EE-219 Project 4

Project 4: Regression Analysis

March 04, 2018

Ashish Shah (804946005)
Ayush Dattagupta (305024749)
Shrey Agarwal (004943082)
Varun Saboo (505028591)

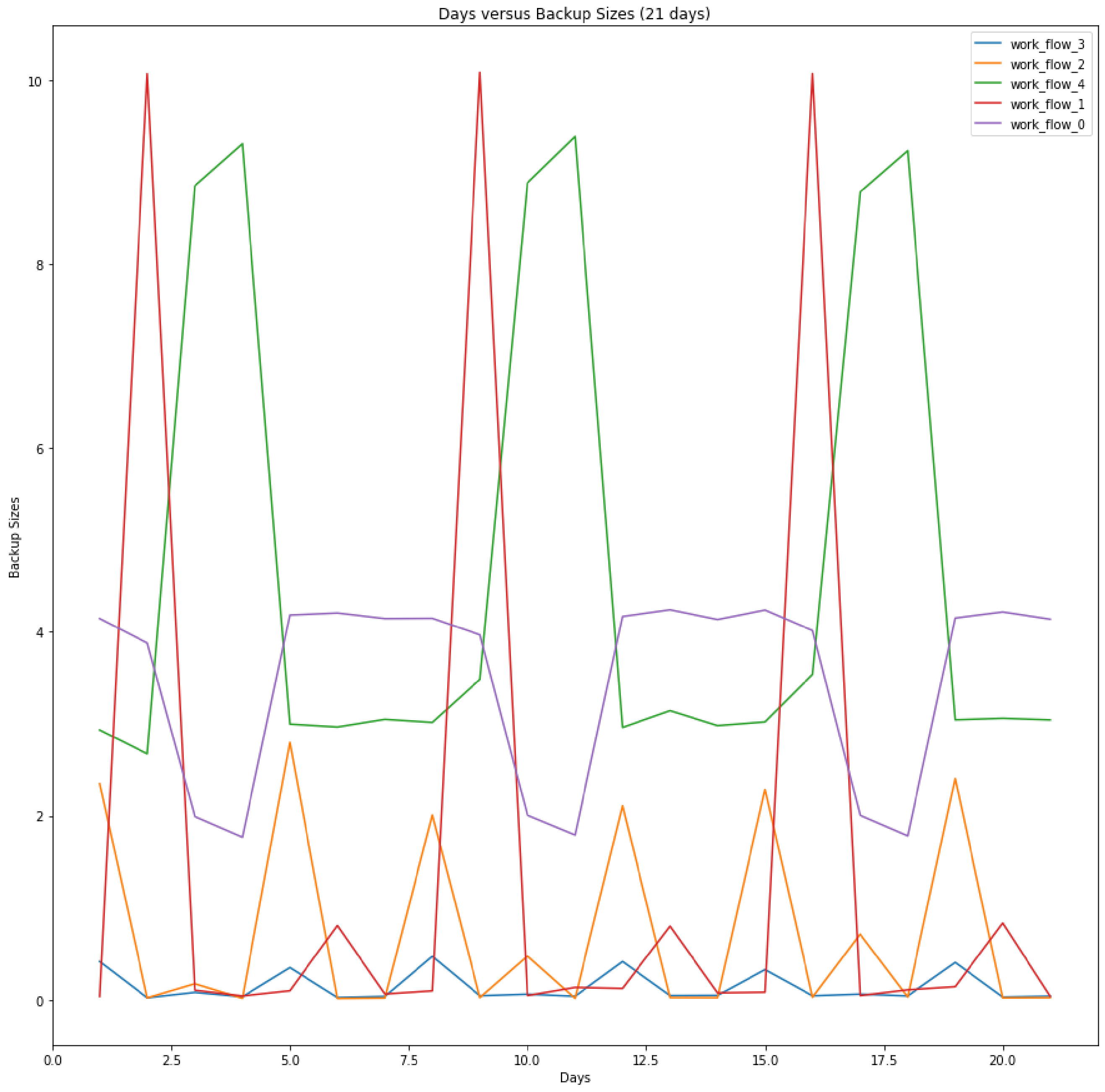
Objective –
In this project, we explore
basic regression models on a given dataset, along with basic techniques to handle over-fitting; namely cross-validation, and regularization. With cross-validation, we test for overfitting, while with regularization we penalize overly complex models.

Dataset –
We use a Network backup Dataset, which is comprised of simulated trac data on a backup system over a network. The system monitors the les residing in a destination machine and copies their changes in four hour cycles. At the end of each backup process, the size of the data moved to the destination as well as the duration it took are logged, to be used for developing prediction models. We define a workflow as a task that backs up data from a group of les, which have similar patterns of change in terms of size over time. The dataset has around 18000 data points with the following columns/variables:

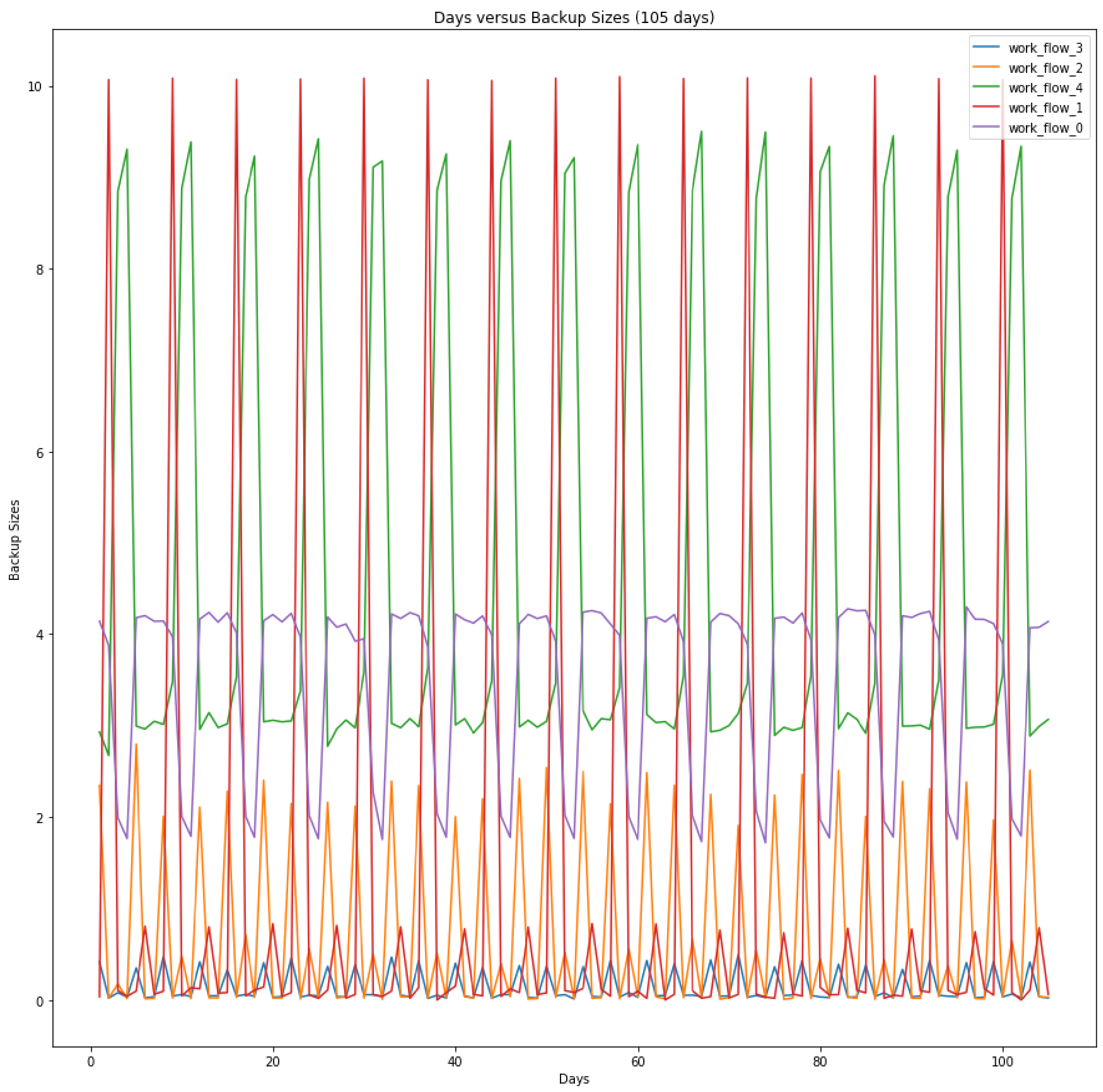
- Week index
- Day of the week at which the le back up has started
- Backup start time: Hour of the day
- Workflow ID
- File name
- Backup size: the size of the le that is backed up in that cycle in GB
- Backup time: the duration of the backup procedure in hour

Question 1 - Visualization:

Load the dataset and plot backup size vs workflows for 20 days and 105 days:



The above figure clearly shows us that there is a pattern in the way each workflow saves its backups and the backup sizes have different periodicity based on different workflows.
The backup sizes of each workflow increases on certain days of the week and the frequency of backup for each workflow is the same.



We can observe that the patterns continue to follow for 105 days as we saw in 20 days initially. This shows us that the workflows are consistent.

Question 2(a): Linear Regression

As mentioned in the instructions, we create 32 datasets for the linear regression part of the project.

I. Baseline Linear Regression

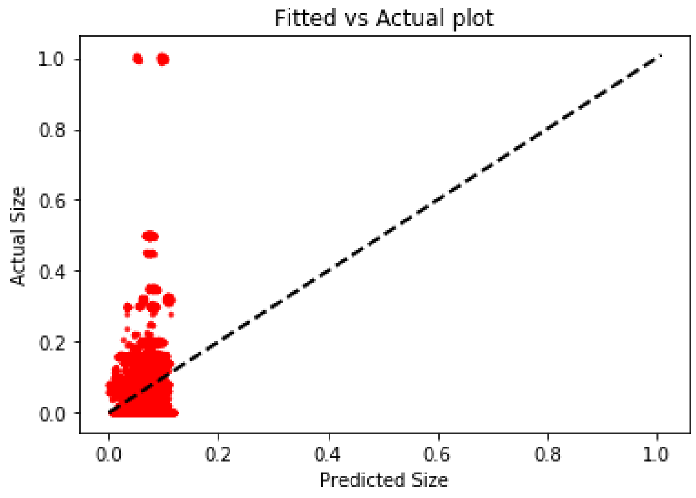
I the first part, we create a baseline regression model in which we only consider the case of the first dataset (i.e. the one with all categorical features expressed in terms of scalar encoding). The results for this model is as follows:

```
[fold 0] Train_RMSE: 0.10271, Test_RMSE: 0.09370
[fold 1] Train_RMSE: 0.10376, Test_RMSE: 0.08262
[fold 2] Train_RMSE: 0.10171, Test_RMSE: 0.10301
[fold 3] Train_RMSE: 0.10192, Test_RMSE: 0.10116
[fold 4] Train_RMSE: 0.10160, Test_RMSE: 0.10396
[fold 5] Train_RMSE: 0.10327, Test_RMSE: 0.08797
[fold 6] Train_RMSE: 0.10047, Test_RMSE: 0.11347
[fold 7] Train_RMSE: 0.10047, Test_RMSE: 0.11343
[fold 8] Train_RMSE: 0.10151, Test_RMSE: 0.10476
[fold 9] Train_RMSE: 0.10090, Test_RMSE: 0.10995
Average Train RMSE = 0.10183
Average Test RMSE = 0.1014
```

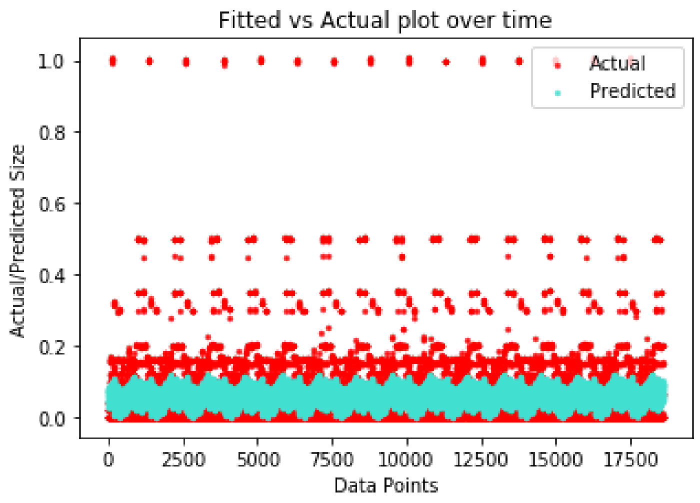
The coefficients for the model are as follows:

```
Coefficients =  {'Week': 2.0000000000000002e-05, 'BackupStart': 0.0054900000000
000001, 'FileType': 0.0018500000000000001, 'WFID': 0.0018500000000000001, 'Da
y': -0.0056800000000000002}
Intercept = 0.03367
```

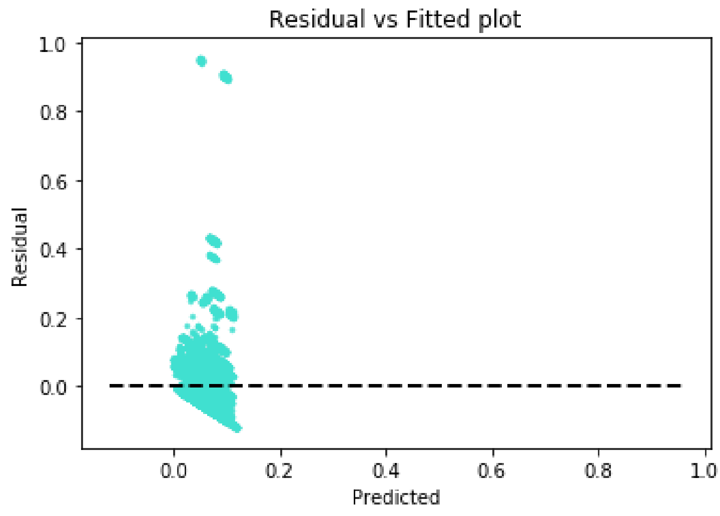
As we can observe from this, the model places least emphasis on week and more emphasis on BackUpStart and Day due to the periodic nature of the backup of the workflows.
The scatter plot is as follows:

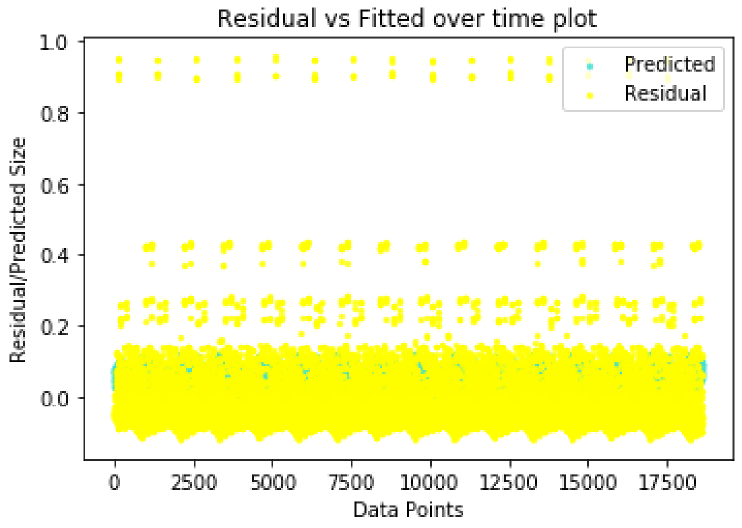


From the above graph, we can visualize how the data was predicted. Clearly, the baseline linear regression does not provide the best fitting because of the deviation of red points from the dotted line.



This plot shows the actual and predicted size for each data point separately. The majority of the predictions are within the range of 0 to 0.175.





From the residual graph, it is evident that while a lot of the points are close to the residual=0 line, there are many outliers that the model hasn't learnt to predict. And because of this, the performance of the linear model is not good.

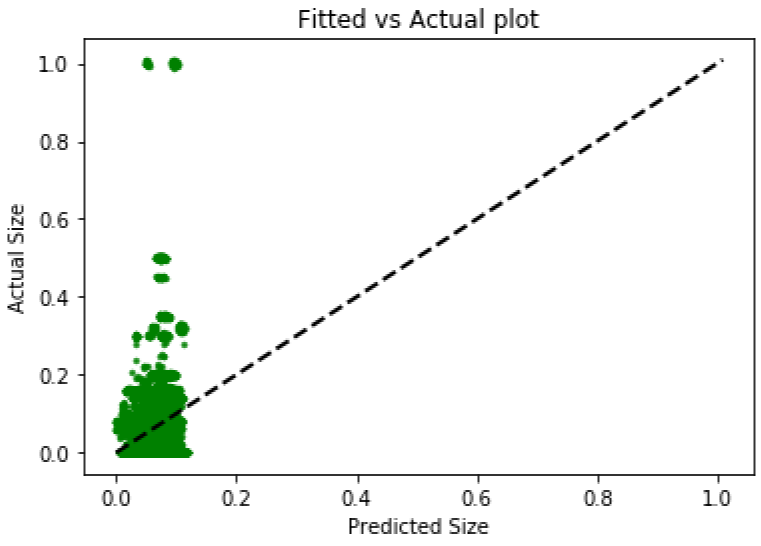
II. Standardization of Data

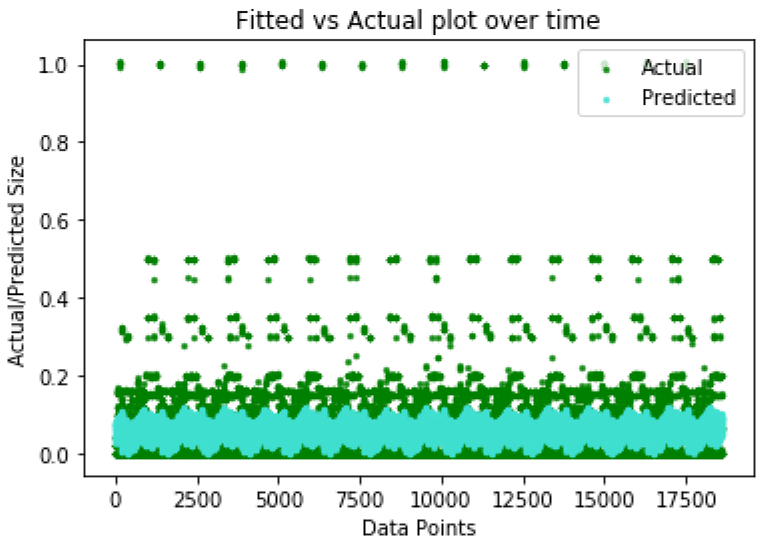
In the second stage, we first preprocess the data with the StandardScalar before learning the regression model for it. For this part the dataset is the same as the one used in the previous part. The results are as follows:

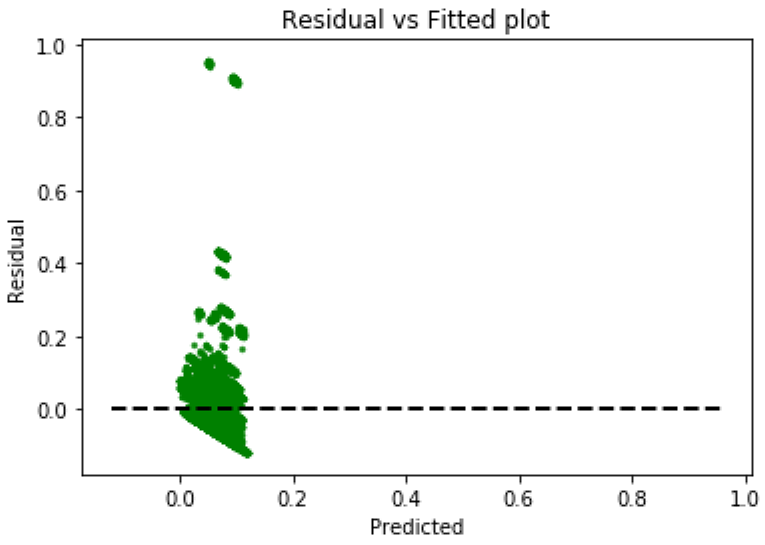
```
[fold 0] Train_RMSE: 0.10220, Test_RMSE: 0.09855
[fold 1] Train_RMSE: 0.10192, Test_RMSE: 0.10111
[fold 2] Train_RMSE: 0.10136, Test_RMSE: 0.10611
[fold 3] Train_RMSE: 0.10210, Test_RMSE: 0.09947
[fold 4] Train_RMSE: 0.10167, Test_RMSE: 0.10336
[fold 5] Train_RMSE: 0.10171, Test_RMSE: 0.10304
[fold 6] Train_RMSE: 0.10219, Test_RMSE: 0.09865
[fold 7] Train_RMSE: 0.10181, Test_RMSE: 0.10213
[fold 8] Train_RMSE: 0.10143, Test_RMSE: 0.10543
[fold 9] Train_RMSE: 0.10198, Test_RMSE: 0.10053
Average Train RMSE = 0.10184
Average Test RMSE = 0.10184
```

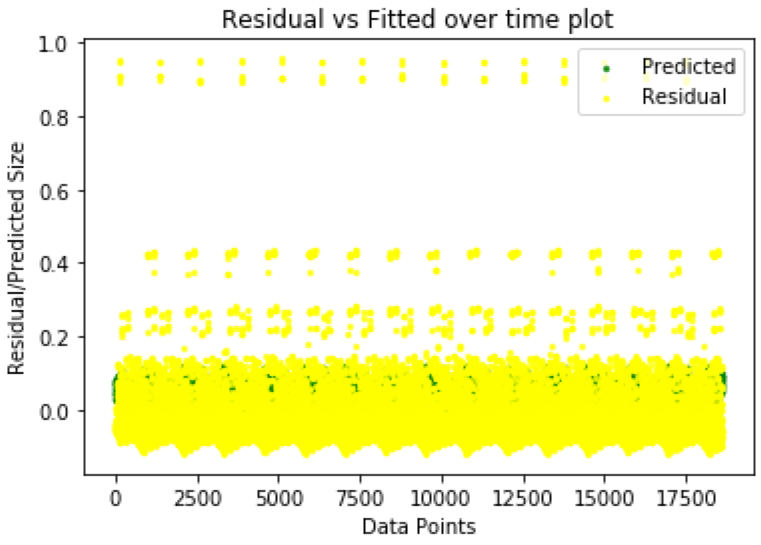
```
Coefficients =  {'Week': 6.9999999999999994e-05, 'BackupStart': 0.0093900000000
000008, 'FileType': 0.01593, 'WFID': 0.0026199999999999999, 'Day': -0.011350000
000000001}
Intercept = 0.06098
```

As we can see from these numbers, the results are almost exactly the same from the previous model. This is because the model is invariant to normalization.The only clear difference in this model is the higher coefficient for BackUpStart indicating the importance of that feature. Even the following graphs look the same.




Fitted vs Actual plot over time
Actual/Predicted Size
Data Points
Actual
Predicted
1.0
0.8
0.6
0.4
0.2
0.0
0
2500
5000
7500
10000
12500
15000
17500

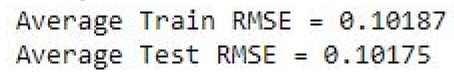

Residual vs Fitted plot
Residual
Predicted
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Residual vs Fitted over time plot
Residual/Predicted Size
Data Points
Predicted
Residual
1.0
0.8
0.6
0.4
0.2
0.0
0
2500
5000
7500
10000
12500
15000
17500

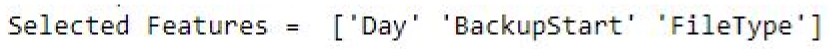
The fitting does not change due to the invariant nature of the model to transformations, and hence the graphs look the same.

III. Feature Selection

F-Regression



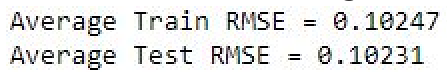
```
Average Train RMSE = 0.10187
Average Test RMSE = 0.10175
```



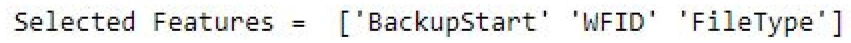
```
Selected Features =  ['Day' 'BackupStart' 'FileType']
```

The results above are for f_regression. The selected features were Day, BackupStart and Filetype. These are periodic attributes and hence are more important to learn to predict the backup size.
Using only these important features, the results do not improve, but they also do not worsen. Thus highlighting that instead of having all features, these three features were sufficient to obtain the same result.

Mutual Information Regression



```
Average Train RMSE = 0.10247
Average Test RMSE = 0.10231
```



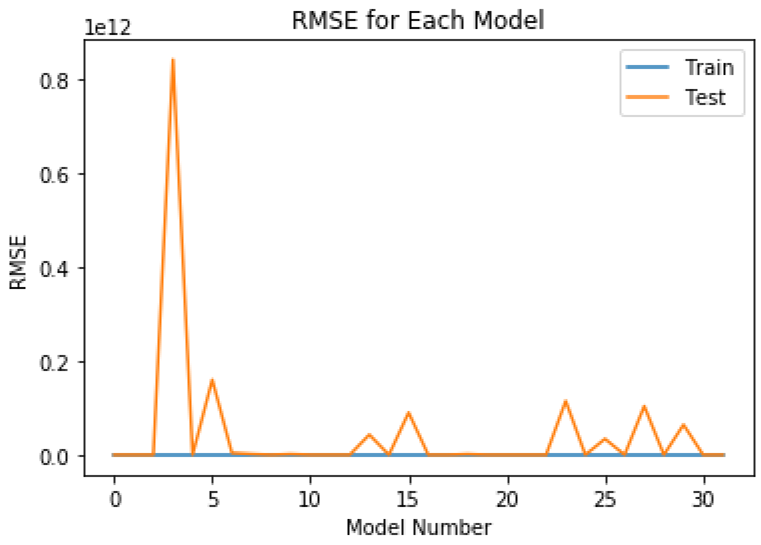
```
Selected Features =  ['BackupStart' 'WFID' 'FileType']
```

Mutual information regression selects BackUpStart, filetype and WFID as the most important features. However, when comparing the performance with f_regression, the model performs poorly as the error increases marginally by 0.001

F-Regression captures only linear dependency between the features. Hence, it produces better results for linear regression. Mutual information captures any sort of dependency, linear or non-linear and hence might produce better results on a nonlinear model.

IV. Feature Encoding

With shuffle=False in KFold CV



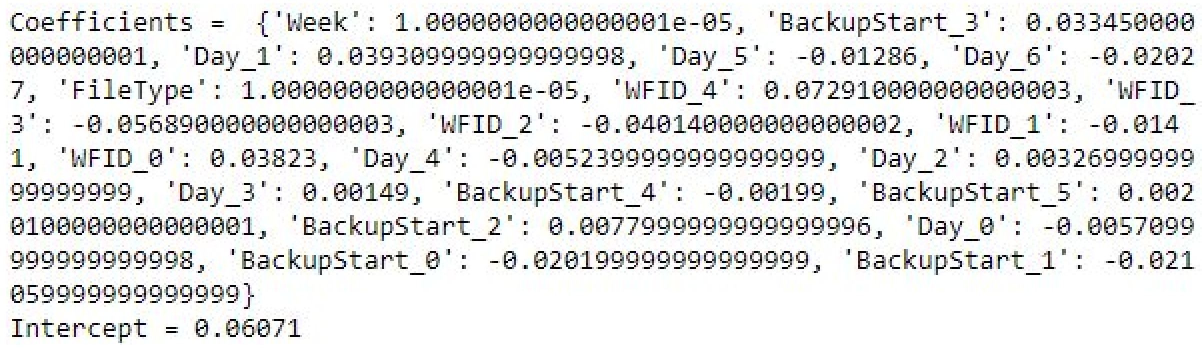
Model ID	Combination of one-hot encoded features	Test RMSE	Train RMSE
1	('Week', 'BackupStart')	0.10904	0.10058
2	('Week',)	0.10898	0.10183
3	('Day',)	0.10094	0.10089
4	('Week', 'Day', 'BackupStart', 'WFID', 'FileType')	840322638320.0	0.08836
5	('Day', 'BackupStart')	0.0997	0.09964
6	('Week', 'Day', 'FileType')	159767665784.0	0.08998
7	('Week', 'Day', 'BackupStart')	4127647143.52	0.09963
8	('Week', 'Day', 'BackupStart', 'WFID')	1787718997.8	0.08833
9	('BackupStart', 'WFID')	0.08944	0.08938
10	('Week', 'BackupStart', 'WFID')	1686127452.2	0.08938
11	('Week', 'WFID')	0.09315	0.09079
12	('BackupStart',)	0.10067	0.10059
13	**('Day', 'BackupStart', 'WFID')**	**0.08837**	**0.08834**

14	**('Day', 'BackupStart', 'FileType')**	**0.08837**	**0.08833**
15	('Week', 'Day', 'WFID', 'FileType')	89834552977.5	0.08977
16	('BackupStart', 'WFID', 'FileType')	0.08944	0.08938
17	('Day', 'FileType')	0.08977	0.08975
18	('Week', 'Day', 'WFID')	1646758925.89	0.08975
19	('WFID',)	0.09083	0.09079
20	('Week', 'WFID', 'FileType')	42813905940.0	0.09079
21	('FileType',)	0.09083	0.09079
22	('Day', 'WFID')	0.08977	0.08975
23	('Week', 'BackupStart', 'FileType')	114471211153.0	0.08943
24	()	0.1019	0.10183
25	('Week', 'FileType')	33920156977.6	0.09105
26	('BackupStart', 'FileType')	0.08946	0.0894
27	('Week', 'BackupStart', 'WFID', 'FileType')	103651914651.0	0.08939
28	('Day', 'BackupStart', 'WFID', 'FileType')	0.08839	0.08835
29	('Week', 'Day')	0.10779	0.10089
30	('Week', 'Day', 'BackupStart', 'FileType')	64060546724.2	0.08834
31	('Day', 'WFID', 'FileType')	0.0898	0.08979
32	('WFID', 'FileType')	0.09083	0.09079

Best combination - Highlighted in the table

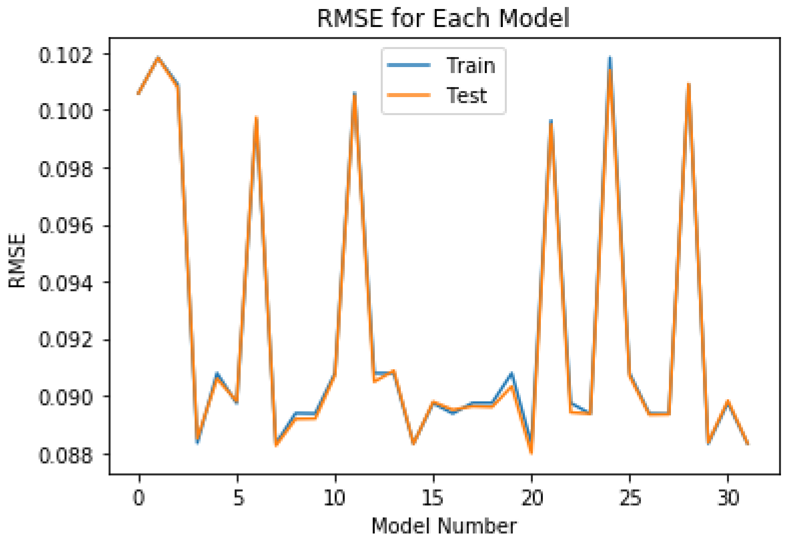
Reason - Making these features as one-hot encoding allows more degrees of freedom in linear regression. The regressor is able to put emphasis on each category of the feature and thus able to fit the data more efficiently. Without one-hot encoding, it would have to put equal emphasis on all days of the week and back up hours which increases the error because some backups happen only on specific days at specific times. Making it one-hot encoded allows the model to learn those specific backups and hence can predict the backup size with lesser error.

Coefficients of Best Model(s):
Without Shuffle - **('Day', 'BackupStart', 'WFID')**



```
Coefficients =  {'Week': 1.0000000000000001e-05, 'BackupStart_3': 0.033450000
000000001, 'Day_1': 0.039309999999999998, 'Day_5': -0.01286, 'Day_6': -0.0202
7, 'FileType': 1.0000000000000001e-05, 'WFID_4': 0.072910000000000003, 'WFID_
3': -0.056890000000000003, 'WFID_2': -0.040140000000000002, 'WFID_1': -0.014
1, 'WFID_0': 0.03823, 'Day_4': -0.0052399999999999999, 'Day_2': 0.00326999999
99999999, 'Day_3': 0.00149, 'BackupStart_4': -0.00199, 'BackupStart_5': 0.002
0100000000000001, 'BackupStart_2': 0.0077999999999999996, 'Day_0': -0.0057099
999999999998, 'BackupStart_0': -0.020199999999999999, 'BackupStart_1': -0.021
059999999999999}
Intercept = 0.06071
```

With Shuffle=True in KFold CV



On setting shuffle to True, the unreasonably large errors disappear.
We feel that this is because the training data is more complete and varied when shuffle is True and hence the regression model is able to learn better. Shuffle=True also prevents ill-conditioning which leads to many inaccuracies in matrix calculations.

V. Controlling ill conditioning and overfitting

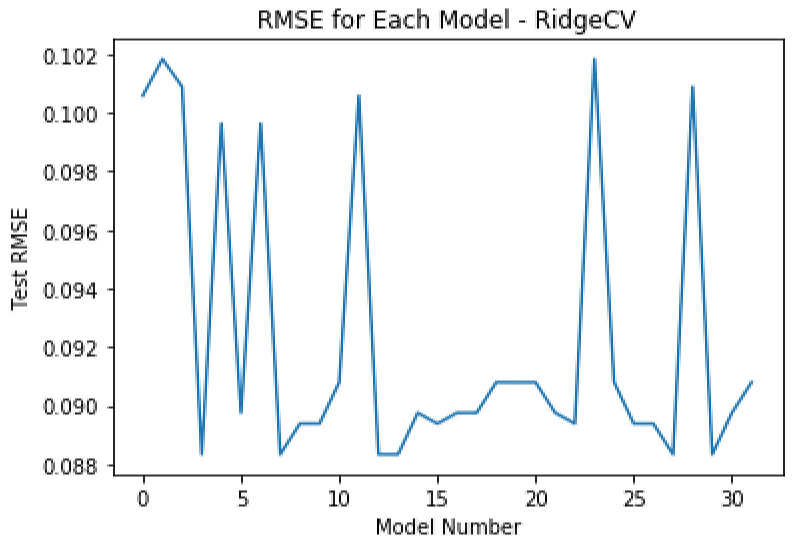
The increases in test error compared to training error are due to overfitting and ill-conditioning . The model ‘memorizes’ the training data and is unable to predict the test set well, which leads to higher RMSE error. This is the reason why the model never predicts any values greater than 0.2 as observed in the fitted vs actual plots in the earlier graphs. Another reason for the large errors are because when shuffle=False, the training data for the cross validation does not have all the types of data. By not training on all types of data, the performs poorly on the test data. When we set shuffle=True, this issue is resolved and all models which had high errors perform much better.

Let’s observe the coefficients for the model trained on one-hot encodings for all the features.
Coefficients = {'Week_14': -91390101122.195526, 'Week_13': 9892793003.6797104, 'Week_12': -21805251204.642071, 'Week_11': 308431915857.99731,
'Week_10': 9892793003.6779308, 'FileType_15': 67433949603.242653,
'FileType_14': 67433949603.243652, 'FileType_17': -203628558218.67944, 'FileType_16': 67433949603.243027, 'FileType_11': 67433949603.243263, 'FileType_10': 67433949603.242973, 'FileType_13': 67433949603.243584, 'FileType_12': -430485400602.67566, 'FileType_19': -203628558218.67911, 'FileType_18': -203628558218.67892, 'FileType_1': -430485400602.67554, 'FileType_0': -430485400602.67664, 'FileType_3': 324851196099.97571,
'FileType_2': 324851196099.9754, 'FileType_5': 99121236992.401398,
'FileType_4': 99121236992.40094, 'FileType_7': 99121236992.400726,
'FileType_6': 99121236992.401947, 'FileType_9': 99121236992.40242,
'FileType_8': 99121236992.403076, 'BackupStart_4': -280609414571.0647, 'BackupStart_5': -280609414571.06097, 'BackupStart_2': -280609414571.0553, 'BackupStart_3': -280609414571.02948, 'BackupStart_0': -280609414571.08325, 'BackupStart_1': -280609414571.08368, 'WFID_4': 110449624842.31075,
'WFID_3': -160612882979.74139, 'WFID_2': -192300170368.88293,
'WFID_1': -418030129476.4325, 'WFID_0': 337306467226.27185,
'FileType_20': -203628558218.68005, 'FileType_21': -203628558218.67929, 'FileType_22': -203628558218.67911, 'FileType_23': -430485400602.67578, 'FileType_24': -430485400602.67615, 'FileType_25': -430485400602.67615, 'FileType_26': 324851196099.97614, 'FileType_27': 324851196099.97577, 'FileType_28': 324851196099.97638, 'FileType_29': 324851196099.9762,
'Day_0': -120105341793.97502, 'Day_1': -120105341793.9285,
'Day_6': -120105341793.98862, 'Day_4': -120105341793.97322,
'Day_5': -120105341793.98141, 'Week_9': 12804640910.573339,
'Week_8': 9892793003.6858292, 'Week_7': 9892793003.6792393,
'Week_6': -78091942999.143845, 'Week_5': 9892793003.6855793,
'Week_4': 9892793003.6790295, 'Week_3': -234119996583.09976,
'Week_2': 246525428020.979, 'Week_1': 9892793003.6796093,
'Week_0': -16101488828.84498, 'Day_2': -120105341793.96494,
'Day_3': -120105341793.9668}

```
Intercept = 484000896738.0
```

As you can see, all the coefficients are of extremely high magnitude. This is due to ill-conditioning. We can solve this by setting shuffle=True to avoid the training data being multicollinear which leads to very inaccurate matrix inverse computations in regression.

A) Ridge Regularization



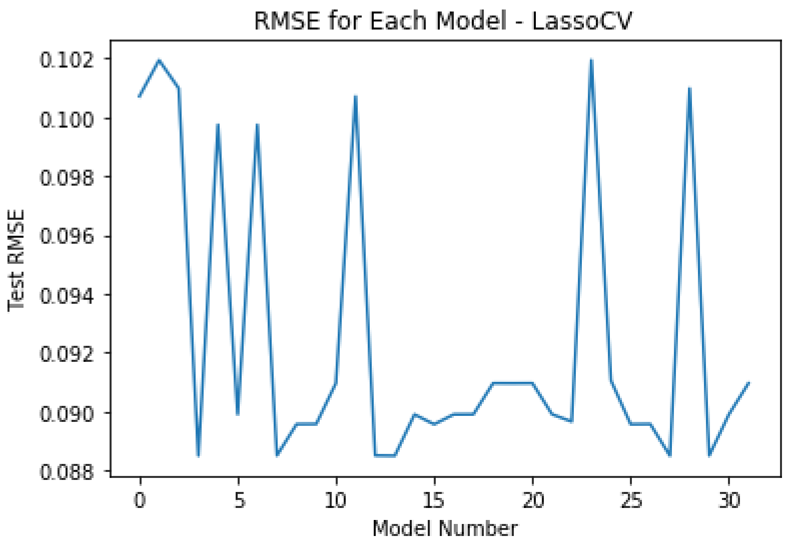
Best model - ('Week', 'Day', 'BackupStart', 'WFID', 'FileType')
RMSE- 0.0883437430779

Alpha = 10

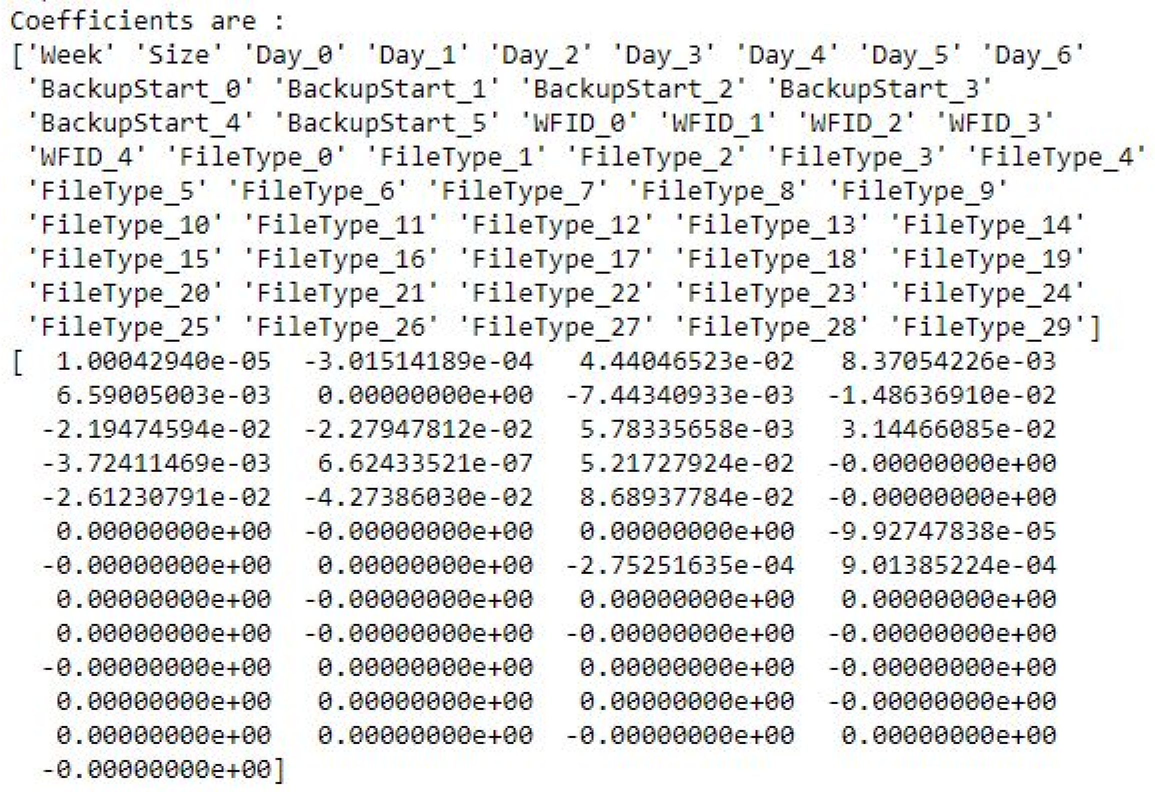
```
Coefficients are :
['Size' 'Week_0' 'Week_1' 'Week_2' 'Week_3' 'Week_4' 'Week_5' 'Week_6'
 'Week_7' 'Week_8' 'Week_9' 'Week_10' 'Week_11' 'Week_12' 'Week_13'
 'Week_14' 'Day_0' 'Day_1' 'Day_2' 'Day_3' 'Day_4' 'Day_5' 'Day_6'
 'BackupStart_0' 'BackupStart_1' 'BackupStart_2' 'BackupStart_3'
 'BackupStart_4' 'BackupStart_5' 'WFID_0' 'WFID_1' 'WFID_2' 'WFID_3'
 'WFID_4' 'FileType_0' 'FileType_1' 'FileType_2' 'FileType_3' 'FileType_4'
 'FileType_5' 'FileType_6' 'FileType_7' 'FileType_8' 'FileType_9'
 'FileType_10' 'FileType_11' 'FileType_12' 'FileType_13' 'FileType_14'
 'FileType_15' 'FileType_16' 'FileType_17' 'FileType_18' 'FileType_19'
 'FileType_20' 'FileType_21' 'FileType_22' 'FileType_23' 'FileType_24'
 'FileType_25' 'FileType_26' 'FileType_27' 'FileType_28' 'FileType_29']
[[ -7.83199355e-04  -3.94976577e-05   3.47525492e-04  -3.47561001e-04
    3.27679475e-04  -2.11829780e-04   8.11762466e-06   8.16983110e-04
    1.25480225e-04   1.22866983e-04  -5.65272589e-04   7.16534904e-04
   -1.83186957e-04  -1.13259218e-05  -3.23314552e-04  -5.68425111e-03
    3.91626179e-02   3.26492031e-03   1.49100259e-03  -5.22906727e-03
   -1.28127550e-02  -2.01924674e-02  -2.01413758e-02  -2.09907238e-02
    7.76661309e-03   3.33486506e-02  -1.98528418e-03   2.00212009e-03
    3.26887279e-02  -1.20284369e-02  -3.43801961e-02  -4.86656502e-02
    6.23855553e-02   4.74047673e-03   5.96978479e-03  -2.27752009e-03
   -1.95498153e-03  -6.46218598e-03  -6.05627270e-03  -5.59003207e-03
   -6.63968500e-03  -4.16187559e-03  -5.47014474e-03  -8.26908879e-03
   -7.94711469e-03   5.64558997e-03  -7.88916195e-03  -8.24317691e-03
   -8.29102327e-03  -8.02608459e-03   1.01417546e-02   1.06963562e-02
    1.08828917e-02   9.83330493e-03   1.04693185e-02   1.03619292e-02
    5.61951734e-03   5.23522357e-03   5.47813554e-03  -1.62509271e-03
   -2.10083770e-03  -1.89554210e-03  -2.17446281e-03]]
```

The best model for RidgeCV is the same as the best model with shuffle=True.

B) Lasso Regularization



The performance of LassoCV and RidgeCV is very similar, as they are both minimizing the same function.
Best model - ('Day', 'BackupStart', 'WFID', 'FileType')
RMSE - 0.0885045028052
Alpha = 2.25183926814e-05

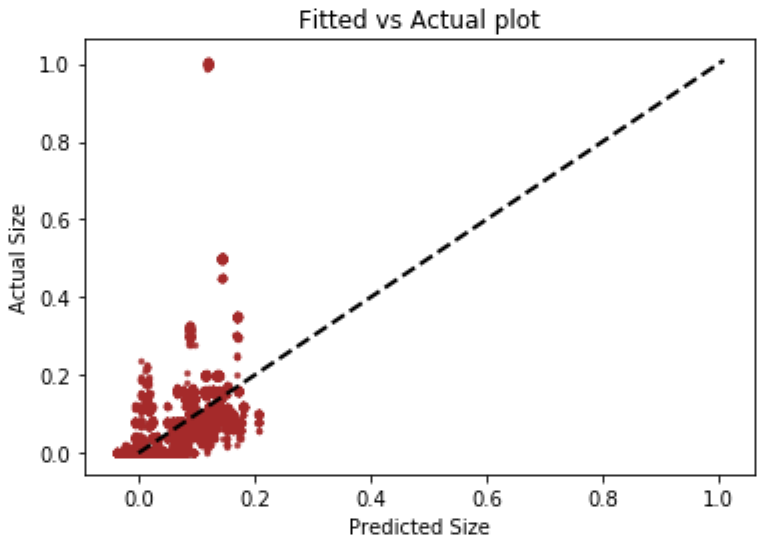


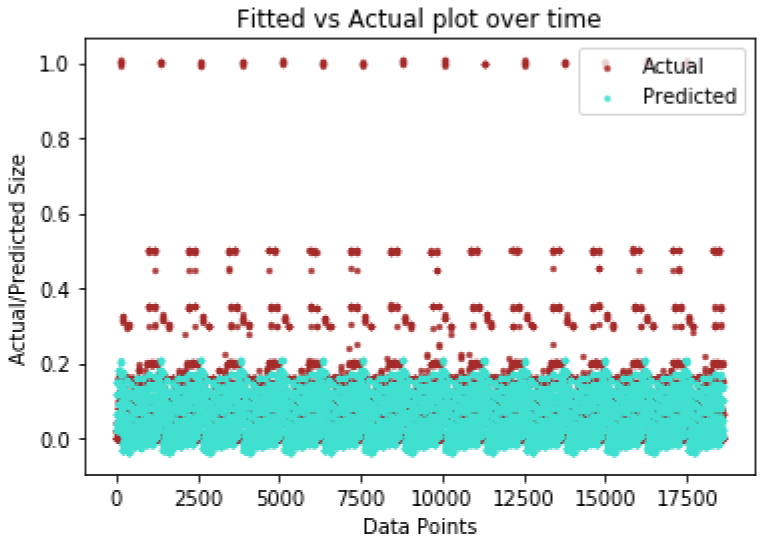
```
Coefficients are :
['Week' 'Size' 'Day_0' 'Day_1' 'Day_2' 'Day_3' 'Day_4' 'Day_5' 'Day_6'
 'BackupStart_0' 'BackupStart_1' 'BackupStart_2' 'BackupStart_3'
 'BackupStart_4' 'BackupStart_5' 'WFID_0' 'WFID_1' 'WFID_2' 'WFID_3'
 'WFID_4' 'FileType_0' 'FileType_1' 'FileType_2' 'FileType_3' 'FileType_4'
 'FileType_5' 'FileType_6' 'FileType_7' 'FileType_8' 'FileType_9'
 'FileType_10' 'FileType_11' 'FileType_12' 'FileType_13' 'FileType_14'
 'FileType_15' 'FileType_16' 'FileType_17' 'FileType_18' 'FileType_19'
 'FileType_20' 'FileType_21' 'FileType_22' 'FileType_23' 'FileType_24'
 'FileType_25' 'FileType_26' 'FileType_27' 'FileType_28' 'FileType_29']
[  1.00042940e-05  -3.01514189e-04   4.44046523e-02   8.37054226e-03
   6.59005003e-03   0.00000000e+00  -7.44340933e-03  -1.48636910e-02
  -2.19474594e-02  -2.27947812e-02   5.78335658e-03   3.14466085e-02
  -3.72411469e-03   6.62433521e-07   5.21727924e-02  -0.00000000e+00
  -2.61230791e-02  -4.27386030e-02   8.68937784e-02  -0.00000000e+00
   0.00000000e+00  -0.00000000e+00   0.00000000e+00  -9.92747838e-05
  -0.00000000e+00   0.00000000e+00  -2.75251635e-04   9.01385224e-04
   0.00000000e+00  -0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00  -0.00000000e+00  -0.00000000e+00  -0.00000000e+00
  -0.00000000e+00   0.00000000e+00   0.00000000e+00  -0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00  -0.00000000e+00
   0.00000000e+00   0.00000000e+00  -0.00000000e+00   0.00000000e+00
  -0.00000000e+00]
```

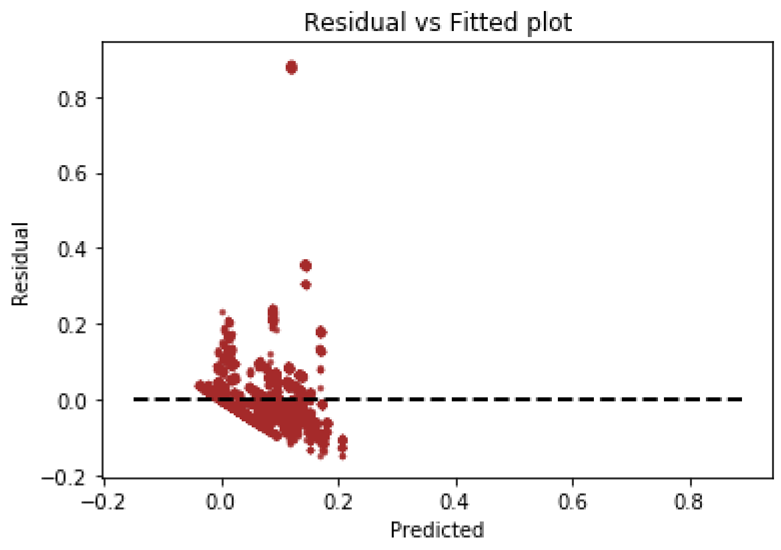
Difference between Ridge and Lasso Regression :
Ridge regression can't zero out coefficients; thus, it either ends up including all the coefficients in the model, or none of them. Lasso regression on the other hand, does parameter selection and variable selection automatically. This difference is clearly noticeable when we observe the coefficients for both the models. Ridge regression reduces the coefficients to as little as possible, whereas Lasso regression sets them to 0.

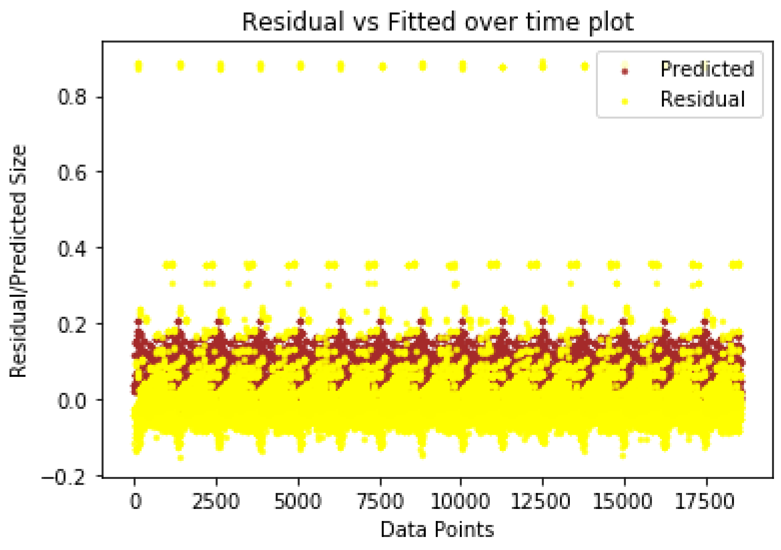
Overall Best Model for Linear Regression

The best linear regression model is the one obtained by RidgeCV. The graphs for that model are as follows:









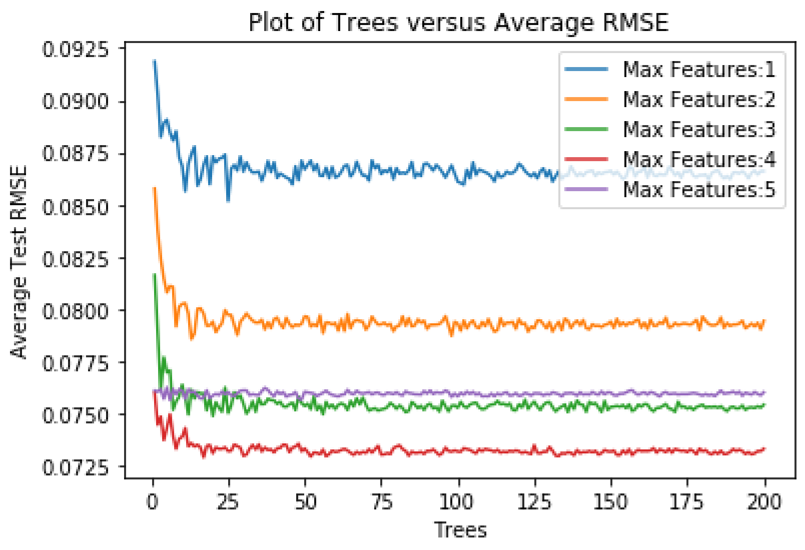
By comparing this graph with the baseline graph, it shows minimal improvements. Although, the case of outliers is still not addressed. But on the whole, residuals are closer to zero than in the baseline model.

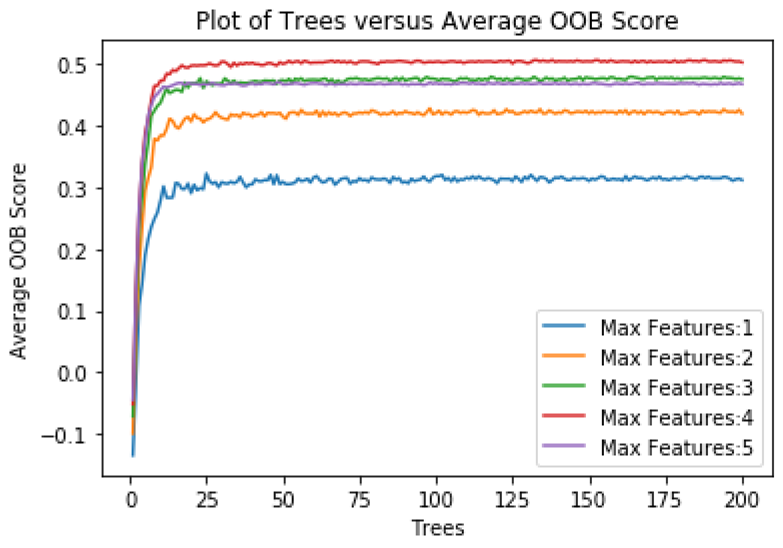
Question 2(b): Random Forest

I. **Average Test RMSE** is: 0.07612258
Average Train RMSE is: 0.07603504
OOB Score is: 0.534581998013456

We can conclude from the average Train and Test RMSE that the model is not overfitting as the errors are almost the same. If we had higher Test RMSE than Train RMSE, we could have said the model is overfitting, but in this case, the errors are same upto the 3rd decimal place, which makes it very comparable. This is a very low mean squared error, which suggests that this initial model performed pretty well. We also observe a low OOB which suggests that these initial parameters are a good start.
We see below in the graph of Fitted vs Actual for the Random Forest Regressor that the distribution follows the line closely stating that the model is good and there is not a significant difference between most of the actual and predicted labels. We can see that the fitted and the actual values over time overlap in majority of the area and this is only achievable if we have low variation between the actual and the fitted values. This implies that our model is performing well on this dataset and gives good fitted results for the initial Random Forest parameters.

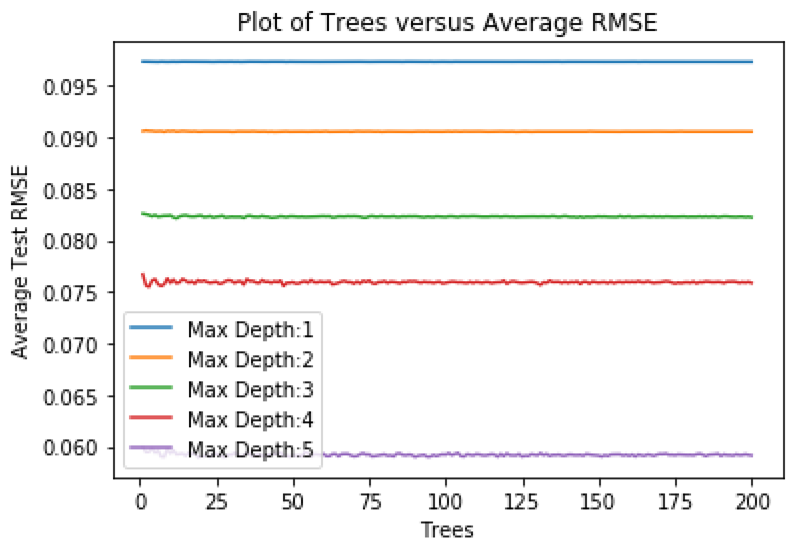
II. Sweep over 1-200 Trees and 1-5 features and plot:

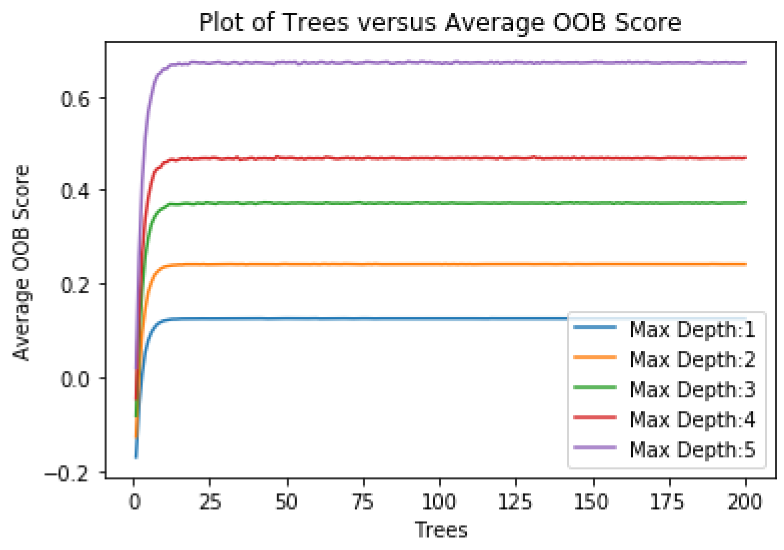




As we can observe from the above two graphs, the test RMSE begins to fall as we increase the number of trees upto a certain point and then saturates. Also, the test RMSE is evidently lesser for higher number of features (in this case features=4). Since we don't see a drastic difference in error for different number of trees, we set trees (n_estimators) to 100 (for further sections). We also note that the Test RMSE for max_features = 5 is higher than 3 or 4. This could be because one of the features of the dataset doesn't contribute to the backup size predictions.

III. We picked **depth** as another parameter to test. Number of trees were in the range **1-200** and features was set to **4**. Depth varied from **1-5**





As we can observe from the above two graphs, the test RMSE begins to fall as we increase the number of trees upto a certain point and then saturates. Also, the test RMSE is evidently lesser as the depth increases (in this case depth=5). Since we don't see a drastic difference in error for different number of trees, we set trees (n_estimators) to 100 (for further sections).

IV. From the above analysis, we find that the best model has the following parameters:

   A. N_estimators = 100
   B. Max_depth = 4 (We wanted to use depth =5, but the instruction manual has specifically asked us to use depth =4 for the next part).
   C. Max_features = 4
   D. Bootstrap = True

The following are the feature importance for the model trained with above parameters (in decreasing order):
WFID: 4.67262975e-01,
Day: 3.41828096e-01,

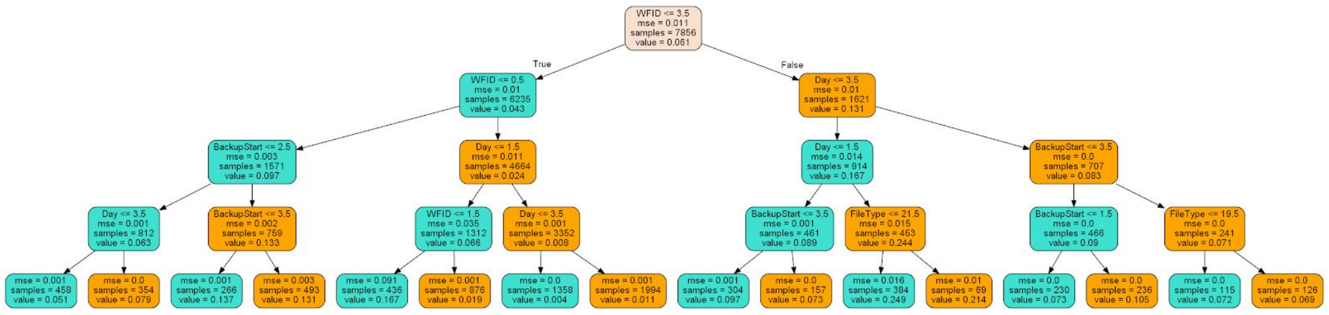
FileType: 9.68016074e-02,
BackupStart: 9.39838222e-02,
Week: 1.23499232e-04

If we observe carefully, it does make sense to see Workflow ID with highest importance, as each workflow has its own pattern and that directly affects the backup size.

V. Visualize your Decision Trees:

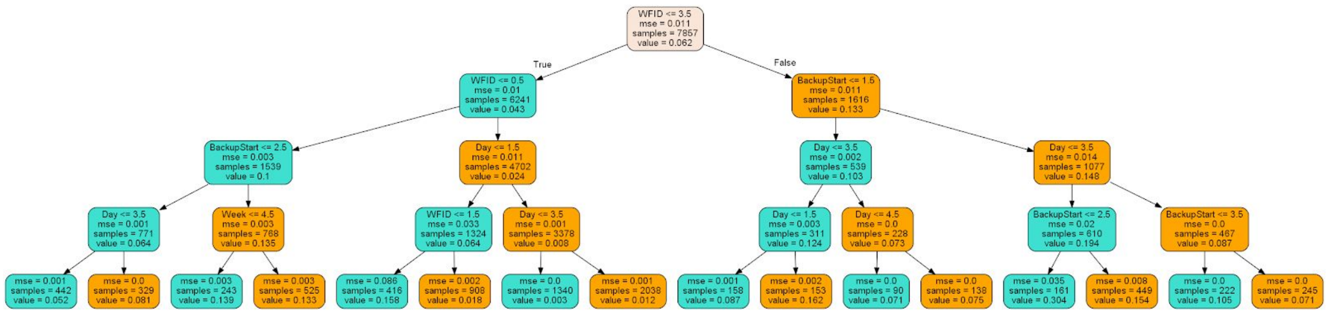
A. We plot the first 5 decision trees in the random forest regressor:

TREE 1



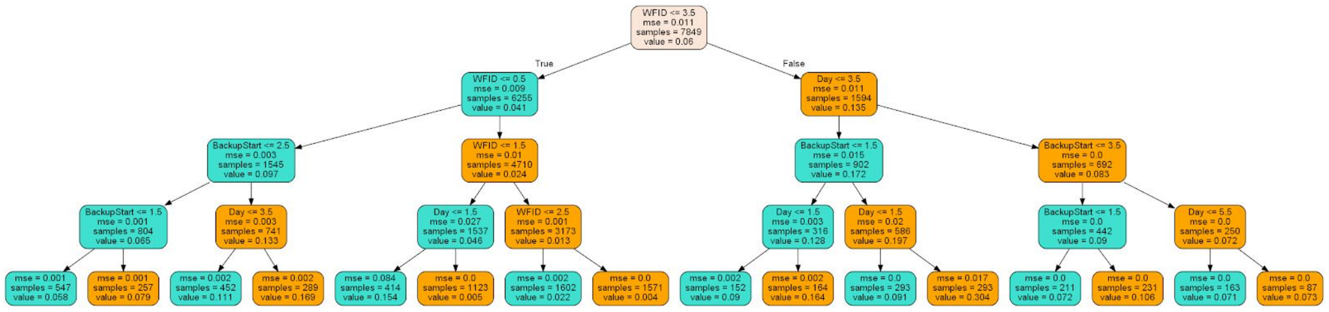
WFID <= 3.5 mse = 0.011 samples = 7856 value = 0.061
True
False
WFID <= 0.5 mse = 0.01 samples = 6235 value = 0.043
Day <= 3.5 mse = 0.01 samples = 1621 value = 0.131
BackupStart <= 2.5 mse = 0.003 samples = 1571 value = 0.097
Day <= 1.5 mse = 0.011 samples = 4664 value = 0.024
Day <= 1.5 mse = 0.014 samples = 914 value = 0.167
BackupStart <= 3.5 mse = 0.0 samples = 707 value = 0.083
Day <= 3.5 mse = 0.001 samples = 812 value = 0.063
BackupStart <= 3.5 mse = 0.002 samples = 759 value = 0.133
WFID <= 1.5 mse = 0.035 samples = 1312 value = 0.066
Day <= 3.5 mse = 0.001 samples = 3352 value = 0.008
BackupStart <= 3.5 mse = 0.001 samples = 461 value = 0.089
FileType <= 21.5 mse = 0.015 samples = 453 value = 0.244
BackupStart <= 1.5 mse = 0.0 samples = 466 value = 0.09
FileType <= 19.5 mse = 0.0 samples = 241 value = 0.071
mse = 0.001 samples = 458 value = 0.051
mse = 0.0 samples = 354 value = 0.079
mse = 0.001 samples = 266 value = 0.137
mse = 0.003 samples = 493 value = 0.131
mse = 0.091 samples = 436 value = 0.167
mse = 0.001 samples = 876 value = 0.019
mse = 0.0 samples = 1358 value = 0.004
mse = 0.001 samples = 1994 value = 0.011
mse = 0.001 samples = 304 value = 0.097
mse = 0.0 samples = 157 value = 0.073
mse = 0.016 samples = 384 value = 0.249
mse = 0.01 samples = 69 value = 0.214
mse = 0.0 samples = 230 value = 0.073
mse = 0.0 samples = 236 value = 0.105
mse = 0.0 samples = 115 value = 0.072
mse = 0.0 samples = 126 value = 0.069

TREE 2



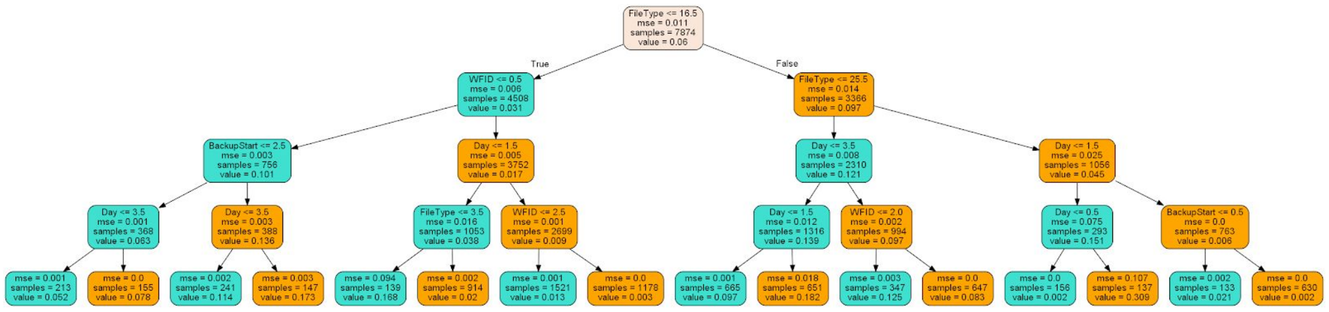
WFID <= 3.5 mse = 0.011 samples = 7857 value = 0.062
True
False
WFID <= 0.5 mse = 0.01 samples = 6241 value = 0.043
BackupStart <= 1.5 mse = 0.011 samples = 1616 value = 0.133
BackupStart <= 2.5 mse = 0.003 samples = 1539 value = 0.1
Day <= 1.5 mse = 0.011 samples = 4702 value = 0.024
Day <= 3.5 mse = 0.002 samples = 539 value = 0.103
Day <= 3.5 mse = 0.014 samples = 1077 value = 0.148
Day <= 3.5 mse = 0.001 samples = 771 value = 0.064
Week <= 4.5 mse = 0.003 samples = 768 value = 0.135
WFID <= 1.5 mse = 0.033 samples = 1324 value = 0.064
Day <= 3.5 mse = 0.001 samples = 3378 value = 0.008
Day <= 1.5 mse = 0.003 samples = 311 value = 0.124
Day <= 4.5 mse = 0.0 samples = 228 value = 0.073
BackupStart <= 2.5 mse = 0.02 samples = 610 value = 0.194
BackupStart <= 3.5 mse = 0.0 samples = 467 value = 0.087
mse = 0.001 samples = 442 value = 0.052
mse = 0.0 samples = 329 value = 0.081
mse = 0.003 samples = 243 value = 0.139
mse = 0.003 samples = 525 value = 0.133
mse = 0.086 samples = 416 value = 0.158
mse = 0.002 samples = 908 value = 0.018
mse = 0.0 samples = 1340 value = 0.003
mse = 0.001 samples = 2038 value = 0.012
mse = 0.001 samples = 158 value = 0.087
mse = 0.002 samples = 153 value = 0.162
mse = 0.0 samples = 90 value = 0.071
mse = 0.0 samples = 138 value = 0.075
mse = 0.035 samples = 161 value = 0.304
mse = 0.008 samples = 449 value = 0.154
mse = 0.0 samples = 222 value = 0.105
mse = 0.0 samples = 245 value = 0.071

TREE 3

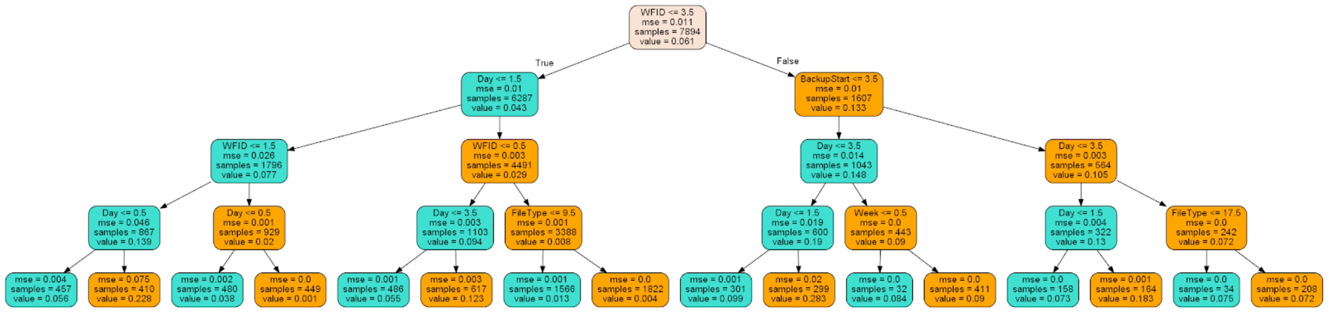


WFID <= 3.5 mse = 0.011 samples = 7849 value = 0.06
True
False
WFID <= 0.5 mse = 0.009 samples = 6255 value = 0.041
Day <= 3.5 mse = 0.011 samples = 1594 value = 0.135
BackupStart <= 2.5 mse = 0.003 samples = 1545 value = 0.097
WFID <= 1.5 mse = 0.01 samples = 4710 value = 0.024
BackupStart <= 1.5 mse = 0.015 samples = 902 value = 0.172
BackupStart <= 3.5 mse = 0.0 samples = 692 value = 0.083
BackupStart <= 1.5 mse = 0.001 samples = 804 value = 0.065
Day <= 3.5 mse = 0.003 samples = 741 value = 0.133
Day <= 1.5 mse = 0.027 samples = 1537 value = 0.046
WFID <= 2.5 mse = 0.001 samples = 3173 value = 0.013
Day <= 1.5 mse = 0.003 samples = 316 value = 0.128
Day <= 1.5 mse = 0.02 samples = 586 value = 0.197
BackupStart <= 1.5 mse = 0.0 samples = 442 value = 0.09
Day <= 5.5 mse = 0.0 samples = 250 value = 0.072
mse = 0.001 samples = 547 value = 0.058
mse = 0.001 samples = 257 value = 0.079
mse = 0.002 samples = 452 value = 0.111
mse = 0.002 samples = 289 value = 0.169
mse = 0.084 samples = 414 value = 0.154
mse = 0.0 samples = 1123 value = 0.005
mse = 0.002 samples = 1602 value = 0.022
mse = 0.0 samples = 1571 value = 0.004
mse = 0.002 samples = 152 value = 0.09
mse = 0.002 samples = 164 value = 0.164
mse = 0.0 samples = 293 value = 0.091
mse = 0.017 samples = 293 value = 0.304
mse = 0.0 samples = 211 value = 0.072
mse = 0.0 samples = 231 value = 0.106
mse = 0.0 samples = 163 value = 0.071
mse = 0.0 samples = 87 value = 0.073

TREE 4

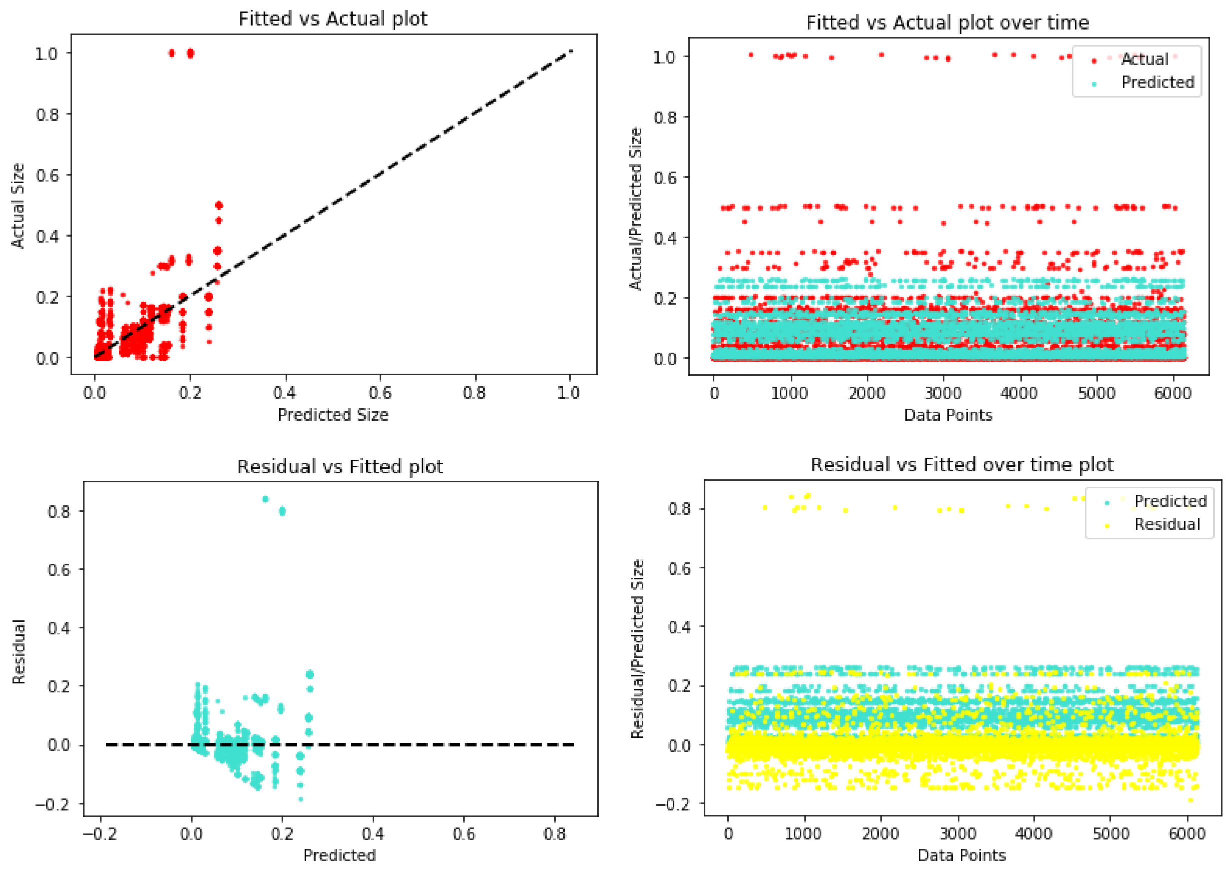


TREE 5



As we can see from all the trees above, the root node is WFID and it is the most important feature as well. As the project spec states, a workflow is defined as a task that backs up data from a group of files, which have similar patterns of change in terms of size over time. We can see that workflow ID is an important feature and will be used in classification as it is used to split and group the data based on similar backup sizes and hence is the root. When we compare them with part 1a graphs,we see that the patterns in backup sizes for each workflow ID repeat in cycles further illustrating how the workflow feature groups similar files together and hence is the root of the tree.

Scatter Plots for Random Forest Regressor:



We can observe from the above graphs that the RF regressor is unable to predict the outlier values. This could be due to the fact that all the workflows are passed together to the RF regressor for training and the model is not able to generalize among all the workflows and hence misses out on outliers between the workflows.

Question 2(c): Neural Network

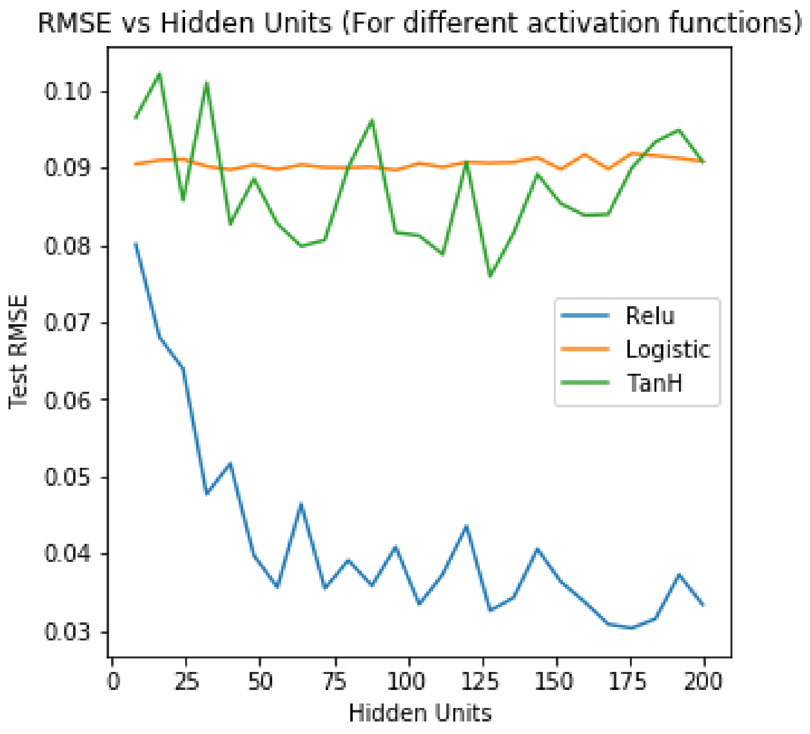
Objective - To Implement a simple a neural network regression model with only one hidden layer and using the input features as one hot encoded input features. To use this implementation to identify the optimal number of parameters in terms of hidden units and activation function.

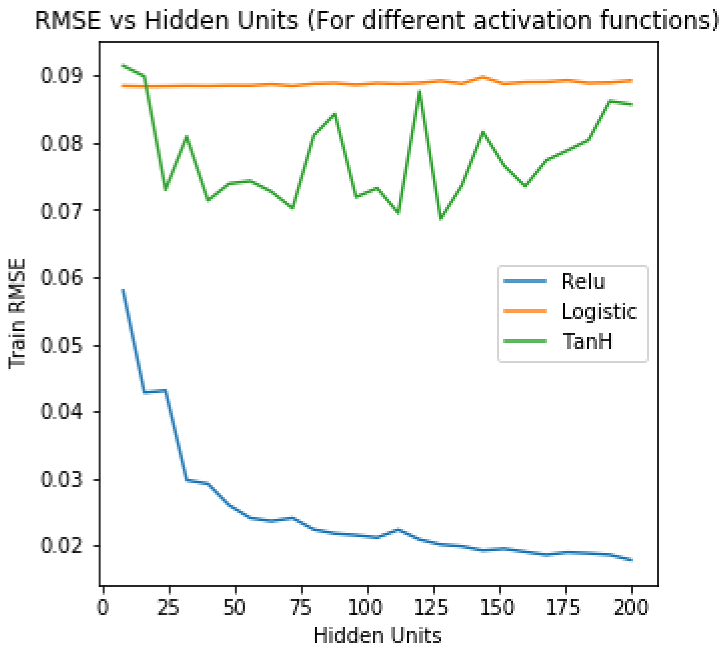
Approach - To implement this neural network model we used the MLP (multilayer perceptron) regressor with one hidden layer and used 10 fold cross validation to test the performance of our model with different parameters.

No of hidden units: For optimizing this parameter we used a range from 8 - 200 hidden units in steps of 8 i.e. 8,16,24…..200 hidden units. The reason for choosing this range is because we have around 64 input features and by general convention the number of hidden units used it upto 3X the number of units in the previous layer +/- 10. Hence we decided to stop at 200 hidden neurons. (64*3 + 8)

Activation function: We tried the above model for different hidden neurons with 3 types of activation functions, namely relu, tanH and logistic.

Results and Observations -





This graph shows the testing RMSE for different hidden units as well as different activation functions. From the graph we can observe that Relu activation function is the best performing of the three and the optimal hidden units were observed at 176.
Additional observations: Since neural networks initialize with random weights, every time we train the model different neurons 'learn' different features and attributes from the dataset and from our observation of running the model multiple times Relu always gave the best results whereas the number of hidden units kept varying for every time the model was execute. But

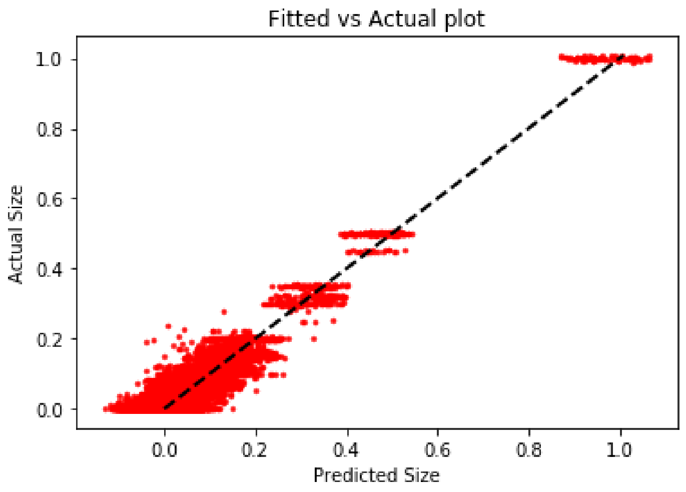
in all the cases the number of hidden units was always above 150 neurons and specifically in the range of 170-200 neurons. We are reporting the optimal values for one of the iterations.

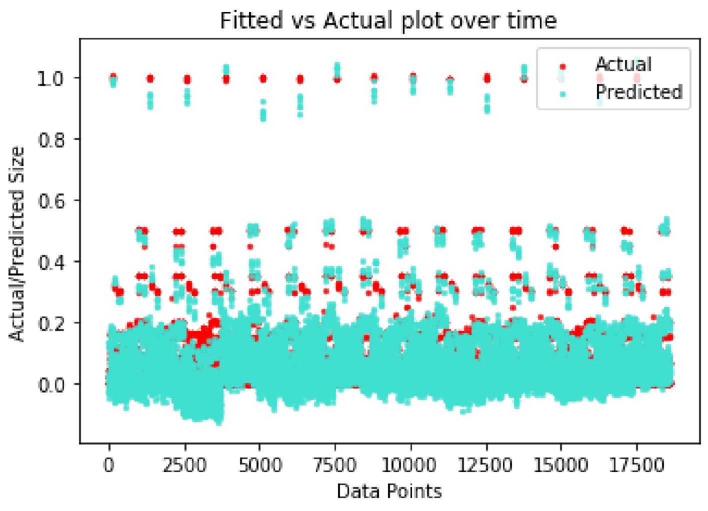
Logistic and Tanh don't do a good job with sparse representations because they always generate a non-zero value for the input 0 (sparse) leading to a dense representation.
For example: Logistic is : 1/(1 + e^{-x})
If x = 0, output is 1/1+1 = 0.5

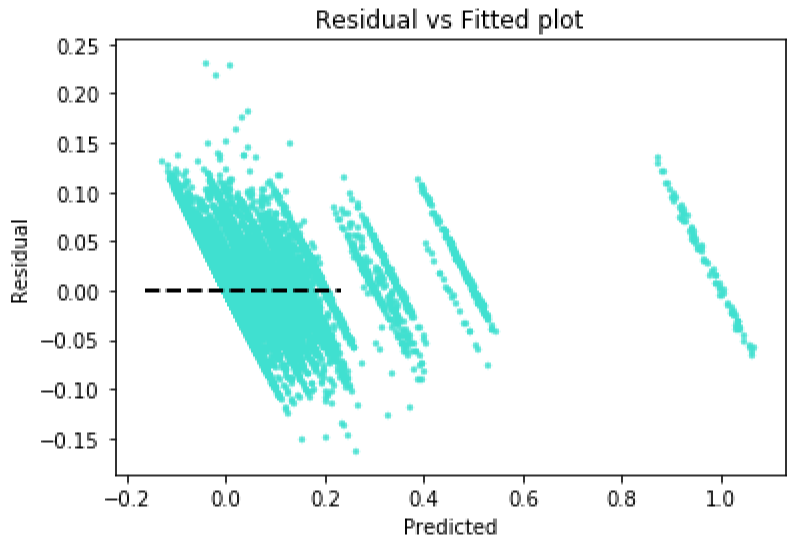
ReLu is performing better than the other two as it greatly accelerates the convergence of stochastic gradient descent compared to the logistic/tanh functions.
The relu activation function allows a network to easily obtain sparse representations and one hot encoding generates all sparse features. On a side node, relu also takes care of vanishing/exploding gradient.

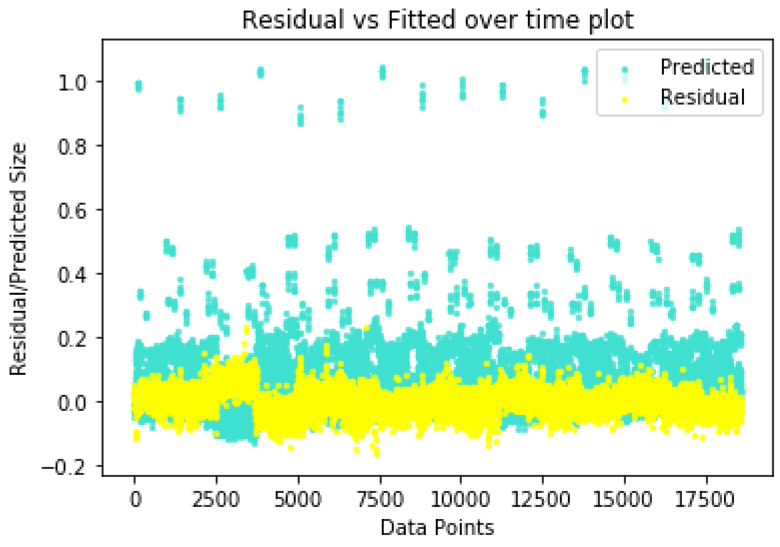
Optimal hidden units: 176
Optimal Activation Function: Relu (Rectified linear unit)
Mean Test RMSE: 0.030340
Mean Train RMSE: 0.018999

Scatter Plots:




Fitted vs Actual plot over time
Actual
Predicted
Actual/Predicted Size
Data Points
0.0
0.2
0.4
0.6
0.8
1.0
0
2500
5000
7500
10000
12500
15000
17500


Residual vs Fitted plot
Residual
Predicted
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


Residual vs Fitted over time plot
Predicted
Residual
Residual/Predicted Size
Data Points
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
0
2500
5000
7500
10000
12500
15000
17500

Interpretation: From the observed results we can infer some information about the performance of the neural network. The optimal number of hidden units in the neural net are roughly around the number of input features obtained after one hot encoding (63 input features). And we ran the model over (1, 200) hidden units (in steps of 8) because in theory the optimal number of hidden units should not exceed twice to three times the size of input features otherwise it'll lead to overfitting. The optimal number of hidden units we obtained is not too less (so underfitting is didn't happen) or more (so overfitting didn't happen), it is optimal.

Question 2(d): Polynomial Regression

Predicting for each workflow individually.

1. **Linear Regression** -

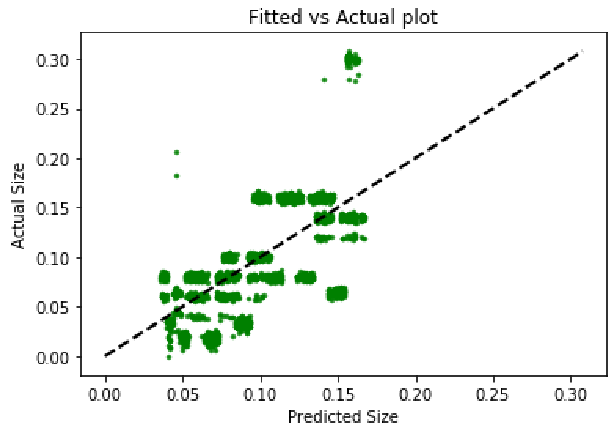
By doing linear regression for each workflow separately, there is a drastic improvement in the performance.
This is based on the error values in the following table.

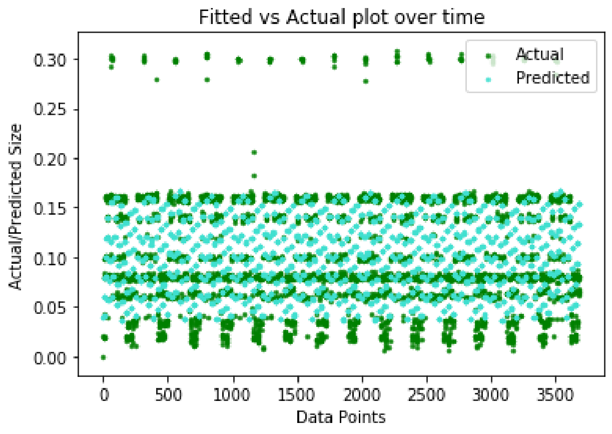
Workflow ID	**Train RMSE**	**Test RMSE**
0	0.04299	0.04302
1	0.15962	0.15856
2	0.04224	0.04224
3	0.00711	0.00711
4	0.10302	0.10297

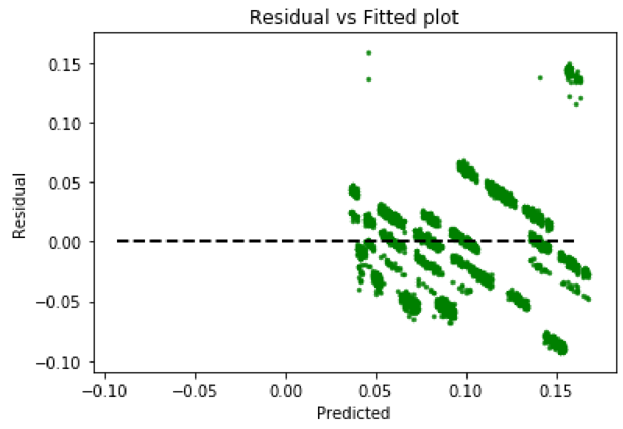
Average Train RMSE for all workflows = 0.070998
Average Test RMSE for all workflows = 0.070782

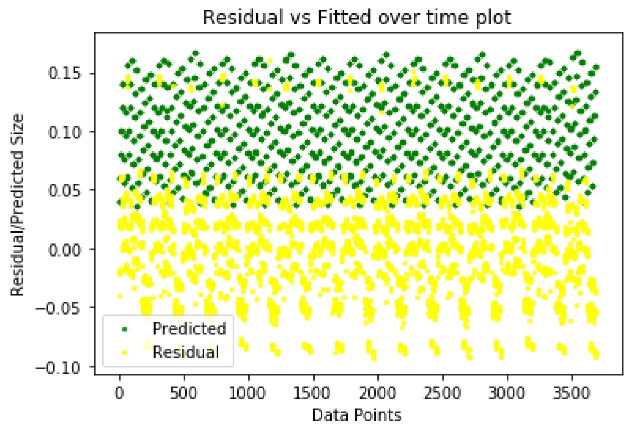
As we can observe from the table, the error is lower for some workflows. By having separate models for each workflow, each model tries to learn the backup size for each specific workflow. And as we've already observed from the visualization graph, each workflow follows a specific pattern - which the model tries to learn. Combining all the workflows and training only one model results in poor performance due to these different patterns.

Graphs for Workflow 0



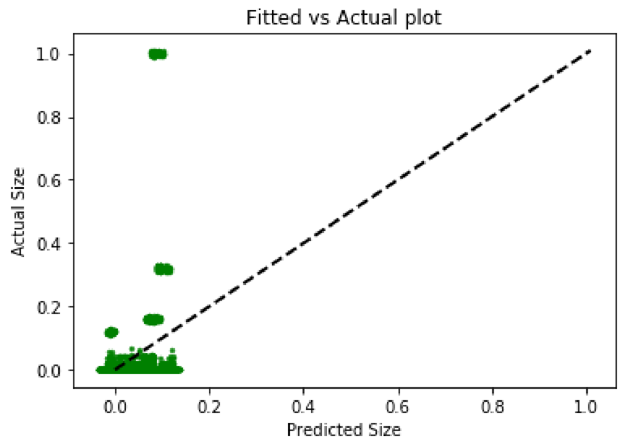


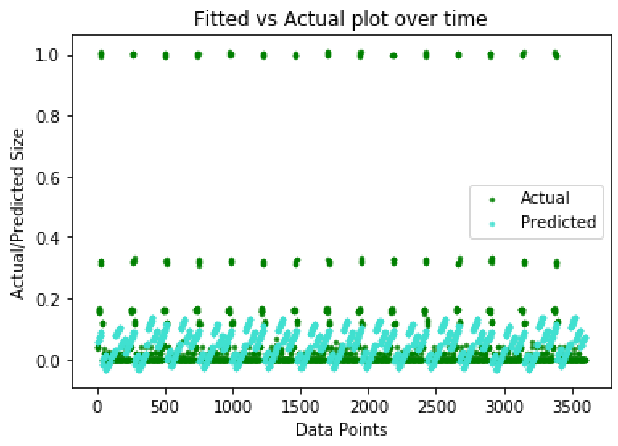


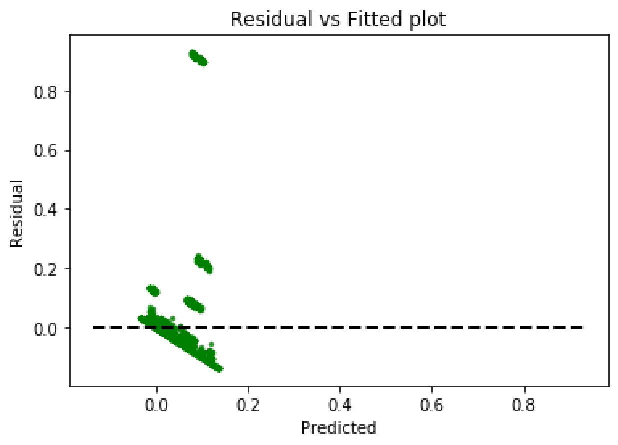


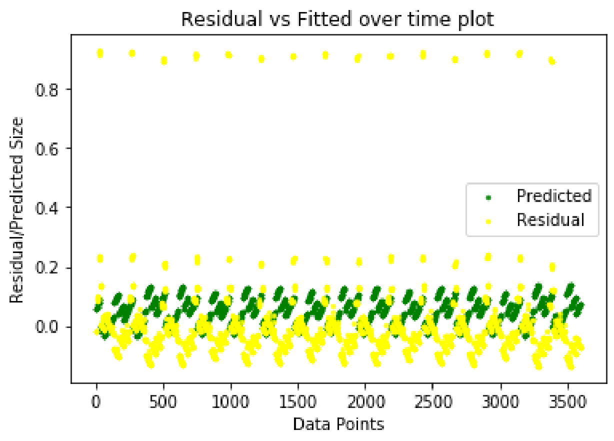
The graphs for workflow 0 indicate a better fit than the baseline model as the residuals are closer to 0. This is vindicated by the lower root mean square error. But the prediction can be better as the model still doesn't predict the outliers.

<u>Graphs for Workflow 1</u>



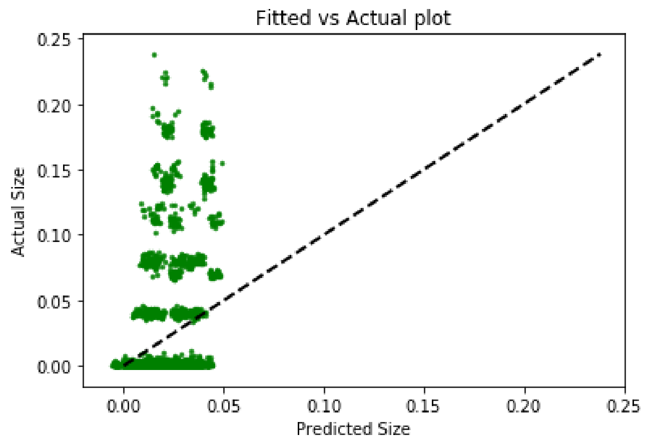


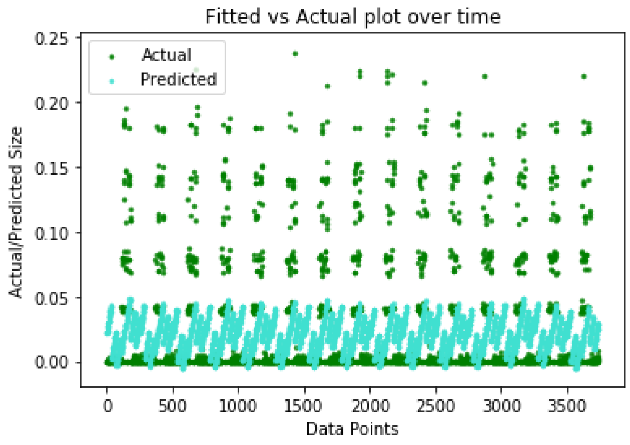


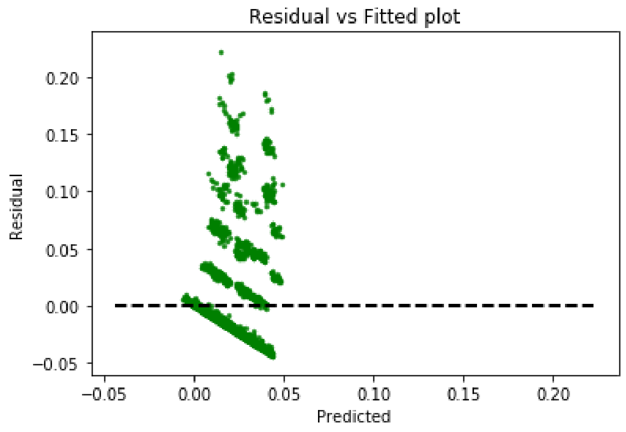


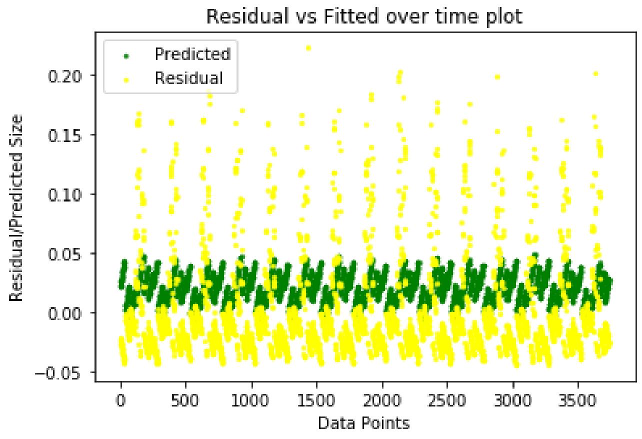
The graphs for workflow 1 indicate a worse fit than the baseline model as the model does not predict the outliers well. This is vindicated by the higher root mean square error.

Graphs for Workflow 2



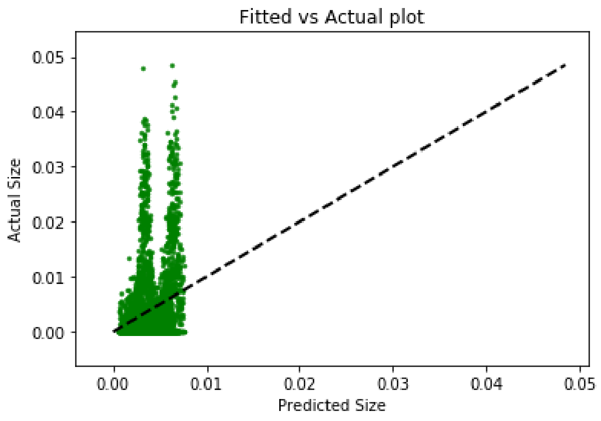


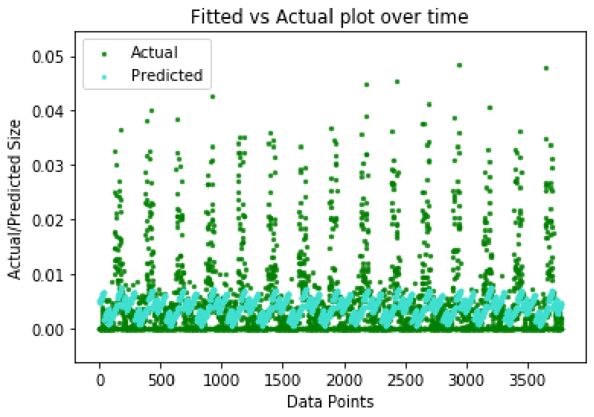


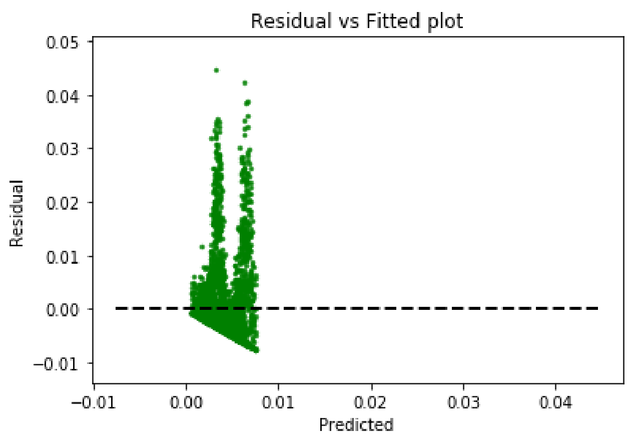


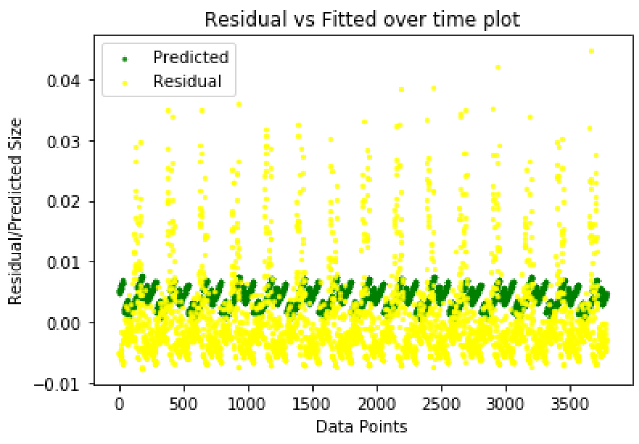
The graphs for workflow 2 indicate a better fit than the baseline model as the residuals are closer to 0. This is vindicated by the lower root mean square error. But the prediction can be better as the model still doesn't predict the outliers.

Graphs for Workflow 3



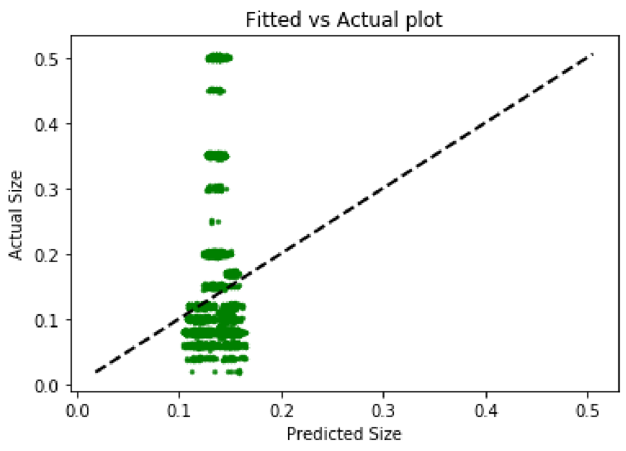


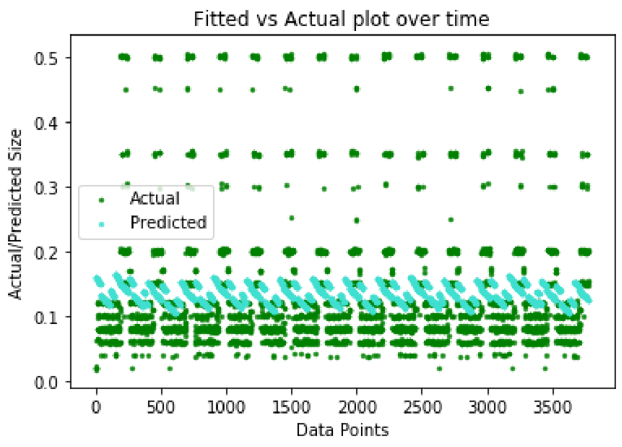


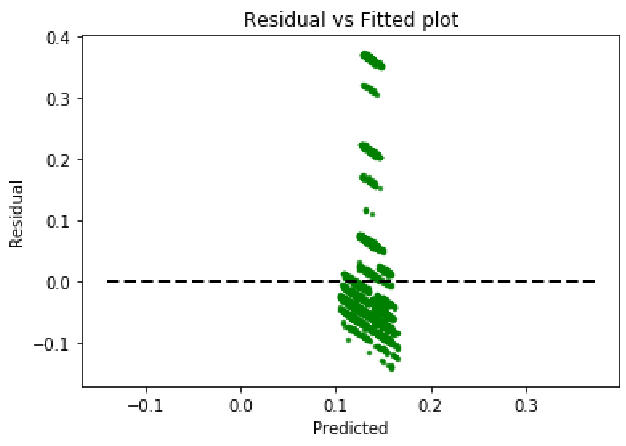


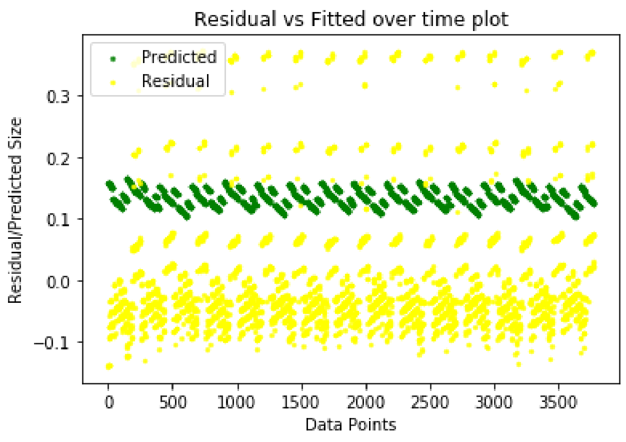
The graphs for workflow 3 indicate a better fit than the baseline model as the residuals are closer to 0. This is vindicated by the lower root mean square error.

<u>Graphs for Workflow 4</u>







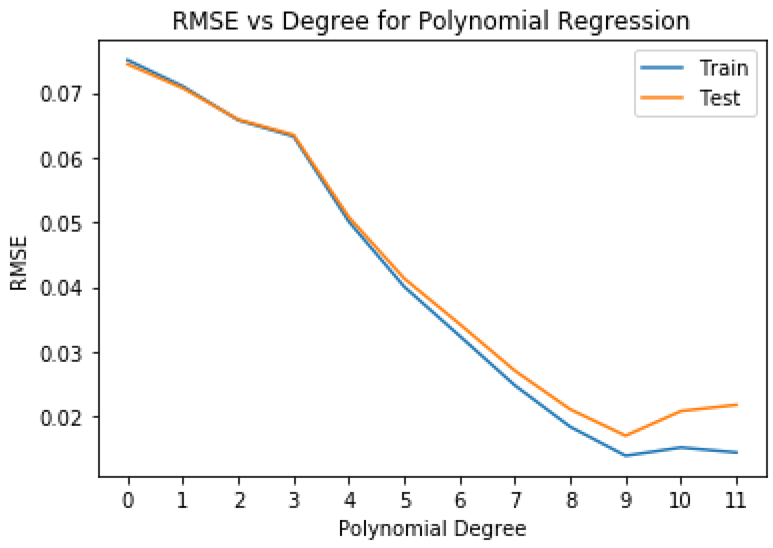


The graphs for workflow 4 indicate a worse fit than the baseline model as the model only learns to predict the backup size in a small range. This is vindicated by the higher root mean square error.

We see that predicting separately for workflows works well for WorkflowID 0. In WorkflowID 1 we see that there are more outliers, and not working as well as other workflows. The same pattern can be seen in test RMSE's. Also, this method works very well for WorkflowIDs 2 and 3. For WorkflowID 4 it works well but worse than the others, but still good.

2. **Polynomial function** -

To address the non-linearity in the data, we express it in terms of polynomials.
The following graph shows the average error across all workflows vs degree of polynomials.



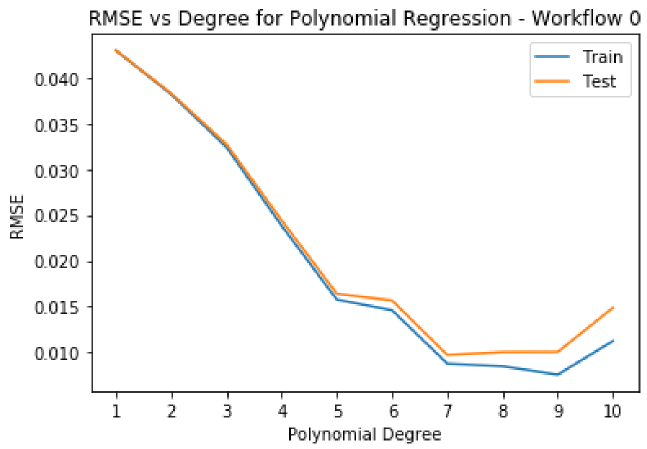
As we can see from the graph, train and test rmse errors decrease with increasing polynomial degree. The threshold is polynomial degree 9. After this degree, the error starts to increase again, this is because of extremely high model complexity to fit so many features.

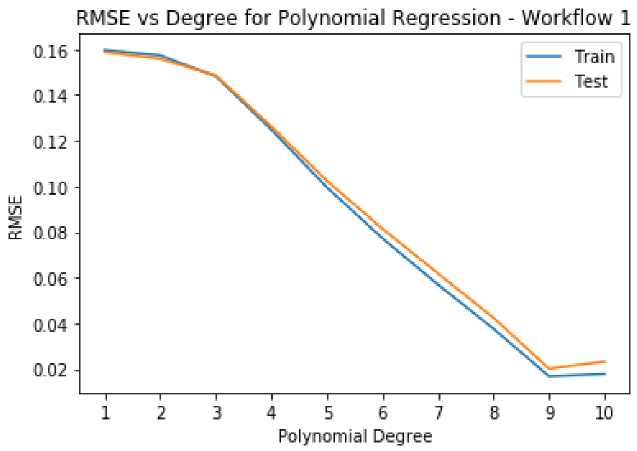
Best Polynomial Regression for degree 9
Average Train RMSE = 0.015132
Average Test RMSE = 0.01887
The RMSE values for polynomial regression are lower compared to linear regression, highlighting the non-linear relation between the dependent and independent variables of the model.

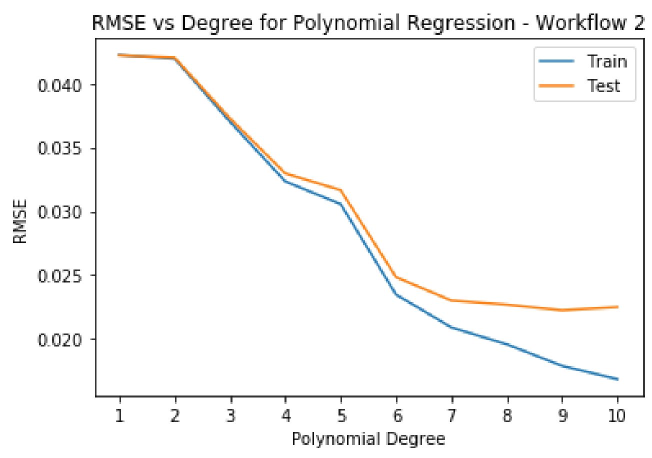
The above results mentioned were averaged out from the models for each workflow. Let us look at the behaviour of models for each workflow. In the table below, P represents the degree of the polynomial and each cell represents the test RMSE error for the workflow model with that polynomial degree transformation.

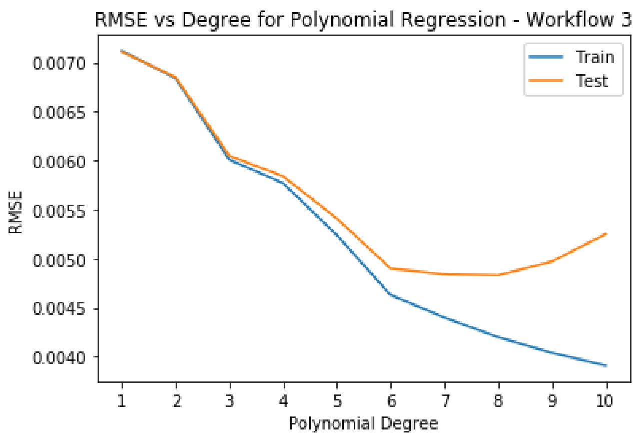
Work flow ID	**P = 1**	**P = 2**	**P = 3**	**P = 4**	**P = 5**	**P = 6**	**P = 7**	**P = 8**	**P = 9**	**P=10**
WFID 0	0.04302	0.03838	0.03272	0.02435	0.01637	0.01548	**0.00976**	0.01007	0.00995	0.01519
WFID 1	0.15748	0.1567	0.14804	0.12698	0.10288	0.08145	0.06243	0.04245	**0.02009**	0.02404
WFID 2	0.04224	0.04204	0.03732	0.03292	0.03167	0.02489	0.02285	0.02264	**0.02241**	0.02289
WFID 3	0.00712	0.00685	0.00605	0.00587	0.00542	0.00492	**0.00484**	0.00487	0.005	0.00532
WFID 4	0.10295	0.08453	0.09035	0.06424	0.05035	0.04477	0.02951	**0.02529**	0.031	0.03343

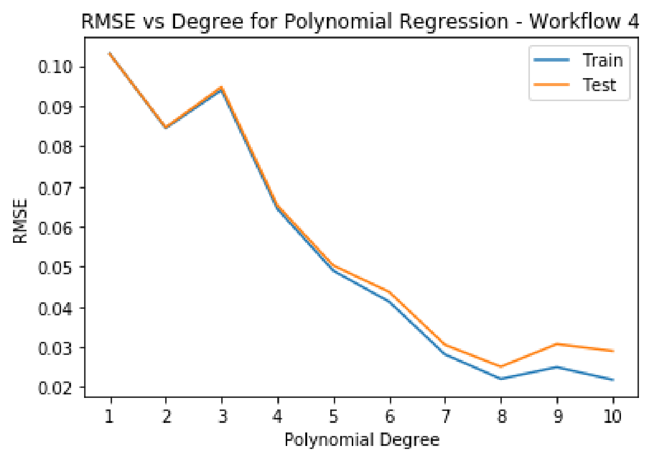
The table above shows the Test RMSE for each workflow with varying degree. As we can see, different models fit differently, highlighting that different workflows have different patterns for their backup size. Workflows 0 and 3 reach their threshold at degree 7, Workflows 1 and 2 reach their threshold at degree 9 whereas the last workflow has a polynomial degree threshold of 8 before the error starts to increase. The following graphs show the same results in a pictorial way.





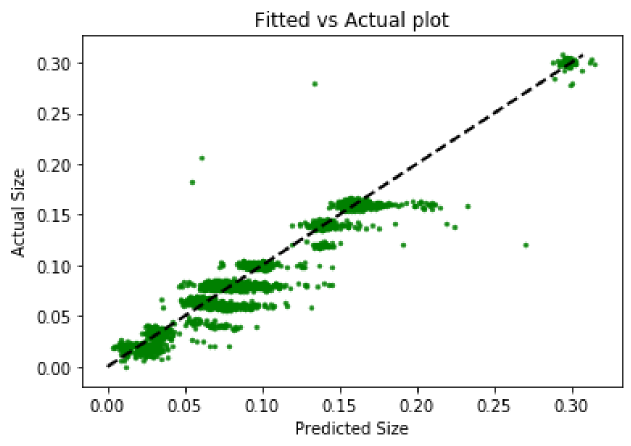


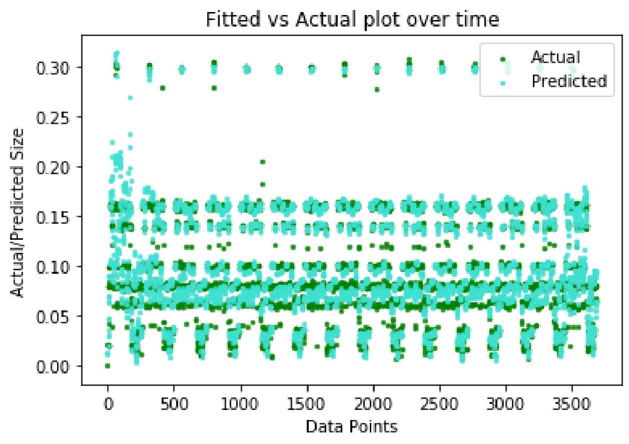


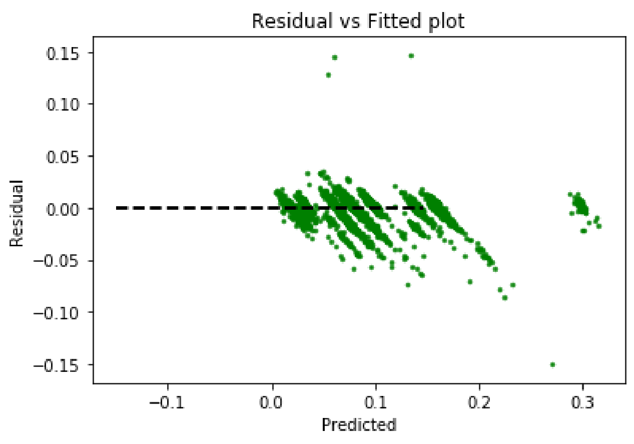


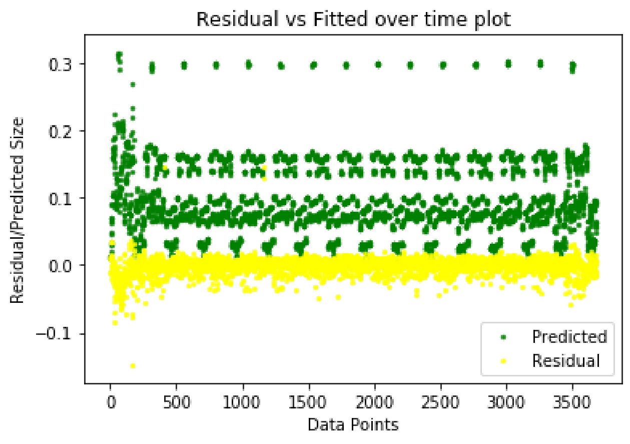
Let us now observe the different scatter plots for the best model for each workflow.

Graphs for Workflow 0



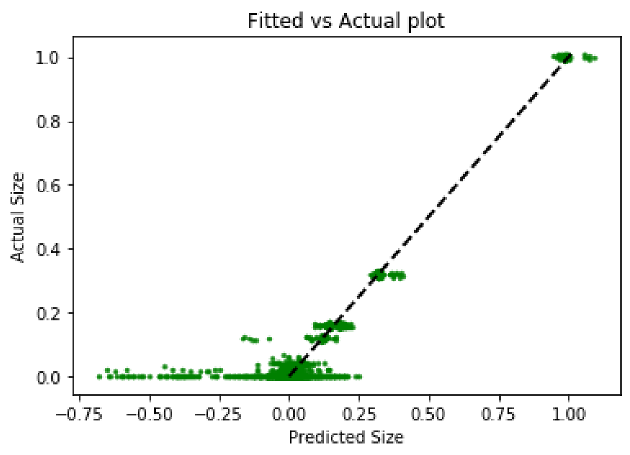


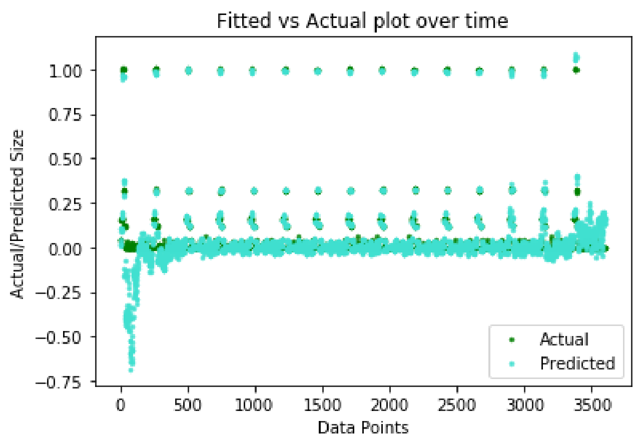


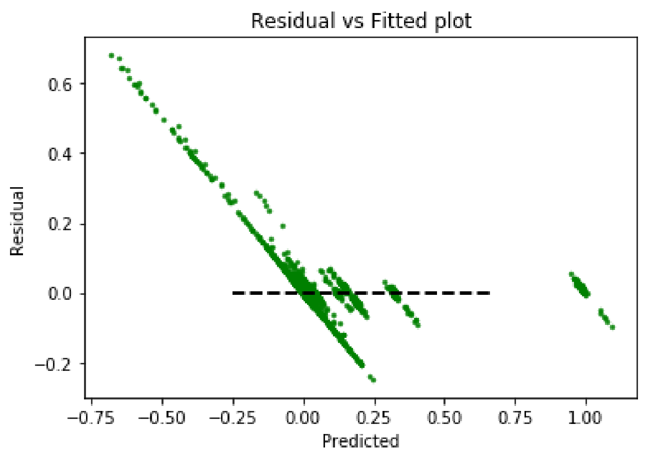


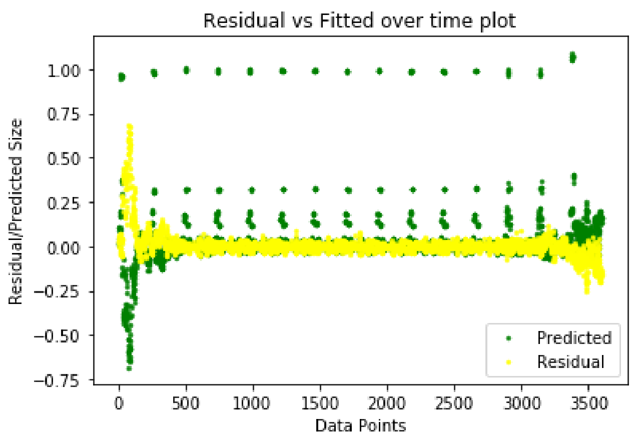
Compared to the graph for workflow 0 using linear regression, there is a clear improvement. The residuals are closer to the zero line and the model is not prone to outliers anymore.

<u>Graphs for Workflow 1</u>



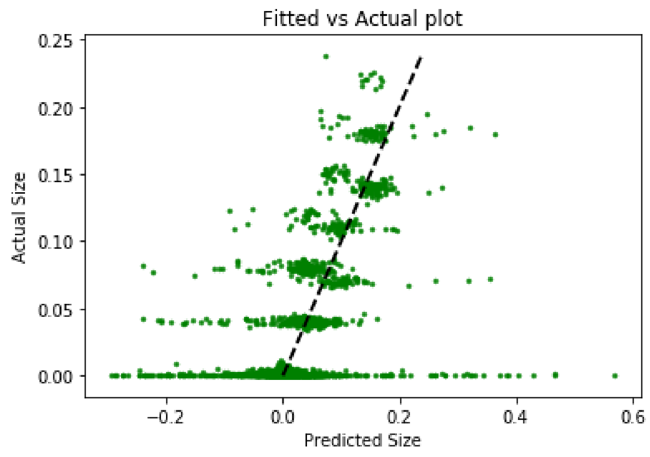


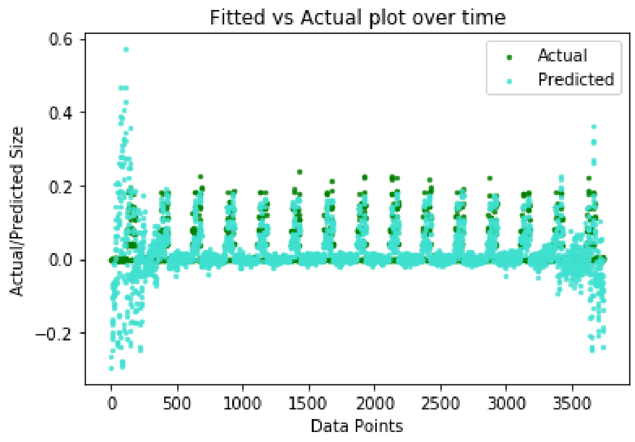


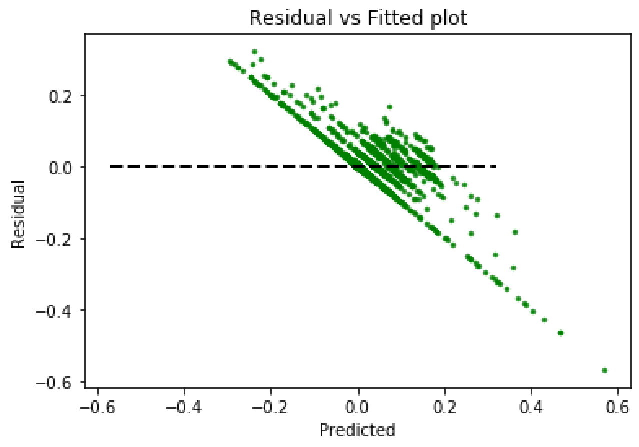


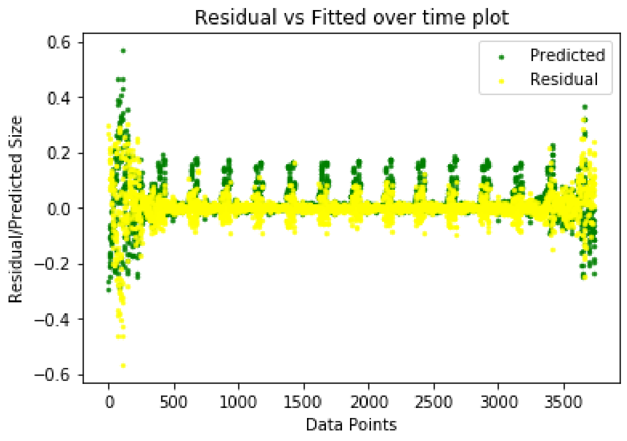
Compared to the graph for workflow 1 using linear regression, there is a clear improvement. The residuals are closer to the zero line and the model is not prone to outliers anymore.

Graphs for Workflow 2



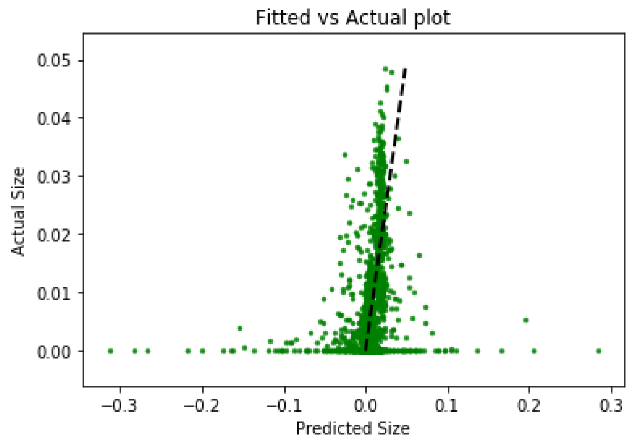


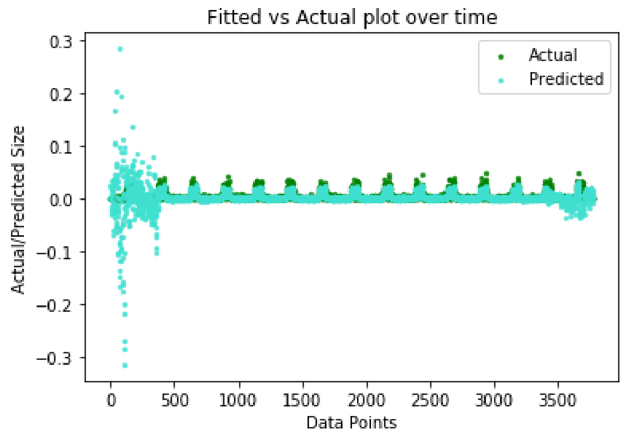


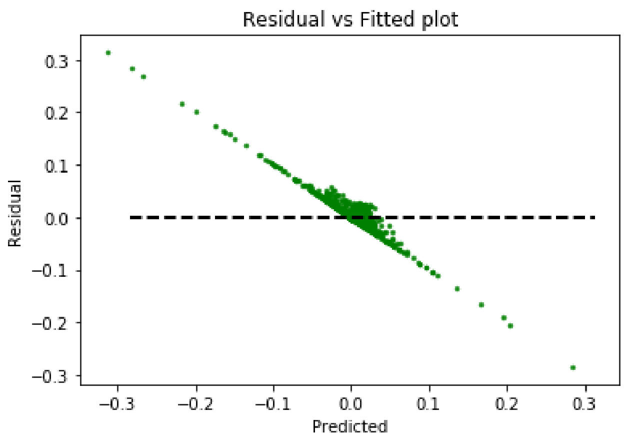


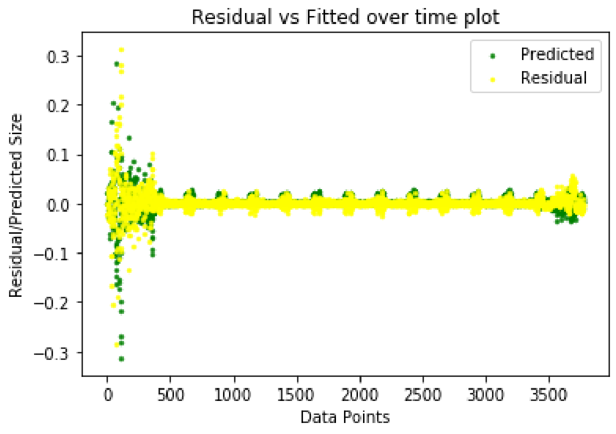
Compared to the graph for workflow 2 using linear regression, there is a clear improvement. The residuals are closer to the zero line and the model is not prone to outliers anymore.

Graphs for Workflow 3



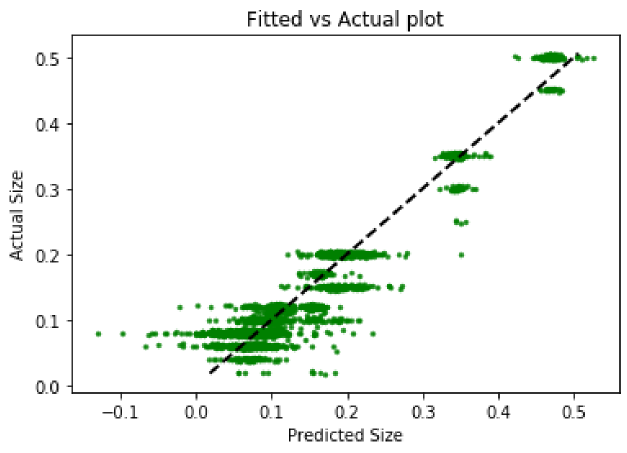


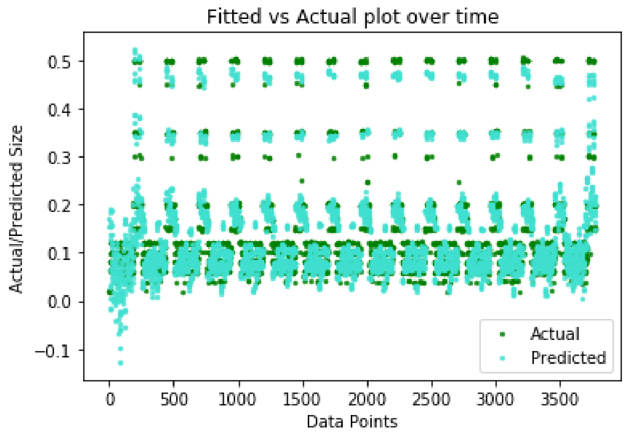


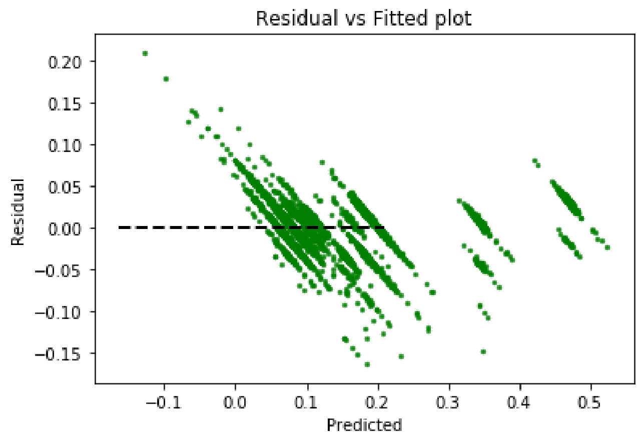


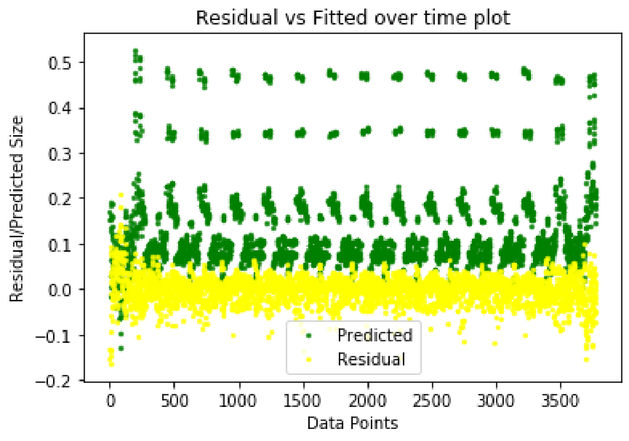
Compared to the graph for workflow 3 using linear regression, there is a clear improvement. The residuals are closer to the zero line and the model is not prone to outliers anymore.

<u>Graphs for Workflow 4</u>









Compared to the graph for workflow 4 using linear regression, there is a clear improvement. The residuals are closer to the zero line and the model is not prone to outliers anymore.

It is evident from the above results that different datasets generate different best-fit polynomials. To overcome this issue when, by using cross validation, we train multiple models to see all the data which can allow us to understand the results and the model performance in a better way.

When we increase the degree of the polynomial initially, it reduces the average train and test error and is useful as it maybe be underfitting before this.

After a certain degree we see that the test error increases while the training error decreases. This is the threshold being talked about and we are able to find it. When this happens, it means we are overfitting the data and hence should not increase the degree of the polynomial anymore.

Cross validation reduces the model complexity and prevents overfitting of data. It generalizes the model across k-folds to find the most general generalized model by varying what you learn and what data you learn on. It helps is in controlling the complexity of our model.

We see from the graphs that our prediction for separate workflows works well for all the workflows. In the fitted vs actual graph, the line should follow closely. The time graphs for the same should have greater overlap which would tell us the model is performing well. For the predicted vs residual, this should be centered around the horizontal line. And the time graphs of above should again have greater overlap.

Question 2(e): K-Nearest Neighbour

Objective - To Implement a k-nearest neighbours regression model using the input features as the simple numerical features. To use this implementation to identify the optimal number of neighbours for the problem.

Approach - To implement this KNN model we tested the model using num of neighbours ranging from 1 to 19. We evaluated the performance of the model using 10 fold cross validation and test RMSE as the scoring metric.

Results and Observations -
This is the following graph observed of test RMSE vs number of neighbours in KNN

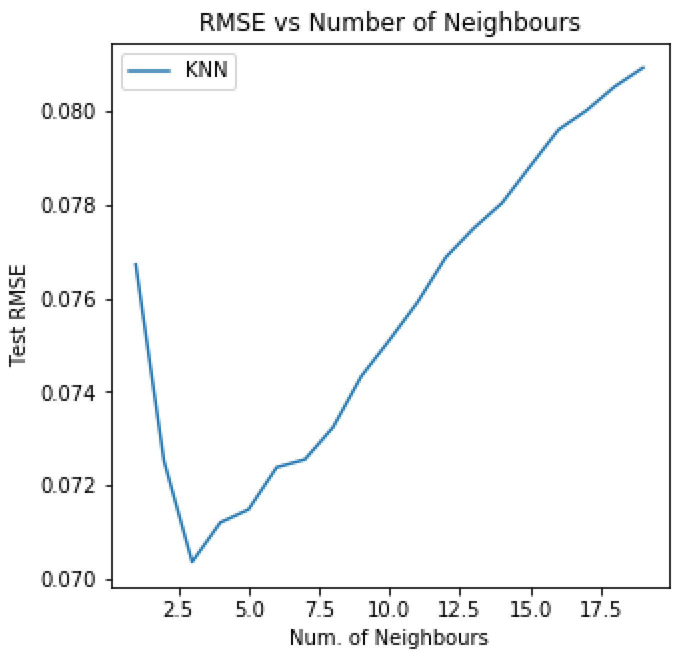
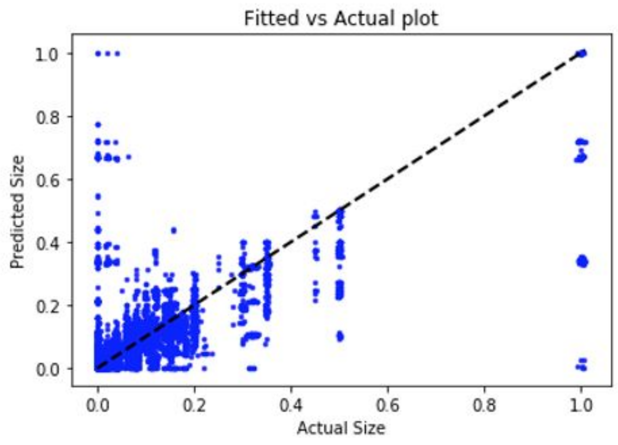


Fig: RMSE vs No. of Neighbours

From the graph we can observe that the optimal value for no. of neighbours is: 3

For N = 3 we get a test rmse of 0.070366 and a training rmse of 0.036799



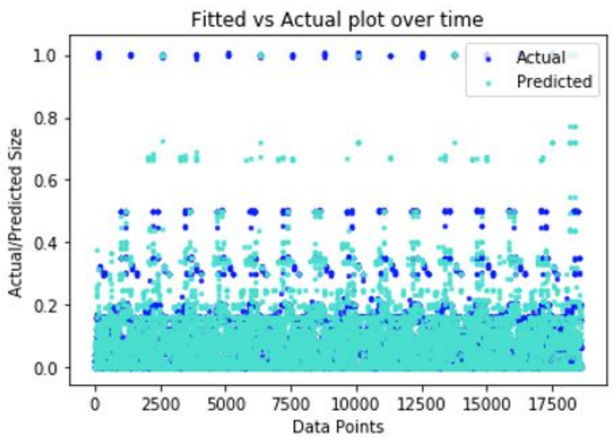


Fig: Fitted vs Actual Labels

Fig: Fitted vs Actual Labels over time

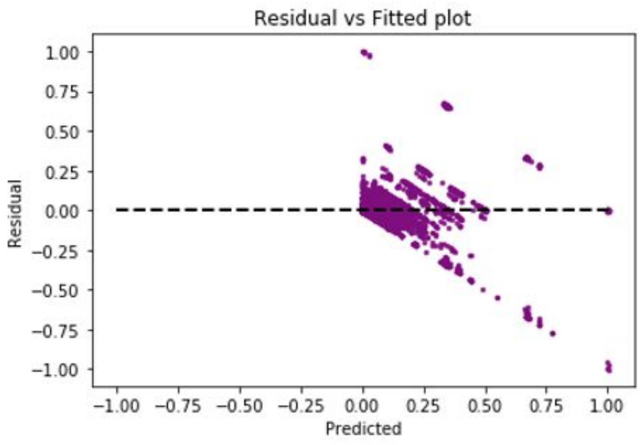


Fig: Residual vs Fitted

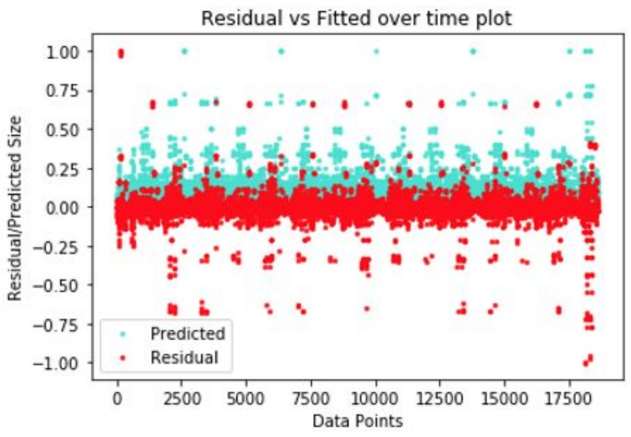


Fig: Residual vs Fitted over time

In the fitted vs actual graph, the distribution should follow the line closely as this would mean that the model is good which means there is not a significant difference between most of the actual and predicted labels. We see the same for knn when k=3. This gives us a good fit. When we see the fitted and the actual values over time graphs, we see that there is an overlap in majority of the area and this is only achievable if we have low variation between the actual and the fitted values. This would hence imply that our model is performing well. For our predicted data vs residual data, the distribution should be centered around the horizontal line with most values lying between a small range, which is [-0.25,0.25] in this case. In the residual vs fitted over time plot, the residual values and fitted values should overlap as much. This shows that the difference between the true labels and the predicted labels are not too different, thus the residuals are low. Again, this leads to believe that our model is performing well.

Question 3: Compare these regression models you have used and write some comments, such as which model is best at handling categorical features, which model is good at handling sparse features or not? which model overall generates the best results?

	Best RMSE Train Score	Best RMSE Test Score	Parameters for Best Model
Linear Regression	0.0883437430779	0.0885045028052	Lasso Regularizer, Alpha = 3.93515307e-05

Random Forest	0.07612258	0.07603504	n_estimators=100
			max_depth=4
			max_features=4

Neural Network regression	0.017285957	0.024586302	#hidden units = 176, activation = 'relu'
Linear Reg - diff workflows	Workflow 0: 0.04299 Workflow 1: 0.15962 Workflow 2: 0.04224 Workflow 3: 0.00711 Workflow 4: 0.10302	Workflow 0: 0.04302 Workflow 1: 0.15856 Workflow 2: 0.04224 Workflow 3: 0.00711 Workflow 4: 0.10297	
Linear Reg + polynomial	Workflow 0: 0.00976 Workflow 1: 0.02009 Workflow 2: 0.02241 Workflow 3: 0.00484 Workflow 4: 0.02529	Workflow 0: 0.01224 Workflow 1: 0.04882 Workflow 2: 0.02563 Workflow 3: 0.00579 Workflow 4: 0.03911	Best Degree: 7 Best Degree: 9 Best Degree: 9 Best Degree: 7 Best Degree: 8
Knn	0.036799	0.070366	k=3

We see that for handling categorical features, **linear regression with polynomial transformation** gave the best results on different workflows. From the above table we see that other cases of scalar encoding perform better than the standard linear regression, the best average RMSE for each workflow is less of linear regression with polynomial transformation hence giving a better fit. Above we saw that workflow ID is an important feature in predicting backup size. This conforms to our explanation as when we try to predict for a data point using only data points belonging to the same workflow ID, we get better results. And we are able to separately fit polynomial function for separate workflows to avoid any kind of underfitting or overfitting using this extra information (workflow ID) leading to the best model. The degree of polynomial associated with the best results is different for different workflows as some workflows can overfit and underfit at different values.

For one-hot encoded data, we see that the **neural network regression** gives us the best results. The activation function ‘relu’ gives us most optimal results. We convert the categorical data to numerical data which the neural network can understand and implement efficiently. It performs the best because it is able to generalize and avoid overfitting over the sparse representation of data. It also has more weights which allow more complicated models. Relu is performing better than the other 2 because it greatly accelerates the convergence of stochastic gradient descent as compares to logistic and tanh functions.

Neural network models work well for workflows 1 and 4 while for others we see that linear regression with polynomial works better. Overall we see that linear regression with polynomials won't be able to generalize as well as the neural network regression model. We can see the reason that the degree of polynomial associated with best results is different for different workflows and when we try to do altogether it might lead to underfitting or overfitting at a value of polynomial degree. Hence neural network regression gives us the best performance for the full model.

Conclusion

In this project, we utilised Regression analysis to estimate the relationship between a target variable and a set of potentially relevant variables. We explored techniques such as simple linear regression, random forests, neural networks and k nearest neighbor regression to predict the backup size of a file given the other attributes for the Network backup Dataset. We computed the RMSE scores over cross validations to measure the performance of our models.