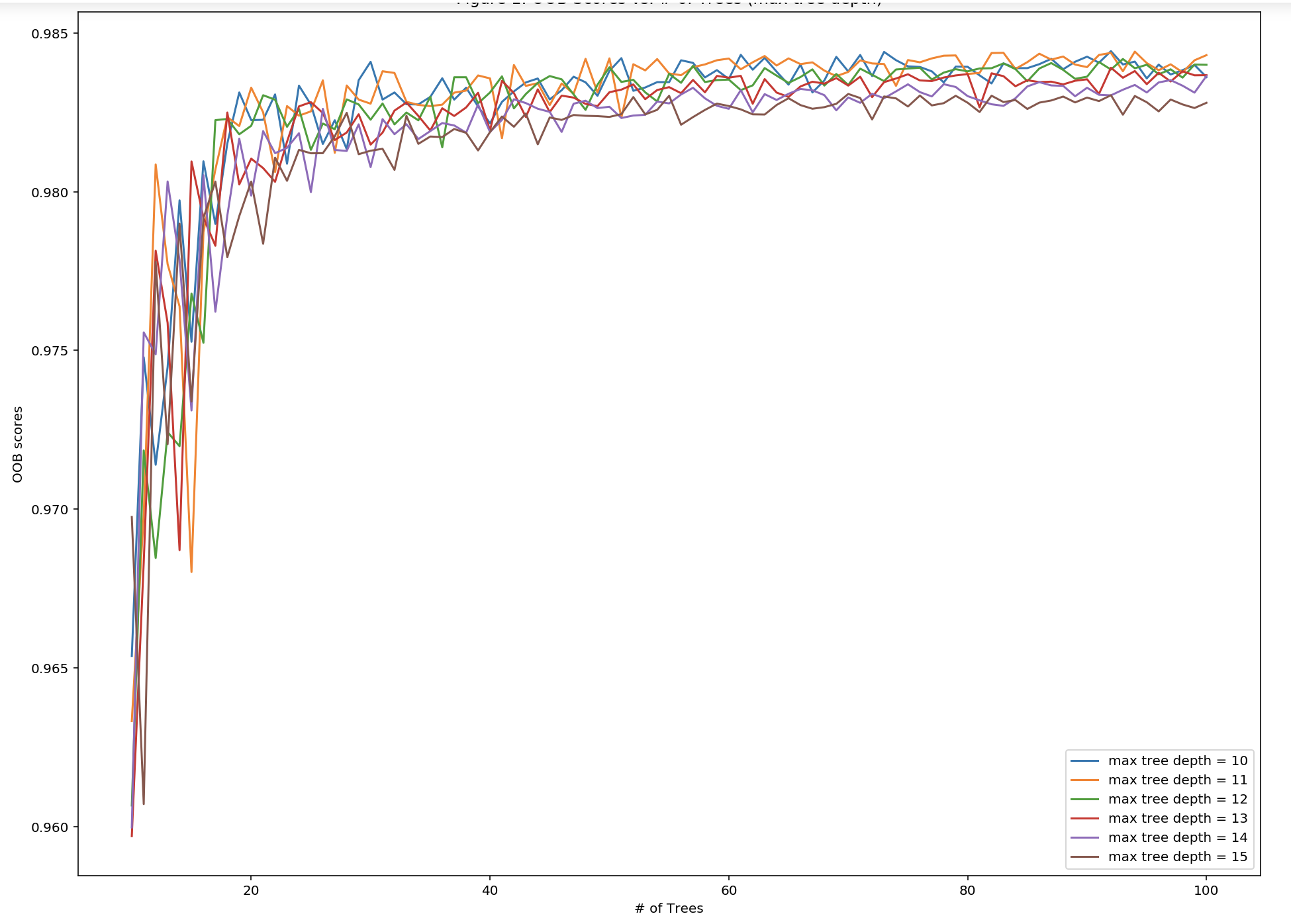


Figure 7 Out-of-bag Error against Number of Trees

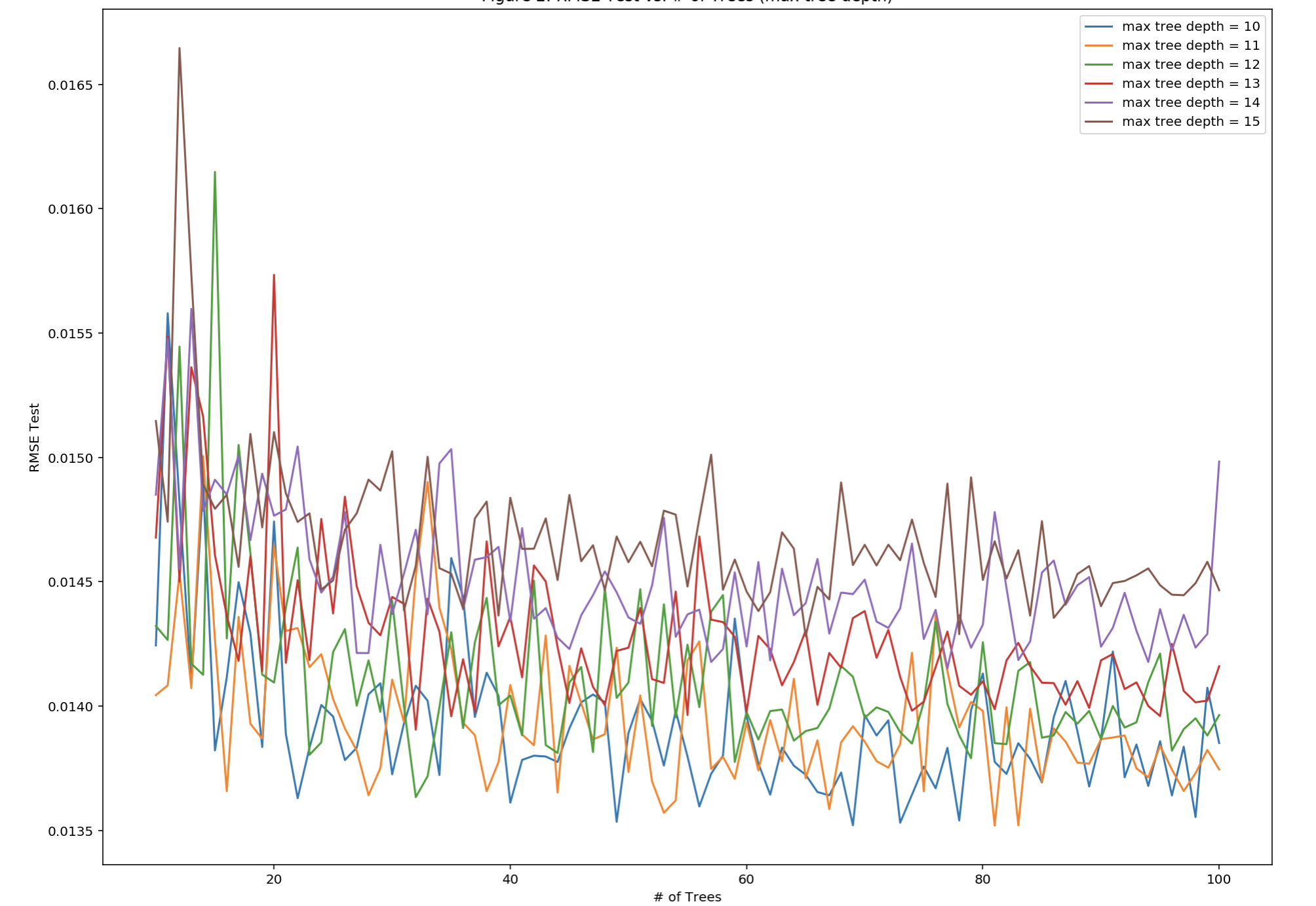


Figure 8 Average Test-RMSE against Number of Trees

Max OOB scores is 0.98442970656926 when max tree depth is 10 and number of trees is 92. Min Test RMSE is 0.0166460144101052 when max tree depth is 15 and number of trees is 12,

iv) Based on these results, we set max number of features to 3, max tree depth to 15 and number of trees to 90 for our best performance model. Min ​Test RMSE is 0.014438802408859043. Figure 9 and figure 10 plots fitted values against true values as scatter plots and residuals versus fitted values as scatter plots respectively.

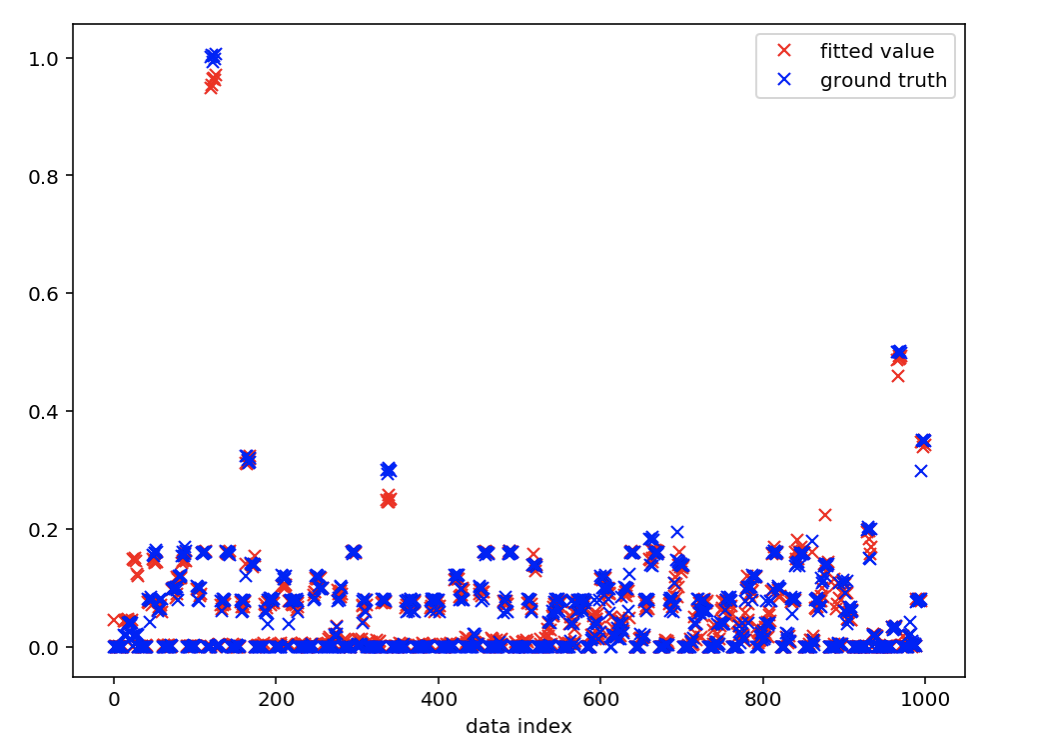


Figure 9 Fitted Values against True Values

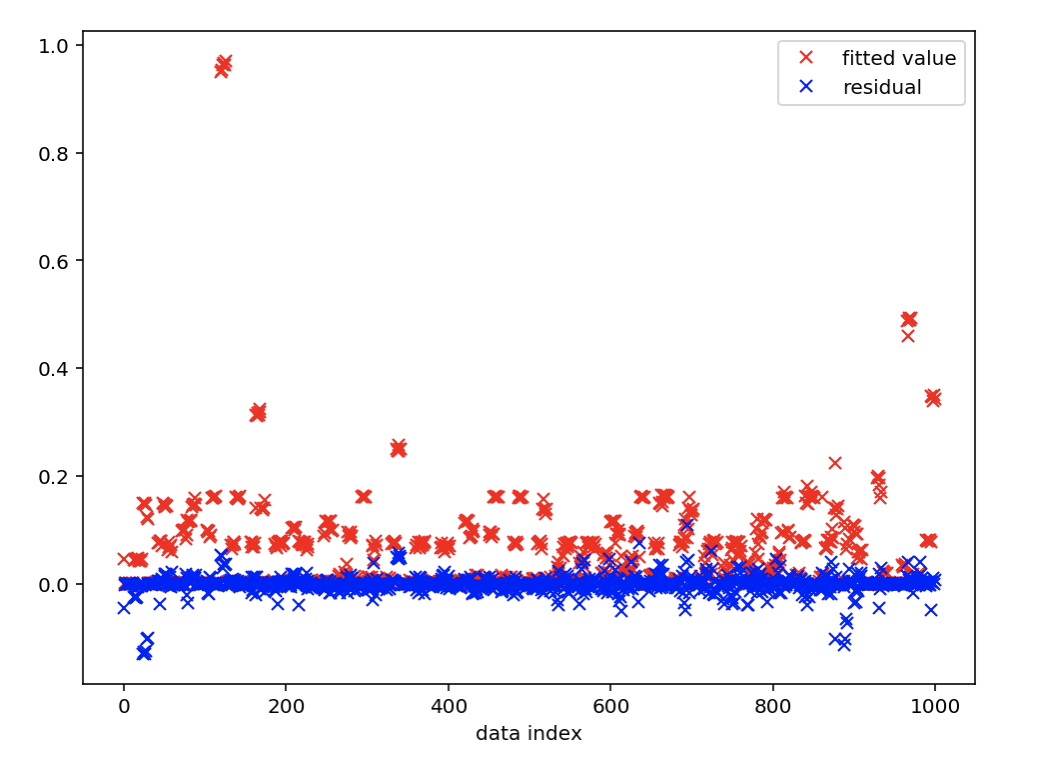


Figure 10 Residuals Versus Fitted values

Feature importance from the best model is: [0.00899751 0.29747771 0.33773083 0.17266517 0.1831287].

v) Tree visualization is shown in figure 11. All the trees in the random forest do not need to use the most important feature as the roots. The root for the tree is Root is “File Name” figure 11(a). The visualized result in figure 11(b) uses the root node “Backup Start Time - Hour of Day”, and it is the most important feature in our best model.

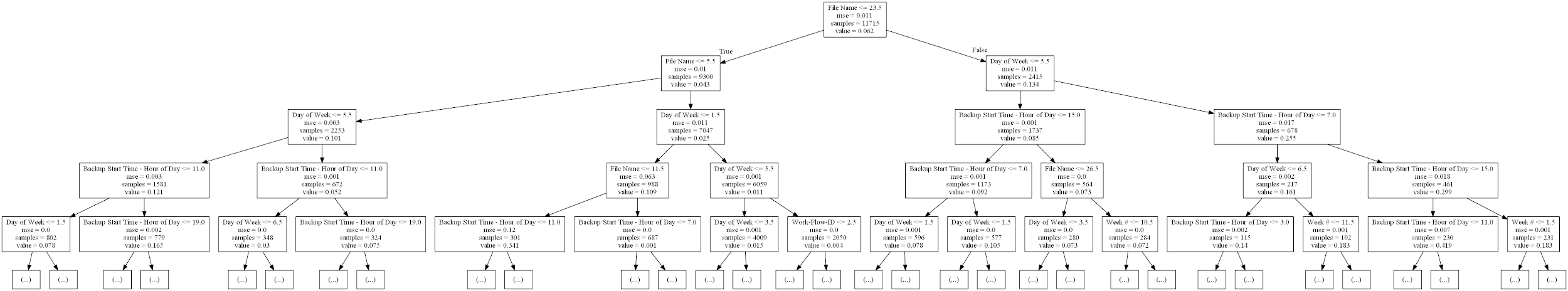


Figure 11(a) Tree Visualization

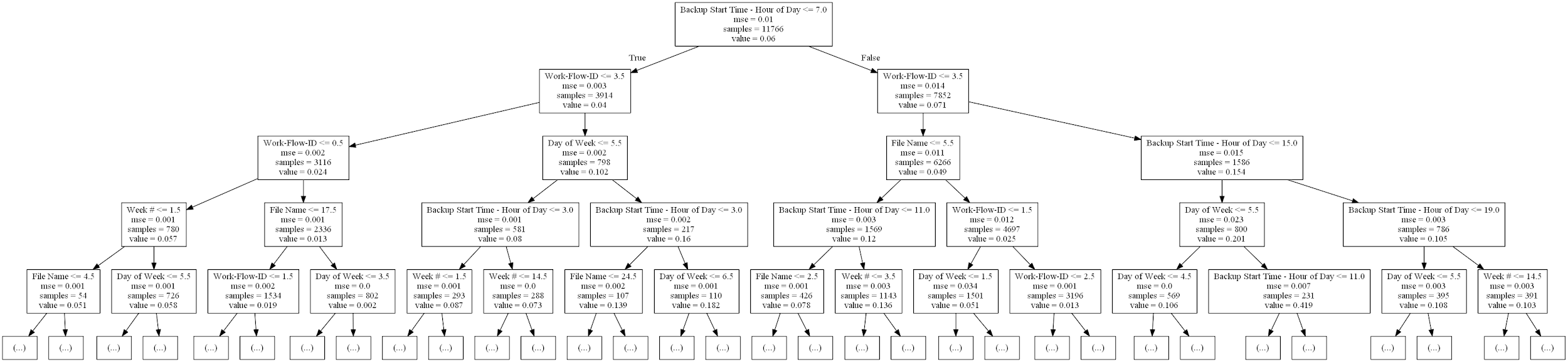


Figure 11(b) Tree Visualization

### Question 2(c)

Figure 12 shows the test-RMSE vs the number of hidden units for each activation function.

A close up of a map

Description automatically generated

Figure 12 Test-RMSE vs the Number of Hidden Units

From out training process, we found that the best combination to reach lowest test RMSE is when the hidden layer size is 195 and the activity function is tanh. The min test RMSE is 0.04399731994592147. Figure 13 and figure 14 plots fitted values against true values as scatter plots and residuals versus fitted values as scatter plots respectively.

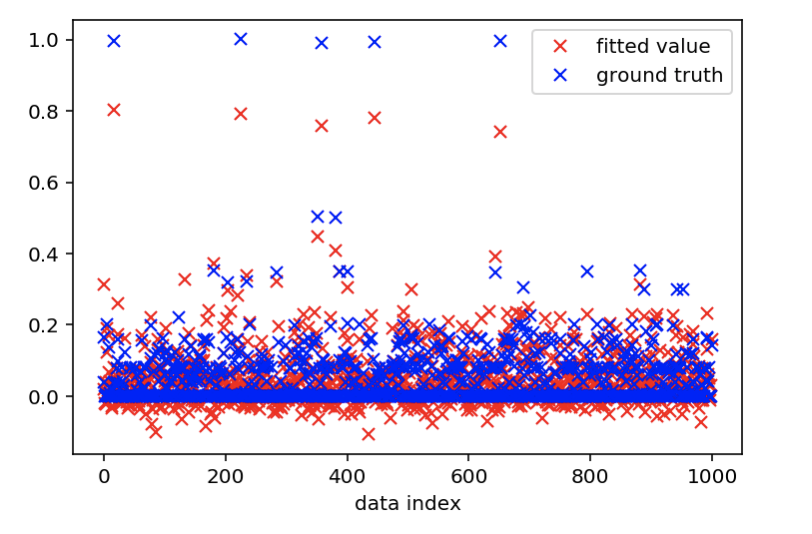


Figure 13 Fitted Values against True Values

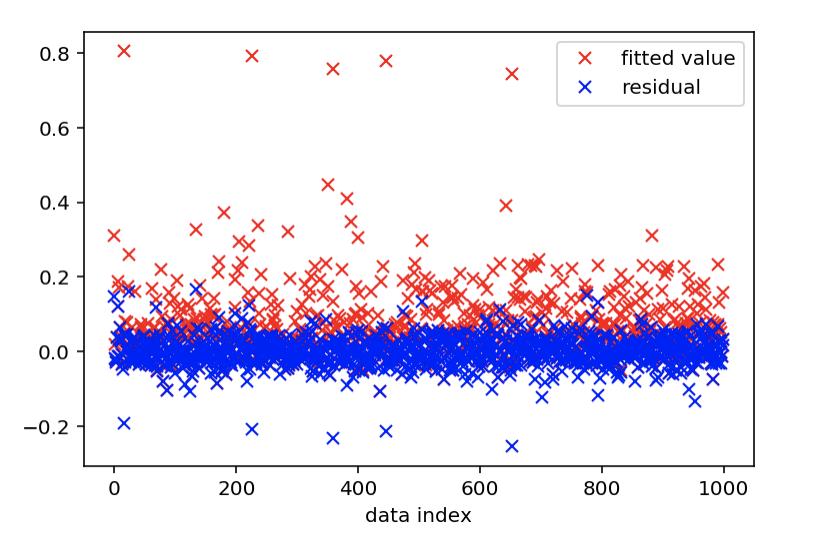


Figure 14 Residuals Versus Fitted values

### Question 2(d)

First, we uses linear regression model to predict the backup size for each of the workflows separately. The results are,

work\_flow\_0: RMSE\_train = 0.035836 RMSE\_test = 0.035887

work\_flow\_1: RMSE\_train = 0.148766 RMSE\_test = 0.148919

work\_flow\_2: RMSE\_train = 0.042909 RMSE\_test = 0.043067

work\_flow\_3: RMSE\_train = 0.007244 RMSE\_test = 0.007261

work\_flow\_4: RMSE\_train = 0.085922 RMSE\_test = 0.085991

Compared with the RMSE in question2 (a), all the models except the one of workflow 1 fit better.

Then we tried polynomial functions to improve the fit. The degrees of the polynomial we tried are from 2 to 9 (inclusive). Figure 15 and 16 shows the average train and test RMSE against the degree of the polynomial for each workflow. From this, we could conclude that for workflow 0, the best degree is 7. For workflow 1, the best degree is 9. For workflow 2, the best degree is 6. For workflow 3, the best degree is 5. For workflow 4, the best degree is 7. After these best degree, the test RMSE is getting larger. When we use polynomial function to fit the data, much high degrees are likely to lead to overfitting. It means that we get small training error but high testing error. In other words， the model lacks generality. However, we use 10-fold cross validation to mitigate this problem, thus we can find a best degree of the polynomial function with relatively lower RMSE without overfitting.

A close up of a map

Description automatically generated

Figure 15 Average Train RMSE against the Degree

A close up of a map

Description automatically generated

Figure 16 Average Test RMSE against the Degree

Then we add the RMSE for all workflow at each degree, and the result is:

[0.2742632566333011, 0.23989085320589076, 0.20280555754723684, 0.15404485000584395, 0.11682121082311742, 0.10088897078851404, 0.11884893262927397, 0.2381457498678617]

Therefore, we choose degree 7 as it reaches the minimum test RMSE. Figure 17 and figure 18 plots fitted values against true values as scatter plots and residuals versus fitted values as scatter plots respectively when degree is 7.

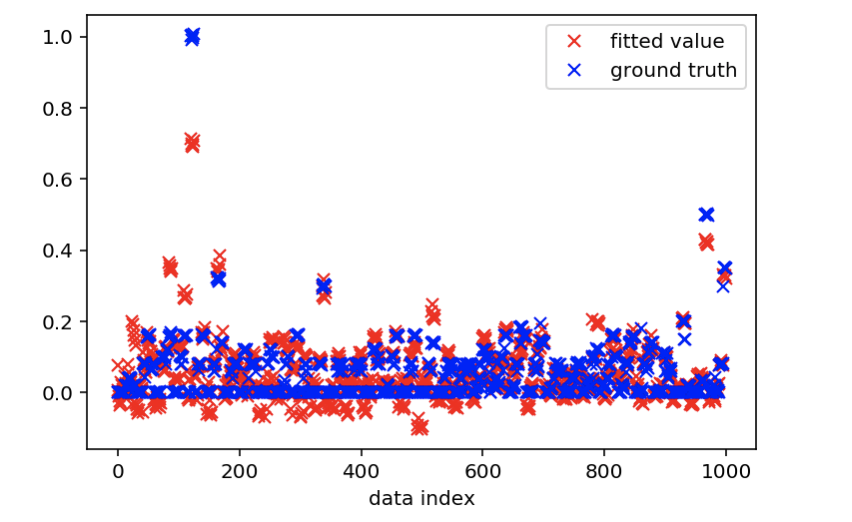


Figure 17 Fitted Values against True Values

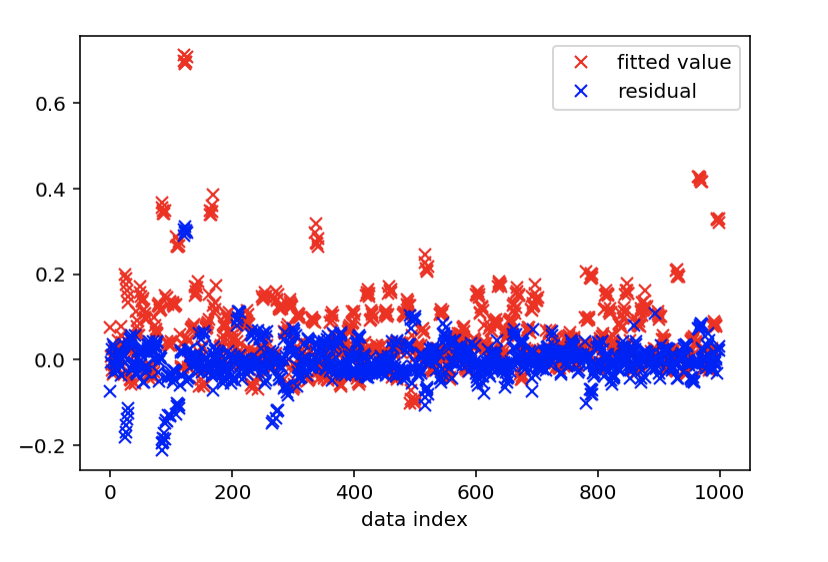


Figure 18 Residuals Versus Fitted values

### Question 2(e)

Now we use KNN. Here we try different values of K and we find that it gets the best test RMSE when k is 5. And the test RMSE is 0.03505389720056441.

rmse\_train: 0.018395953497860964 rmse\_test: 0.061895994437848406

rmse\_train: 0.02964644147913671 rmse\_test: 0.0429609569810752

rmse\_train: 0.029390321261297918 rmse\_test: 0.03689983416264968

rmse\_train: 0.027993827879178014 rmse\_test: 0.035308581099246235

rmse\_train: 0.027469757999663547 rmse\_test: 0.03505389720056441

rmse\_train: 0.02970683407352238 rmse\_test: 0.03824456254773458

rmse\_train: 0.03328090731956981 rmse\_test: 0.04271716505395125

rmse\_train: 0.037249305618435535 rmse\_test: 0.04251009355462779

rmse\_train: 0.03982541799582406 rmse\_test: 0.04840877808253201

rmse\_train: 0.04179583938765732 rmse\_test: 0.048088173515841715

rmse\_train: 0.04372205621491336 rmse\_test: 0.04941767025659589

rmse\_train: 0.045153129052575035 rmse\_test: 0.05321103081222519

rmse\_train: 0.0465624678953089 rmse\_test: 0.05329370653245115

rmse\_train: 0.048515321958958274 rmse\_test: 0.055060590157674304

rmse\_train: 0.050078985477129634 rmse\_test: 0.05489521168379978

rmse\_train: 0.0512417205674884 rmse\_test: 0.055996568644377874

rmse\_train: 0.05159035186474146 rmse\_test: 0.053905760015369064

rmse\_train: 0.051934562557542384 rmse\_test: 0.05300885318062385

rmse\_train: 0.05192335336575136 rmse\_test: 0.054173404951373516

rmse\_train: 0.05204811491403675 rmse\_test: 0.054111771067971236

rmse\_train: 0.05230068651001267 rmse\_test: 0.05560674669957599

rmse\_train: 0.052879732196522165 rmse\_test: 0.05920876174002219

rmse\_train: 0.05390591074677592 rmse\_test: 0.05410047239586326

rmse\_train: 0.05417886637740878 rmse\_test: 0.055655531458769655

rmse\_train: 0.05460125066881737 rmse\_test: 0.0547890888746814

rmse\_train: 0.05463063209646215 rmse\_test: 0.05736708333623343

rmse\_train: 0.05484639186580033 rmse\_test: 0.0540463396843488

rmse\_train: 0.05472332010423475 rmse\_test: 0.05664962095664979

rmse\_train: 0.05452665288018339 rmse\_test: 0.0570440302951612

rmse\_train: 0.0549285947236401 rmse\_test: 0.0555248500899867

rmse\_train: 0.05509011848463018 rmse\_test: 0.057570036125508164

rmse\_train: 0.05532774923961524 rmse\_test: 0.058972590167942006

rmse\_train: 0.05611420609956791 rmse\_test: 0.05555388110835513

rmse\_train: 0.05634206437714941 rmse\_test: 0.058492838320569104

rmse\_train: 0.05694459983635428 rmse\_test: 0.06009072762617733

rmse\_train: 0.05746778395888989 rmse\_test: 0.058211831388971035

rmse\_train: 0.058112828907641766 rmse\_test: 0.06007052190477481

rmse\_train: 0.05879423956666987 rmse\_test: 0.059529005614337674

rmse\_train: 0.059259336367679286 rmse\_test: 0.062428232258226225

rmse\_train: 0.06001693389924675 rmse\_test: 0.058718520841075206

rmse\_train: 0.060292131277255634 rmse\_test: 0.06296881909439728

rmse\_train: 0.06085184655388535 rmse\_test: 0.06320296088958896

rmse\_train: 0.06117071427784533 rmse\_test: 0.06528625302262818

rmse\_train: 0.06171014636578297 rmse\_test: 0.061607013394890865

rmse\_train: 0.062015298740194943 rmse\_test: 0.06489062427332797

rmse\_train: 0.062462733774756196 rmse\_test: 0.06361177888229622

rmse\_train: 0.0626348816811648 rmse\_test: 0.06293194357649677

rmse\_train: 0.0627636635654317 rmse\_test: 0.06373771059226976

rmse\_train: 0.06304321990226046 rmse\_test: 0.06470260366927452

rmse\_train: 0.06309643343558523 rmse\_test: 0.06491193654679031

rmse\_train: 0.06372397994674954 rmse\_test: 0.06102481955934482

rmse\_train: 0.0634795536163374 rmse\_test: 0.06671068749162598

rmse\_train: 0.06351182511575316 rmse\_test: 0.06861650015947202

rmse\_train: 0.06395468572853183 rmse\_test: 0.0675284973593489

rmse\_train: 0.06437525083223253 rmse\_test: 0.06507310121208518

rmse\_train: 0.06475676420592325 rmse\_test: 0.06351543083722917

rmse\_train: 0.06478408029541531 rmse\_test: 0.06609757801637287

rmse\_train: 0.06495041511841262 rmse\_test: 0.06781675142265264

rmse\_train: 0.06552488403271658 rmse\_test: 0.06505264502281193

rmse\_train: 0.0656212037670044 rmse\_test: 0.06756176872672223

rmse\_train: 0.06579733522325718 rmse\_test: 0.06735148708951784

rmse\_train: 0.06598555018902123 rmse\_test: 0.06614531583691398

rmse\_train: 0.0661528856573966 rmse\_test: 0.06803934620182907

rmse\_train: 0.06613909096196993 rmse\_test: 0.07048150458366914

rmse\_train: 0.06658545726366911 rmse\_test: 0.06633222121383332

rmse\_train: 0.06646431079060847 rmse\_test: 0.06826428823688371

rmse\_train: 0.0666599206722235 rmse\_test: 0.06881380554211843

rmse\_train: 0.06706642026009008 rmse\_test: 0.06525028080137547

rmse\_train: 0.06710919544159413 rmse\_test: 0.06616844432190126

rmse\_train: 0.06712502284692581 rmse\_test: 0.06677045159773141

rmse\_train: 0.06707731308317247 rmse\_test: 0.06795597242017418

rmse\_train: 0.06714485081481783 rmse\_test: 0.06927756110433496

rmse\_train: 0.0672076505646587 rmse\_test: 0.07006668835567889

rmse\_train: 0.06751231800657809 rmse\_test: 0.06791439610430655

rmse\_train: 0.06778490279929737 rmse\_test: 0.06516146166373042

rmse\_train: 0.06753893674963742 rmse\_test: 0.0696880769547683

rmse\_train: 0.0678021523923474 rmse\_test: 0.06680955610623115

rmse\_train: 0.06792612623514198 rmse\_test: 0.06713872821196375

rmse\_train: 0.06798384374130181 rmse\_test: 0.06816959368840024

rmse\_train: 0.06802442163342276 rmse\_test: 0.06896920740356716

rmse\_train: 0.06776587235422564 rmse\_test: 0.0743947249426771

rmse\_train: 0.06855596751364797 rmse\_test: 0.06615035706068187

rmse\_train: 0.06860647426698889 rmse\_test: 0.06853301259911661

rmse\_train: 0.06868294344325523 rmse\_test: 0.06817944680026972

rmse\_train: 0.06869255196122229 rmse\_test: 0.0710855524449942

rmse\_train: 0.06926020777871356 rmse\_test: 0.06730997630756626

rmse\_train: 0.06939234791101544 rmse\_test: 0.0670443646721359

rmse\_train: 0.06953626012545674 rmse\_test: 0.06886549410434584

rmse\_train: 0.06953103422523949 rmse\_test: 0.07203701178806429

rmse\_train: 0.0700339242524808 rmse\_test: 0.06910819017460892

rmse\_train: 0.07013530561025434 rmse\_test: 0.0690025478802459

rmse\_train: 0.07029611769490421 rmse\_test: 0.06937865401487363

rmse\_train: 0.0704290416089062 rmse\_test: 0.06975943813542355

rmse\_train: 0.0705094155513655 rmse\_test: 0.07084492634896708

rmse\_train: 0.07069098205985955 rmse\_test: 0.0704891078111007

rmse\_train: 0.07079974730129289 rmse\_test: 0.07001584073254714

rmse\_train: 0.07077609564322616 rmse\_test: 0.07173380715140379

rmse\_train: 0.07080959942885924 rmse\_test: 0.0726159301565258

rmse\_train: 0.07093350951477034 rmse\_test: 0.07121078756790576

min test rmse: 0.03505389720056441 at neighbour size: 5

Figure 19 and figure 20 plots fitted values against true values as scatter plots and residuals versus fitted values as scatter plots respectively at the best time.

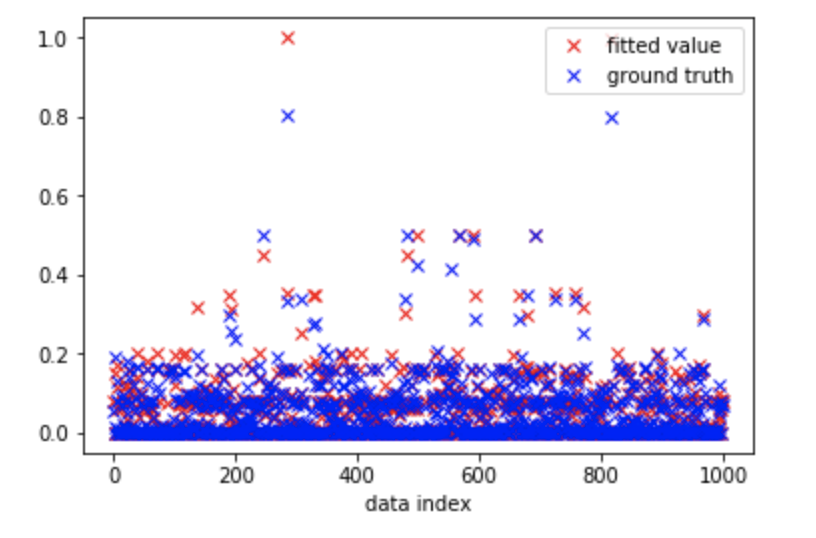


Figure 19 Fitted Values against True Values

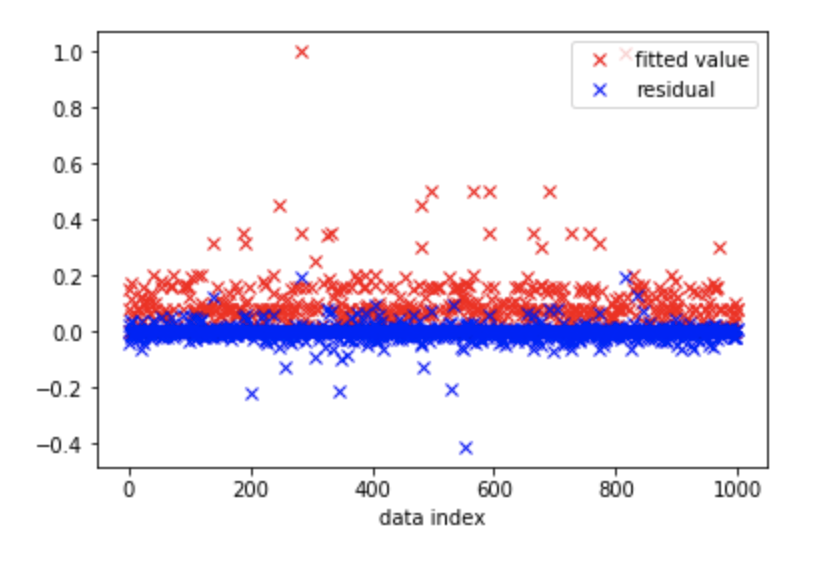


Figure 20 Residuals Versus Fitted values

## Question 3

In this project, we try linear regression model, random forest, neural network and KNN for categorical data.

Linear Regression model cannot accept categorical inputs, so we have to map those categorical data into numerical data first. There are two ways to do so, one is scalar encoding and the other is one-hot encoding. In our experiments, we use scalar encoding. linear model is great when the relationship is more like linear. The performance varies a lot with respect to the work flow we work on. By adding the degree could improve the performance. However, when choose some degree that is too large, it may cause overfitting, leading to the back performance on generality. Linear Regression could handle sparse inputs by adding regularization, which we did not dig deeper in dataset 1.

Random forest model is the best for handling categorical data. ​It achieves a very good test RMSE of 0.014. It could effectively reduce the variance of result and evaluate the importance of each features. The type of input is not a kind of constraint, so we do not need to do any encoding here. The downside is that it is comparatively hard to interpret the result visually. Also, it is not very suitable for sparse dataset. Bagging and suboptimal selection​ of splits can waste Random Forest model’s insight on zero-only areas of data.

Neural Network cannot directly handle categorical values, so we use the one-hot-encoding here. It works exceptionally good sometimes with fine tuning. However, it takes great time and memory during the training process. Also, the choice of hyperparameters is really important and there are several hyperparameters that we need to tune in order to achieve the best performance. Neural Network will not perform very well for sparse data due to the active function. For step-like activity function such as sigmoid, tanh, they are continuous function and map the raw data into the range of 0 and 1 or the range of -1 and 1. Thus, most of the data will be mapped to the end of the range when the input values are sparse.

KNN is not the best choice when dealing with categorical data the “meaningful” distance function for KNN is also very subtle to find. KNN is the one of the most straightforward models we tried in this project. It is easy to implement and sometimes has great effect on multi-variables regression. However, it takes long time to do every prediction and hard to deal with extreme huge dataset in the real world. Also, it does not work well for high-dimensional data.

When it comes to sparse features, KNN cannot work well as highly sparse data tend to have high dimensions and it brings much difficult in defining and calculating the distance quickly.