EE219 LARGE SCALE DATA MINING

Project-4 REGRESSION ANALYSIS

Devanshi Patel	504945601	devanshipatel@cs.ucla.edu
Ekta Malkan	504945210	emalkan@cs.ucla.edu
Pratiksha Kap	704944610	pratikshakap@cs.ucla.edu
Sneha Shankar	404946026	snehashankar@cs.ucla.edu

INTRODUCTION

Regression analysis is a type of **predictive modelling technique** which investigates the relationship between a dependent and one or many independent variables. This technique is used for forecasting, time series modelling,etc. In this project we explore regression models like Linear , Random Forest, Neural network, Polynomial and K-Nearest neighbour regressions. We also analyze basic techniques to handle ill-conditioning and overfitting, namely Cross Validation and Regularization.

Dataset

The "**Network Backup DataSet**" consists of simulated traffic data on a backup system in the network. We develop prediction models for a system that monitors changes in a remote data, and copies over the changed data every 4 hours. The prediction models can be used to predict the size of data to be backed up as well as the time it takes for backing up.

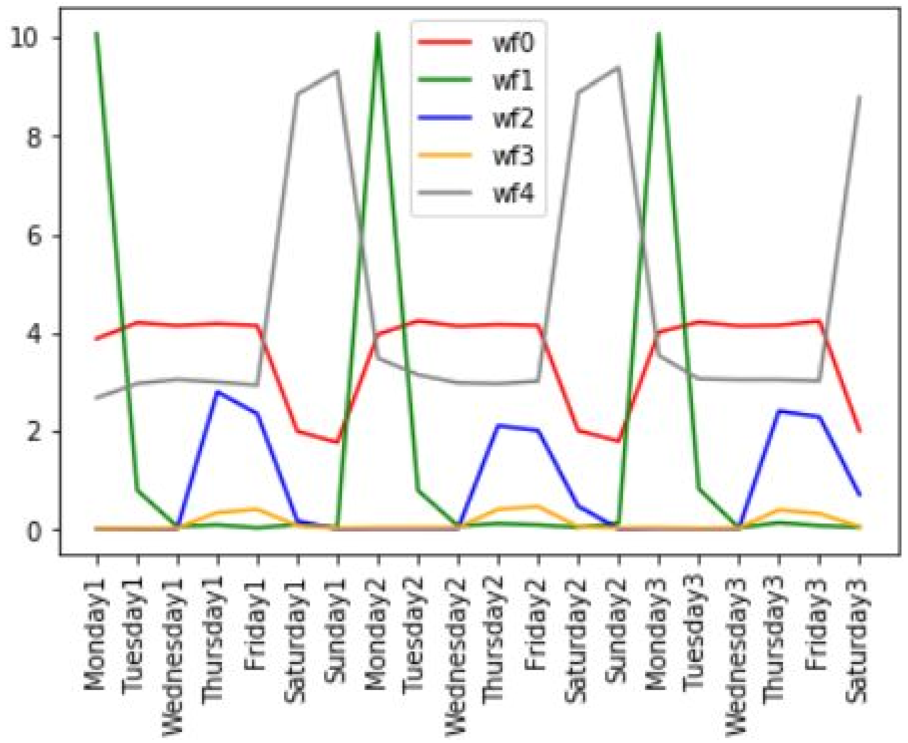
This dataset has 18000 data points with following attributes : Week ID, Day of Week, Backup start time, WorkFlow ID, File Name, Backup size, Backup time. Each Workflow process backs up data from a group of files with similar change patterns of size and time.

In this project, our primary purpose is to predict the backup size of a file given other attributes. All features except backup time are used as candidate features for prediction of backup size. To achieve this, we explore different methods and analyze.

We now run a preliminary tests on our dataset to understand the relationships between the underlying features in the dataset, before analyzing regression.

Task 1 : Plot and Analyze WorkFlow Patterns

a) For a twenty-day period (X-axis unit is day number) plot the backup sizes for all workflows (color coded on the Y-axis)

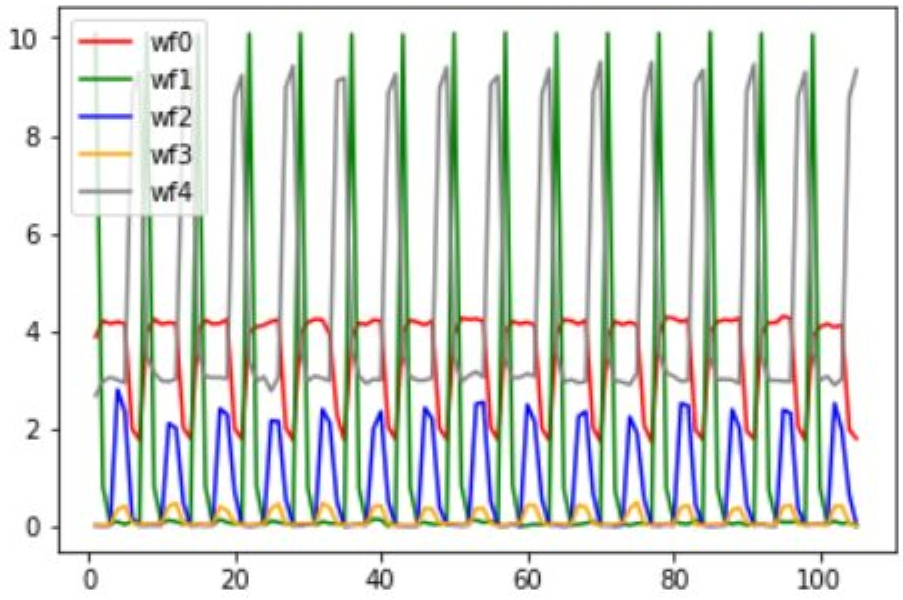


Backup sizes for 20 days of all workflows

From the above graph for 20 days, we can make the following observations:

1. For every workflow, we can observe some pattern in backup sizes
2. For workflow 0,the backup sizes drop during weekends
3. For workflow 1, Mondays have highest backup sizes and Tuesday has less compared to it. For remaining days of the week, there is very less backup taking place
4. For workflow 2, Thursdays and Fridays have the highest backup whereas remainder of the days have close to 0 backup
5. Workflow 3 has the lowest backup size amongst all the workflows.Workflow 3 has backups on Thursdays and Fridays and almost zero backups on the remaining days
6. Workflow 4 has highest backup size of weekends and consistent size for weekdays

B) Do the same plot for the rest 105-day period.



Backup sizes for 105 days of all workflows

From the above graph for 105 days, we can make the following observations:

1. The observations made in the first graph (of 20 days period) are still valid
2. There is a repeated pattern for all the workflows which was observed in the 20 day period.

Task 2 : Regression Analysis

Regression Analysis is an important tool for Data Modelling and Analysis. In Regression, we basically fit a line or a curve to the data points such that the distance between data points and the fitted curve is minimized.

a) Linear Regression

Linear Regression captures relationship between variables , by fitting a linear equation to the observed data. It does this by finding the best fit straight line also known as the **regression line.** Of these variables, one or more variables are the independent or explanatory variables, whereas the other variable is the dependant variable.

We analyze our data using Linear Regression for the below combinations , and for each combination we test for overfitting by using Cross Validation and ill-conditioning using regularization.

1. Regression over complete dataset
2. Regression over selected feature sets
3. Regression after Data standardization.
4. Different encoding schemes- Scalar encoding and One-Hot Encoding
5. All 32 possible combinations of feature vectors with Scalar and One-Hot Encoding.

6. Applying Ridge, Lasso and ElasticNet Regularization techniques and check for improvements over unregularized data.

Overfitting is an issue that occurs when your regression model learns too much of your training data, and cannot generalize or perform well on the test data. Using 10-Fold Cross Validation, we find Average Train Root Mean squared Error (RMSE) over 10 folds, and compare it with Average Test RMSE.

We have used ordinary least square method for finding the regression line in this project. It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no cancelling out between positive and negative values.

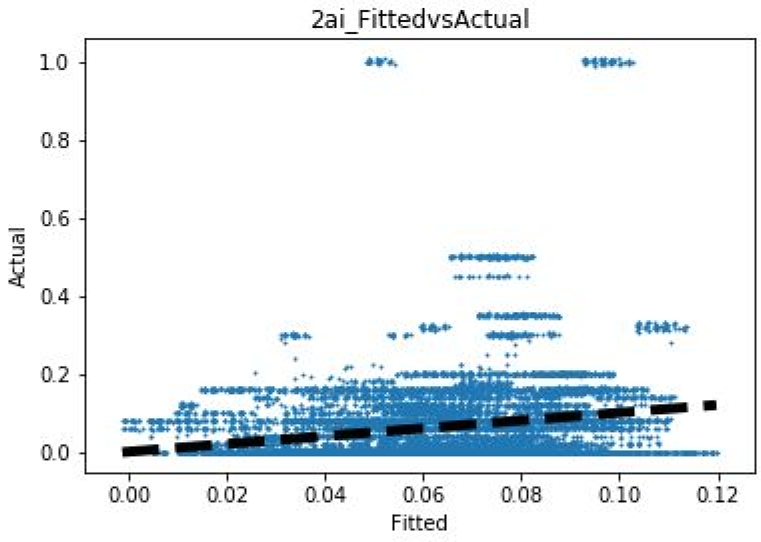
We have also checked our linear model for Heteroskedasticity, autocorrelation,etc in addition to overfitting.

i) Scalar Encoding, Fit and plot

The Data in the Network Backup Dataset has certain non-numeric attributes like ‘Monday’, ‘work_flow_3’, ’file_13’, etc. Performing mathematical calculations which are intrinsic to any model is difficult with non-numeric attributes.

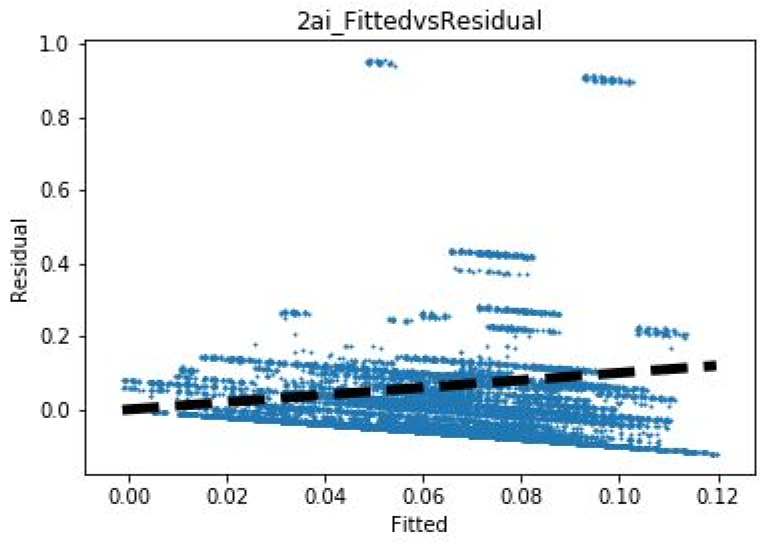
Average Train RMSE across 10 folds: 0.10183
Average Test RMSE across 10 folds: 0.10194

As we can see that train RMSE is less than Test RMSE, This shows **Overfitting** occured in our model. It implies that the model performs better on the train-data but not so well on the test-data.
In the below graph, we have plot Predicted Backup size values against the Actual Backup size values.



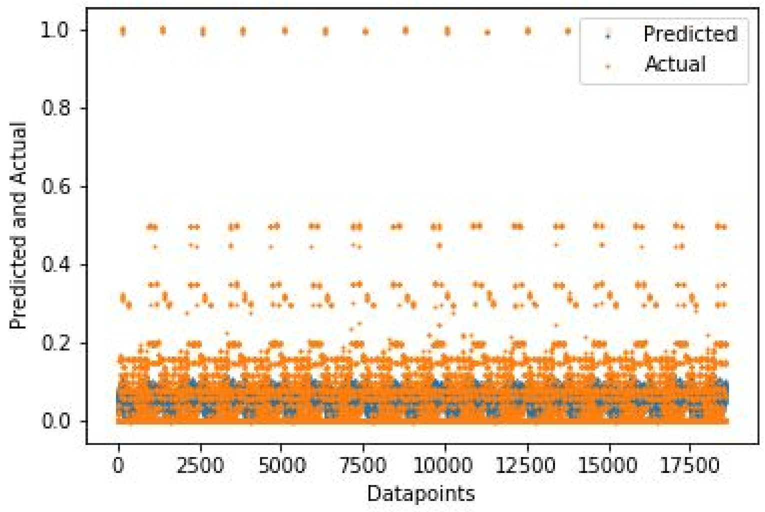
In the above plot we can observe that most of the data is separated with the regression line. However due to the presence of a few outliers, the regression line is affected and slightly tilted upwards. This tells us that Linear Regression is extremely sensitive to **Outliers.**

Now let us see the plot of Fitted vs Residual values:



In this plot, we see that regression line is again affected due to the presence of outliers. Positive values in the "The Residual Plot" above indicate that the predicted output was too low as compared to the actual, and negative values indicate that the predicted output was too high as compared to the actual.
We observe in this plot **that the vertical range of the residuals increases as the fitted values increases**. This forms a cone-shaped pattern, which is an indicator of "**Heteroskedasticity**" in our model. Heteroscedasticity means a systematic change in the spread of the residuals over the range of measured values. **Heteroskedasticity is a problem in our model because ordinary least squares (OLS) regression used here assumes that all residuals are drawn from a population that has a constant variance.**

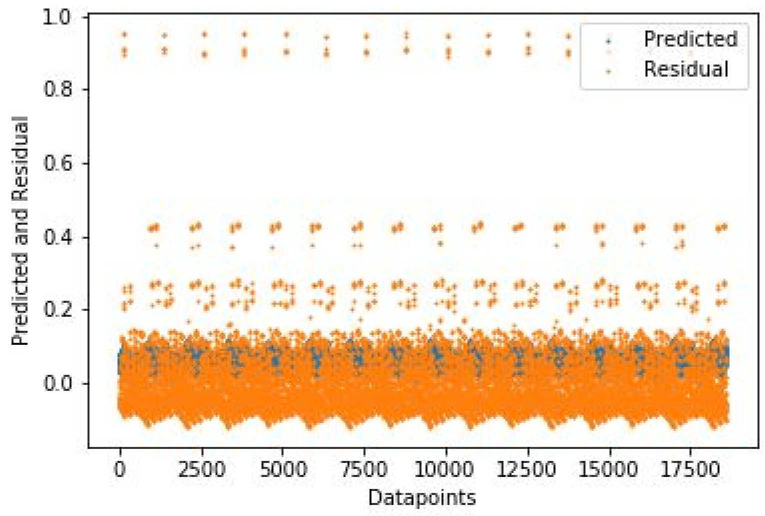
Now, we plot Predicted and Actual values for each of the 18588 data points. This would help us better evaluate our model .



In this plot, we see that the predicted values lie in the lower range of 0 to 0.1 GB, whereas the Actual values lie in the range of 0 to 1 GB. To cross-verify we printed the maximum values from our predicted and actual arrays as follows:

Maximum Actual Backup size from DataSet :	1.0088 GB
Maximum Predicted Backup Size after Regression :	0.1198 GB

This shows that predicted values using Linear regression are not good. To further confirm our analysis, we now plot Predicted values against Residual Values:



As we can see in this plot, the residual values are very high. The maximum residual value is almost close to 1, similar to the actual Backup size values**. This is because the predicted values were very low as compared to the actual, hence the residual values are almost same as actual values.**

The maximum residual value of backup size obtained was **0.956791200105** which is very close to the maximum actual backup size of **1.00882658176.**

We have implemented Durbin-Watson test to check for autocorrelation. This is available as part of statsmodels.stats.stattools package in python.

Durbin-Watson Statistic for autocorrelation : 0.4272

The test statistic value lies between 0 to 4, where 0 means positive serial correlation and 1 means negative serial correlation.

Thus we observed the following issues with ordinary Linear Regression above:

1. **It is sensitive to Outliers**
2. **HeteroSkedasticity exists in the model due to ordinary least square function used.**
3. **Autocorrelation was observed in the residuals, using Durbin-Watson Test.**

ii) Standardize the Data, Fit, Report Train and Test RMSE and Plot

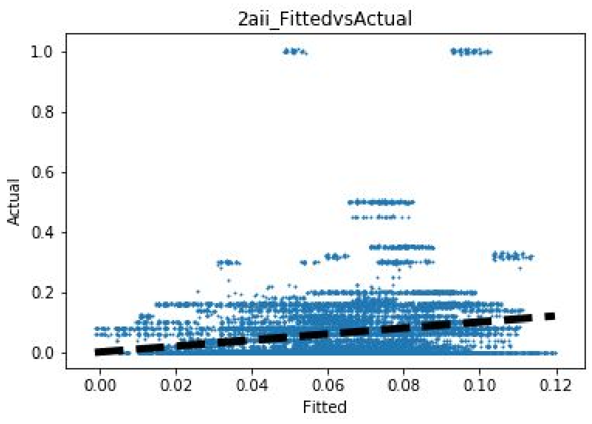
As we have observed that the results of linear regression are not so good. Now we check whether standardization of the data helps in this regard. We use sklearn's StandardScaler for this purpose.It standardizes features by removing the mean and scaling to unit variance.

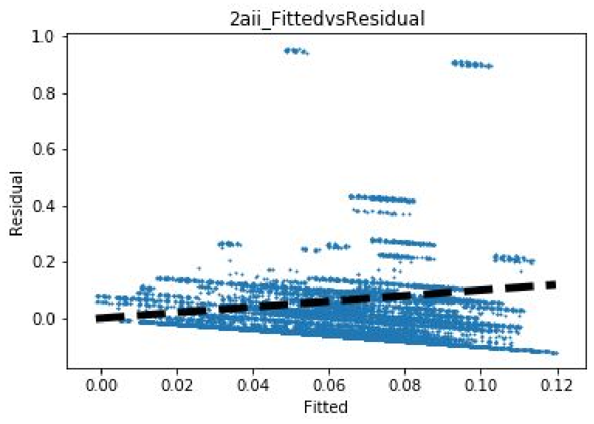
The intuition behind standardization is that some variables may assume high values, whereas other take very low values. For example in out Network Backup Dataset, The Day of Week features assumes 7 values, Backup start times assumes 6 values, workflow id assumes 5 values, filename assumes 23 values and so on.

The Average Train and Test RMSE for the 10-Folds for Linear Regression are :

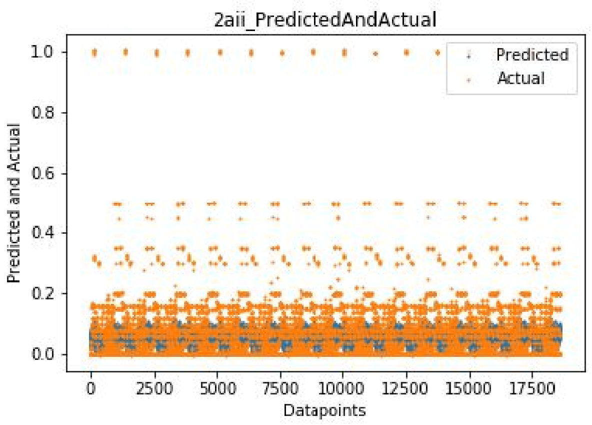
Average Train RMSE across 10 folds:	**0.10183**
Average Test RMSE across 10 folds:	**0.10190**

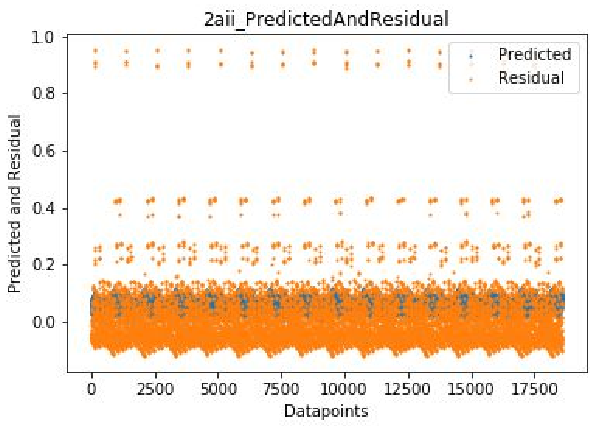
We observe that these standardized values are **very similar** to the average train and test RMSE for unstandardized dataset. To confirm our observation, we plot the graphs as before:





Also for plotting individual values against the data points, we get :





We observe that the graphs for standardized data are almost same to those of unstandardized dataset. This means that Standardization did not help us improve our results from regression. When we normalize the features we learn a linear model with normalized coefficient parameters on the new normalized features. However, this is effectively the exact same model, and thus the same accuracy. That’s why you don’t get a different result on standardization.

This means that Linear Regression is invariant to standardization of features.

iii) Feature Selection with f_regression and Mutual Information Regression

Till now, we have considered all the 5 features. In this task we identify top 3 features in our model, and fit the model with those features.

F_regression is a Linear model for testing the individual effect of each of many regressors. It first finds the correlation between each feature and the target. It then provides and F-score to each feature.

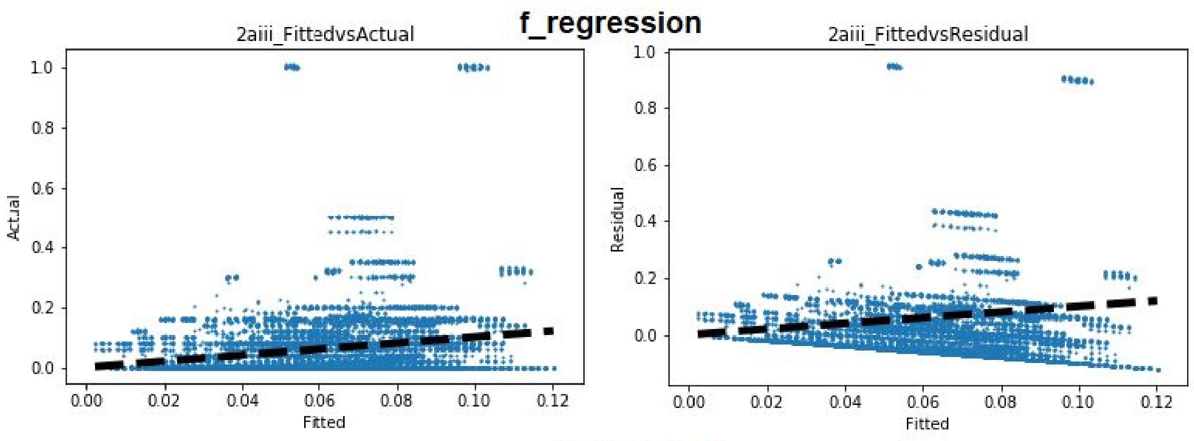
As per f_regression, the top 3 features are 5, 2, 3 i.e.
FileName , Day Of the Week, Backup Start Time-Hour of the day

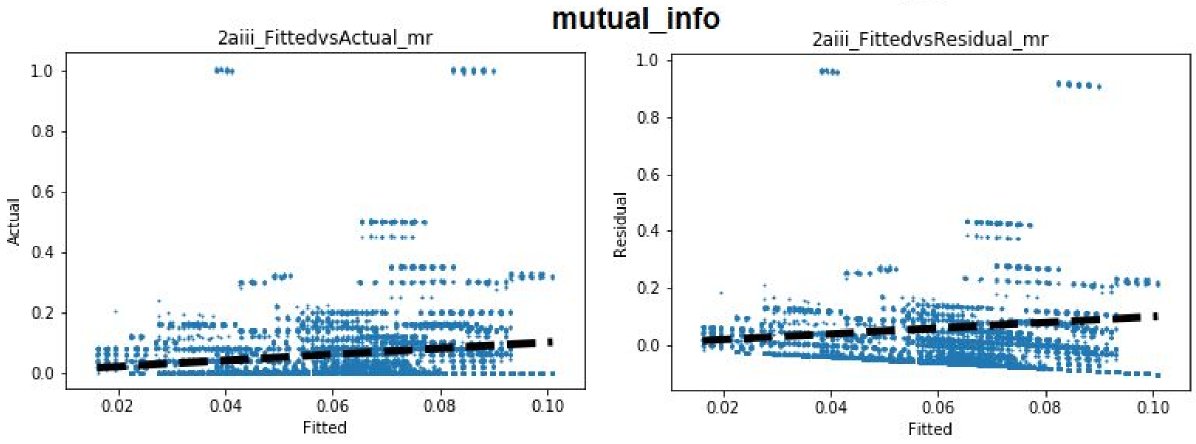
Mutual Information Regression estimates mutual information(dependency) for a continuous target variable.

As per Mutual Regression top 3 features : i.e.
Work-Flow-ID, FileName,Backup Start Time-Hour of the day

The performance after feature selection is as follows:

	All 5 features	Standardized	f_regression	mutual info regression
Average Train RMSE	0.10183436	0.10183388	0.10187197	0.10254729
Average Test RMSE	0.10193945	0.10190086	0.10189343	0.10246541



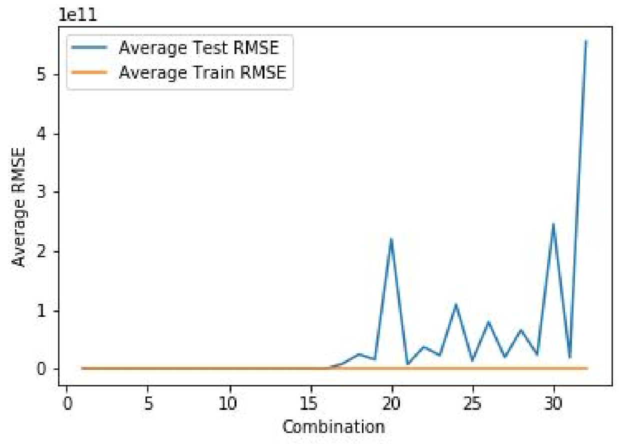


We observe that test RMSE for f_regression is lower than mutual_info regression. Hence f_regression is a better regularization for linear regression than mutual_info regression.
F-regression captures only linear dependence, whereas mutual info regression captures any dependency between variables.
On comparing with all the scores till now, F_regression seems to provide the lowest Test RMSE. Hence it is advisable to use f_regression in a Linear Regression model.

iv) Generating all 32 possible combinations of Scalar and OneHot encodings for 5 features , and then plot Train and Test RMSE these combinations

One-Hot Encoding transforms each categorical feature with n possible values into n binary features, with only one active. The Day of the Week, Monday could be encoded as [1; 0; 0; 0; 0; 0; 0] and Friday as [0; 0; 0; 0; 0; 0; 1]

The average train and test RMSE for each 32 combinations are as follows:

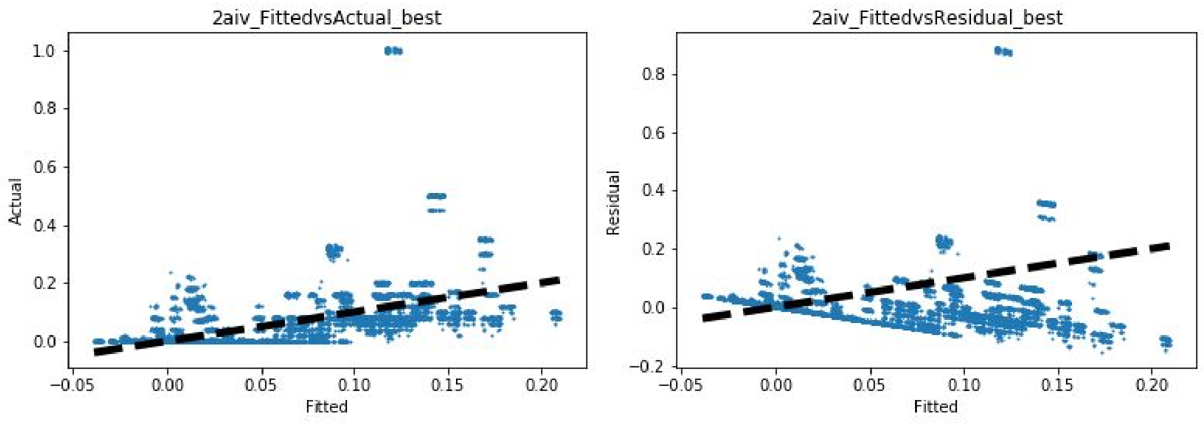


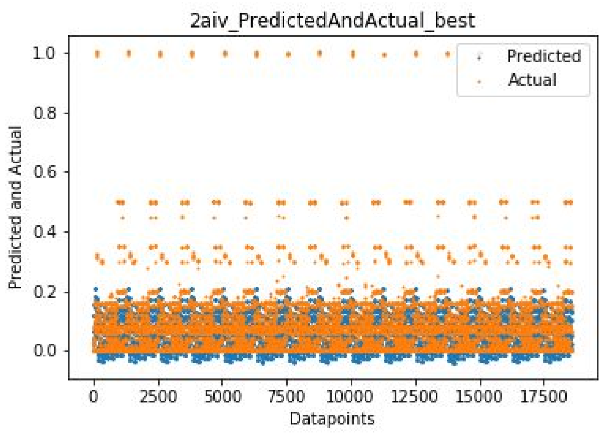
From the plots,can see that the average train and test RMSE is lowest for the combination when WeekID is Scalar and Rest All OneHot. Also we observe extremely high test RMSE whenever WeekID is encoded as OneHot.

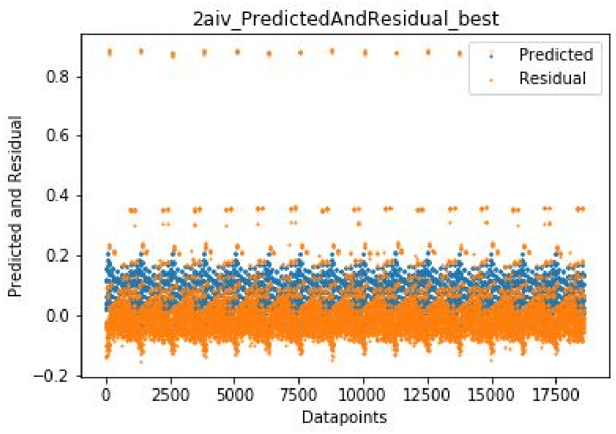
Scalar Encoding results in assigning a numeric value to labels. This results in ordinal relationship between variables, as numbers are smaller or greater to each other intrinsically. OneHot Encoding helps to encode labelled features numerically, without any **ordinality**.

In our data, **WeekID feature** has an ordinal relationship between values, hence it should be **scalar.** Which Week of the period the data is being backed up is important. As in the initial weeks, more data will be backed up, and this would reduce as time progresses, because only small modifications to existing data are now being backed up. **For other features like Day of the week, Backup Start Time, WorkFlowID and FileName, the categories do not have any ordinal relationships with each other, hence having onehot encoding would be best for them.**

The plots for the best model are as follows:







As we can see from the plots above, the plots have improved after doing encoding as compared to earlier plots. Also, the average test RMSE obtained is better as compared to only Scalar encoding of all variables.

v) Controlling ill-conditioning and overfitting

In the previous question we observed very high test RMSE values for the combinations in which WeekID was onehot encoded. We also saw the reason for this. We now apply Ridge, Lasso and ElasticNet Regularizers on our data, and identify the best models with lowest test RMSE. Regularization helps to combat overfitting by adding a penalty term to the target function being optimized.

Ridge Regularizer uses **L2** regularization. It includes a shrinkage parameter called alpha. This is added to least square term in order to shrink the parameter to have a very low variance. It does not shrink the coefficients to 0, which means no feature selection is carried out. Ridge assumes Normality of Residuals as OLS.

Lasso Regularizer uses **L1** regularization. It includes a shrinkage parameter called alpha. It shrinks the coefficients to 0, which means feature selection is carried out. If group of predictors are highly correlated, lasso picks only one of them and shrinks the others to zero. The assumptions in Lasso are same as OLS and Ridge, however it does not assume normality of residuals as them.

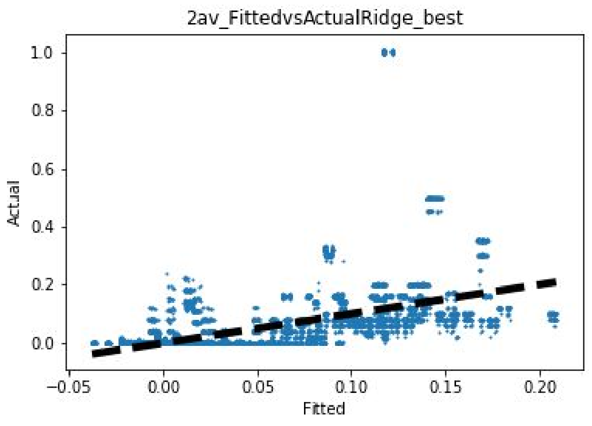
ElasticNet Regularizer uses both L1 and L2 regularization. It includes two shrinkage parameters, lambda1 and lambda2 corresponding to L1 and L2. ElasticNet is a hybrid of Lasso and Ridge techniques. It is used when multiple features are correlated. It encourages group effect in case of correlation between features. However it can suffer with double shrinkage.
We have used **RidgeCV, LassoCV, and ElastinetNetCV** from pythons sklearn package. These methods automatically scan over different values of hyperparameters like alpha and chose the best alpha for a given combination.

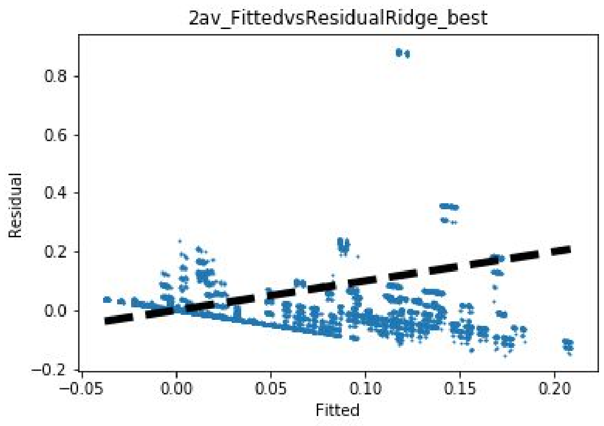
For each of the above regularizations we found the best model with the same combination : (S,O,O,O,O)
Below is the comparison of the three models , along with their best parameters and Average Test RMSE.

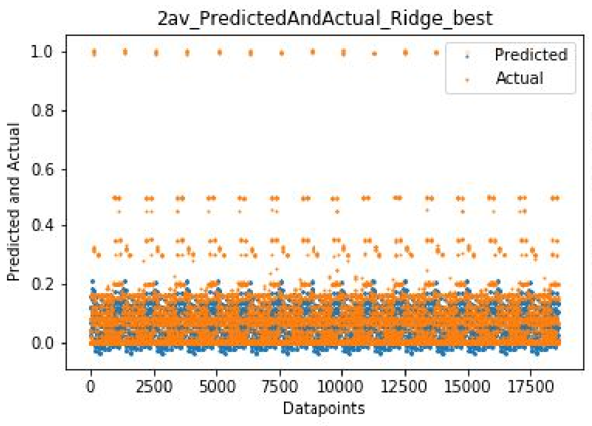
	unregularized	Ridge	Lasso	Enet
Best alpha		1.0	0.00002252	
lambda1				0.00002415
lambda2	NA			0.00001207
Best Average Test RMSE	0.088508239	0.088505399	0.10193932	0.08850448

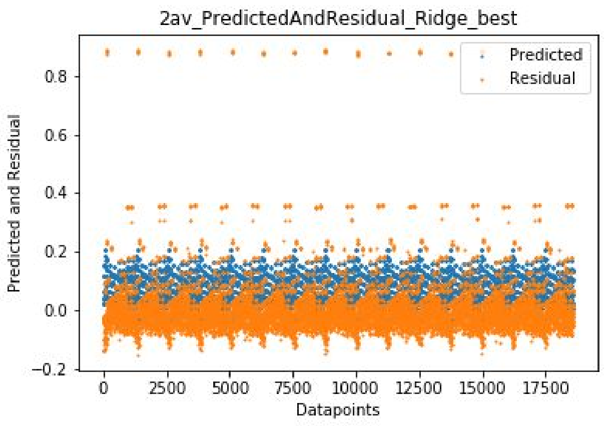
As we can see from the table able, ElasticNEt Regularization provides the best lowest Test RMSE.

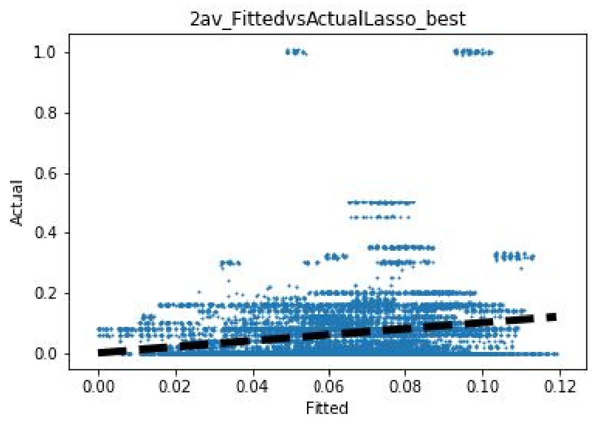
The plots for the best models are as follows:

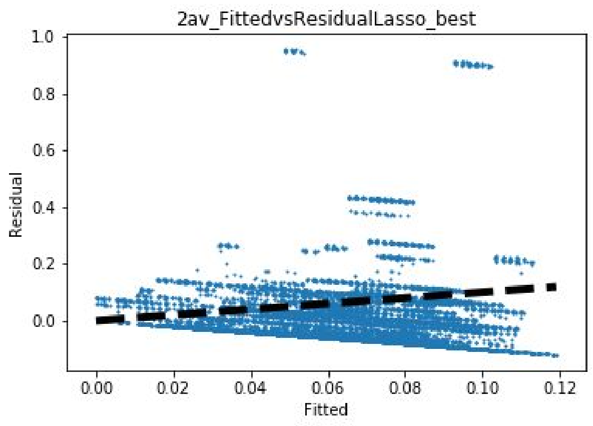


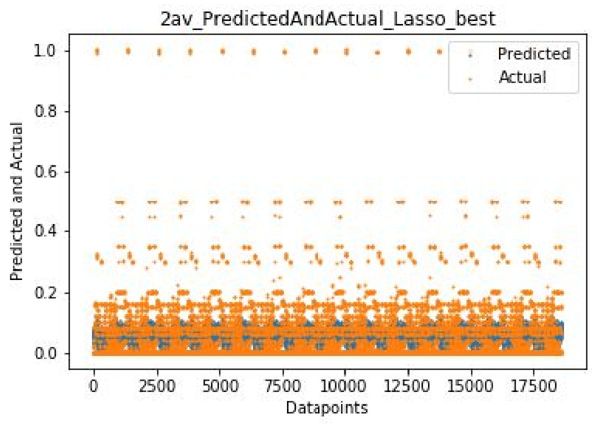


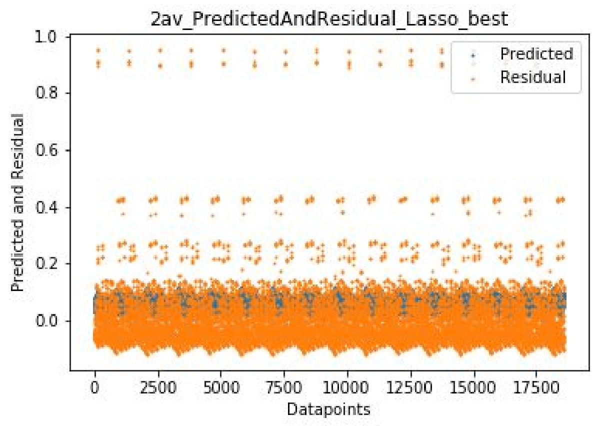


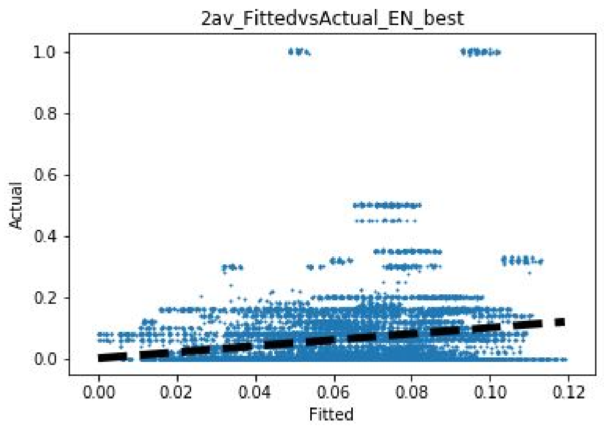


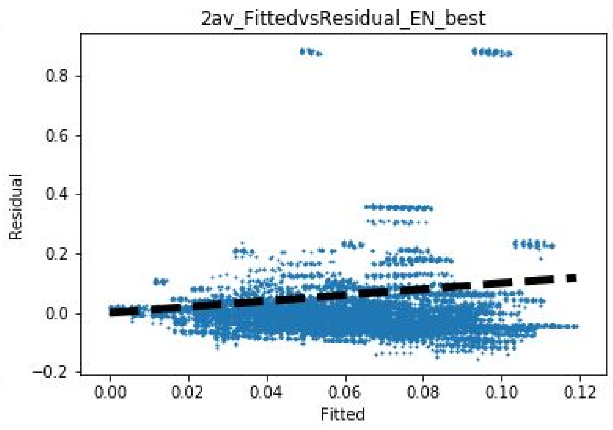

2av_FittedvsActualLasso_best
Actual
Fitted

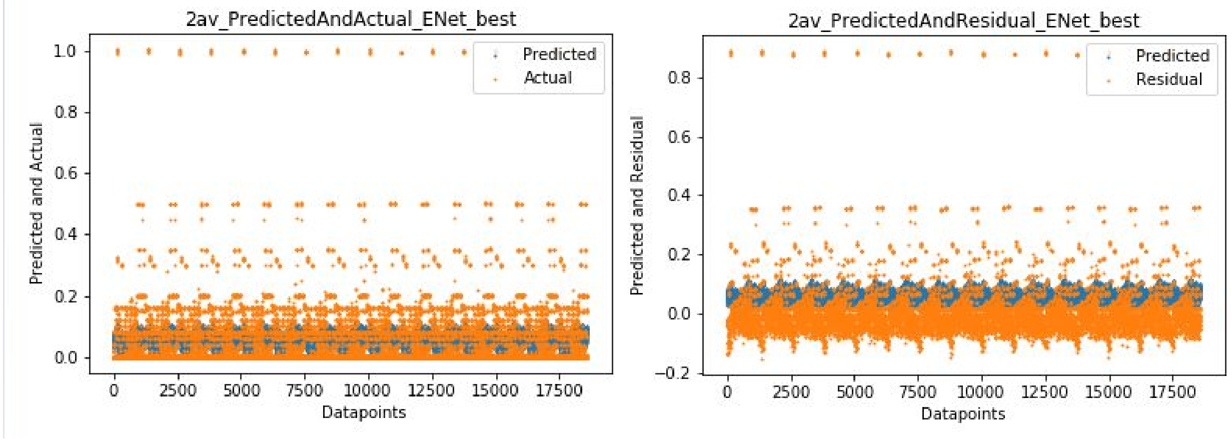

2av_FittedvsResidualLasso_best
Residual
Fitted


2av_PredictedAndActual_Lasso_best
Predicted and Actual
Datapoints
Predicted
Actual


2av_PredictedAndResidual_Lasso_best
Predicted and Residual
Datapoints
Predicted
Residual


2av_FittedvsActual_EN_best
Actual
Fitted


2av_FittedvsResidual_EN_best
Residual
Fitted

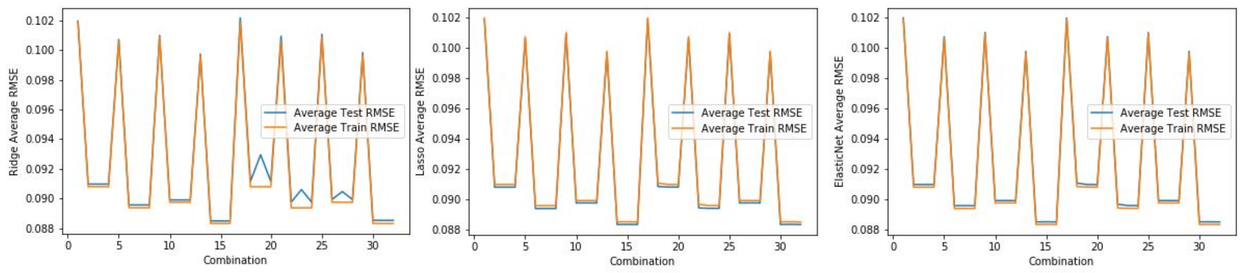


From the above graphs we can see that following:

1. The Predicted values for Ridge are better than unregularized graph, and residuals are lower.
2. The Lasso and Elastic Net regularizers give better accuracy than Ridge.
3. ElasticNet, which is a hybrid of Ridge and Lasso, always gives the best accuracy of all the above.
4. The best combination of encoding obtained for each of the models is :

WeekID: Scalar, **DayOfWeek**:OneHot, **BackupStartTime**:OneHot,**WorkFlowID**:OneHot, **FileName**: OneHot

We now compare the average train and test RMSE for each regularizer :



From the graph we can see that

1. Ridge had some higher test RMSE than Train RMSE, which means a little amount of Overfitting still exists.
2. For both Lasso and ElasticNet achieve almost similar Test RMSE which match their respective Train RMSE , hence no **overfitting.**

Thus we have successfully removed overfitting from our model by Regularizing it.

This can be verified by our table above as well.

We even found coefficients for each regularized best model and for the unregularized best model. On Comparing we found as follows:

1. For unregularized model, the coefficients have extremely high positive and negative value in the range -30421517800.00 to 32628198700.00. A high intercept of 5479706680.90 was observed.

2. For Ridge Regularizer, the coefficients obtained were non-zero, with positive and negative values in the range from -0.04880710 to 0.06114733. An intercept of 0.06082282 was observed.
3. For Lasso, many coefficients obtained were zero. The values ranged from -0.04442886 to 0.08361318. The values were generally lower as compared to Ridge. An intercept of 0.035244189 was observed.
4. For Elastic Net, again many zero-value coefficients were obtained. The values ranged from -0.04442477 to 0.08360132 and were lower than all the other models namely unregularized, Ridge, and lasso. An intercept of 0.04498755 was obtained, which is lower than all other models.

The above observations can be interpreted as follows:
Intercept of a linear equation is the value that the dependent variable will have when all the other independent variables have zero value. We see a very high negative intercept is present for unregularized model. For Ridge, this value decreases considerably to 0.06. It further decreases for Lasso and Elastic Net to 0.035 and 0.04. However since we have used one hot encoding for some features, it is highly unlikely that all the 49 feature values (obtained by combining scalar and one how encoding for the best model) will be zero at the same time. Hence for our case, intercept does not have any intuitive interpretation. It just anchors the regression line to the right place.
The coefficients in the linear regression mean that by how much will the backup size increase or decrease per unit change in the value of that variable. Extremely high positive and negative values in unregularized model indicates that for each unit change in the value of a variable, the backup size would change by a high value. This results in inaccurate predictions in the linear regression model. After regularization , a lot of improvement can be seen. The coefficients in Ridge are very low , meaning every change in value of categorical features will not highly impact the backup size. Lasso and ElasticNet coefficients even more accurately capture the dependency between backup size and each encoded categorical variable. Hence many of the coefficients are zero, for the onehot encoded features, and only important features are retained. This means they perform feature selection and regularization both, resulting in highly accurate predictions.

b) Random Forest Regression Model

In this section, we build a random forest regressor to predict the backup size of a file. The essence of this regression is based up on the working of a decision tree. Many such decision trees make up a forest. Random Forests is an ensemble method of classification/regression and they construct a number of decision trees at the time of training and gives an output class which is the mode of the classes (classification) or an output which determines the mean prediction of individual trees (regression). Here, since we use the Random Forests for regression, we get the prediction information as required.

So how does a decision tree work?
A decision tree breaks the dataset into smaller subsets by making a decision and builds an associated decision tree incrementally during the training phase. Here, the decision is made based on the most important feature at that instant. The final result is thus a tree with leaf nodes and decision nodes (non-leaf nodes). Thus, in our case, the decision node has two branches. The leaf nodes all represent a decision. Since every decision is made based on the best feature, it is obvious that the topmost node or the root should be the best feature of that decision tree.

Feature importance in random forest algorithm

We mentioned above that the branching decision is made based on the most important feature. But how does the algorithm determine which is the most important feature? This is where measures of impurity comes into picture. The algorithm picks up the feature whose measure of impurity is the least. If we use the random forest model for classification, then the decision is made by using the Gini impurity or the Entropy/Information Gain for each feature. On the other hand, for regression, the decision relies on the variance/standard deviation.

Decision tree algorithm for regression:

a) Using Standard Deviation

A decision tree for regression uses standard deviation/variance to calculate the homogeneity of a numerical sample. If the sample is completely homogenous, then its standard deviation is zero.

b) Standard Deviation Reduction

This step aims at choosing the feature which has the highest standard deviation reduction. Intuitively, the reduction is based on the decrease in standard deviation after the dataset is split on a feature. Thus, the decision and hence the construction of the decision tree is based on the feature which has the highest standard deviation reduction i.e. the most homogenous branches. Following are the steps which the algorithm undertakes for constructing a decision tree based on standard deviation reduction.
Step 1: The standard deviation of a target is calculated
Step 2: The dataset is then split on different attributes. Then the standard deviation for each branch is calculated. This is then subtracted from the standard deviation before the split. The result obtained is termed as the standard deviation reduction.
Step 3: The algorithm then selects the feature with the largest standard deviation reduction and creates a node for it.
Step 4: The branching then takes place and again, the dataset is divided based on the selected feature. A branch with a standard deviation more than 0 needs to be splitted further. Thus, the algorithm is recursively run on these decision nodes (non-leaf) till all data is processed. Once everything is processed, we get the leaf nodes of the tree which signifies the prediction done by the algorithm. The algorithm then terminates.

Bootstrapping

Bootstrap ***agg***regat***ing*** is also called as ***bagging*** and is used to improve the stability and accuracy of algorithms in classification and regression. In our case, we use bagging in our decision tree algorithm for regression.
So how does bagging work?
Suppose we have a training set D of size n. The bagging method will then create m new training sets Di of size n’ each. This is done by uniform sampling and replacement from D itself. But, by using the sampling with replacement technique, some observations may be repeated in each Di. If n’=n, then for large n, then the set Di is expected to contain the fraction *1-1/e* (approx 63.2%) of the unique samples of D. The rest of them will be duplicates. This kind of sample is called as the bootstrap sample. Then, the m

training sets/models are fitted using the m bootstrap samples (samples are created for each set Di) and then combined by averaging the output for regression.

Out-of-bag error
Since, we are using bagging (bootstrap aggregating), it is pretty intuitive that there would be some error associated with it too. This error is called as the Out-of-bag (OOB) error. It measures the prediction error of our random forests. OOB is actually the average prediction error on each of the data point used for training, using only the trees in our forest that did not have this data point in their bootstrap sample. In our regression model, we have the oob_score_ which returns the out-of-bag score. The out-of-bag error is then calculated by the expression *1-oob_score_.*

i) We now create our random forest regression model with the following parameters:

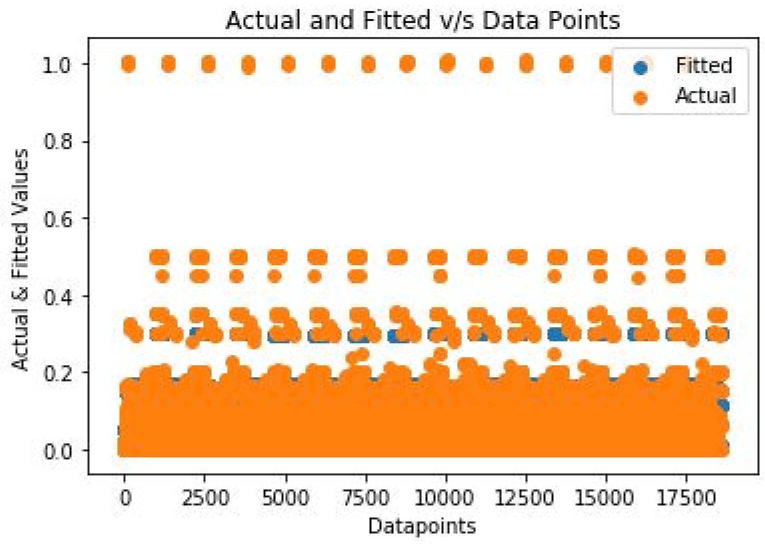
- Number of trees = 20
- Depth of each tree = 4
- Bootstrap = True
- Max features = 5

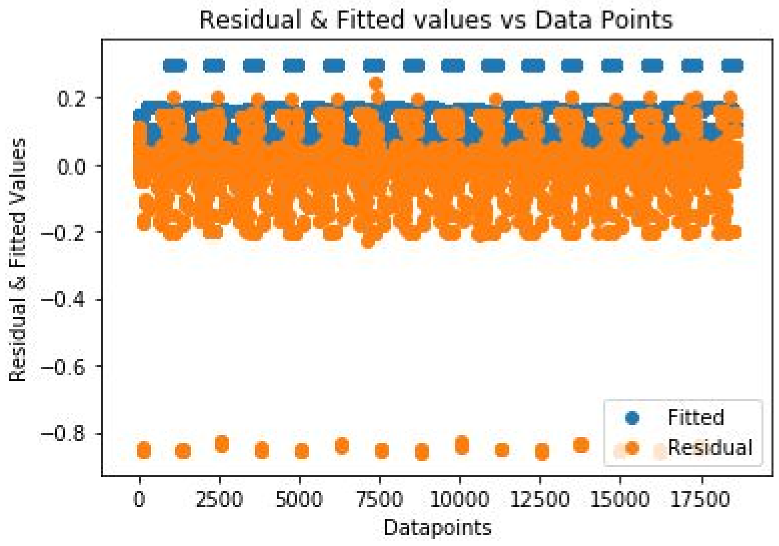
We have used scalar encoding to encode the five features ('Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name'). After doing scalar encoding, each categorical variable is converted into a one dimensional numerical value.
We then used the 10 fold cross validation and for the average training and testing RMSE. Here, we got the mean squared error of each fold using sklearn's mean squared error package (i.e. squared errors divided by number of data). Then we took the mean of all such values obtained for each fold and took the square root to get the final root mean squared error (RMSE value).
We also calculated the Out-of-bag error using *1-oob_score_* for the regression model we created.
Observations:

Mean Training RMSE	**Mean Testing RMSE**	**Out-of-bag error**
0.07595	0.07618	0.5408

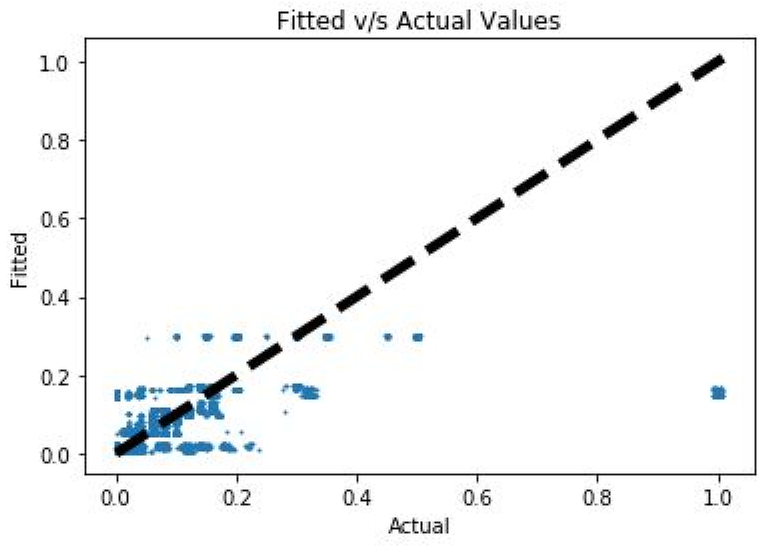
We then plotted two plots:

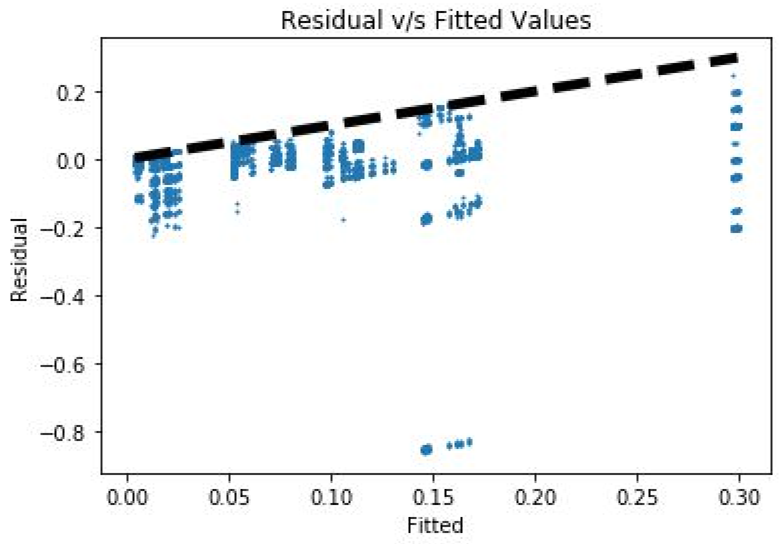
- Fitted and actual values versus the data-points.
- Fitted and residual values versus the data-points.





We have also plotted actual v/s fitted values and fitted v/s residual values. They can be observed as follows:





Analysis:
We observe that the fitted and the actual values overlap in a fairly decent way which is a good sign signifying good prediction output (though not the best). We see that the residual values are concentrated near to 0 or below them. Since residual value is the difference between the actual and fitted values, getting the residual values near and around zero is a good sign of prediction. However, since there is a no perfect overlap between actual and fitted values, as well as there are some residual values below -0.2, we can't claim that this is the best prediction model. Though this model has done a fairly decent prediction, we can further tune the parameters we supply to the model so as to get better plots than these.

ii) We now sweep over number of trees from 1 to 200 and analyze the results. We also increase the maximum number of features from 1 to 5 for a particular number of trees in the forest and infer from the same. Inference is drawn based on the out of ba error and the test RMSE observed in each case. Here, we have kept the depth of each tree as 4.

Observations:

Max features	#Trees	Depth	OOB error	Mean training RMSE	Mean testing RMSE
1	1	4	1.15127	0.09443	0.09491
2	1	4	1.13703	0.08882	0.08892
3	1	4	1.07337	0.07844	0.07875
4	1	4	1.06527	0.0764	0.07671
5	1	4	1.06502	0.07635	0.07666
1	5	4	0.8449	0.08948	0.08965
2	5	4	0.71185	0.07851	0.07902
3	5	4	0.65863	0.07654	0.07694
4	5	4	0.62763	0.07577	0.07607
5	5	4	0.62833	0.07633	0.0766
1	10	4	0.74239	0.08804	0.08881
2	10	4	0.58049	0.07636	0.07699
3	10	4	0.57472	0.07666	0.0772
4	10	4	0.53262	0.07342	0.0736
5	10	4	0.55393	0.07633	0.07659
1	20	4	0.7075	0.08675	0.08739
2	20	4	0.56954	0.07784	0.07849
3	20	4	0.53599	0.07554	0.0761
4	20	4	0.51761	0.0737	0.07396
5	20	4	0.5408	0.07595	0.07618
1	30	4	0.7051	0.08707	0.08777
2	30	4	0.56102	0.07752	0.07806
3	30	4	0.52112	0.07487	0.07535
4	30	4	0.50657	0.0735	0.0737
5	30	4	0.53212	0.07577	0.07597
1	40	4	0.68997	0.08648	0.08721
2	40	4	0.57214	0.07848	0.07903

3	40	4	0.51954	0.07487	0.07534
4	40	4	0.50684	0.0736	0.07382
5	40	4	0.53453	0.07574	0.07595
1	50	4	0.67746	0.0856	0.08629
2	50	4	0.58002	0.07899	0.07956
3	50	4	0.52591	0.07524	0.07577
4	50	4	0.50061	0.07341	0.07362
5	50	4	0.53561	0.07575	0.07597
1	100	4	0.68331	0.08593	0.08659
2	100	4	0.56755	0.07855	0.0791
3	100	4	0.52531	0.07491	0.07535
4	100	4	0.50051	0.07336	0.07359
5	100	4	0.53142	0.07581	0.07604
1	150	4	0.68805	0.08585	0.0865
2	150	4	0.57236	0.07886	0.07943
3	150	4	0.52368	0.07483	0.07524
4	150	4	0.49454	0.07301	0.07324
5	150	4	0.52914	0.07577	0.07601
1	200	4	0.68812	0.08589	0.08651
2	200	4	0.57705	0.07901	0.07957
3	200	4	0.52603	0.0751	0.0755
4	200	4	0.49311	0.07292	0.07315
5	200	4	0.53192	0.07588	0.07612

Figure 1: Out-of-bag error v/s Number of trees

Figure 1: Out-of-bag error v/s Number of Trees

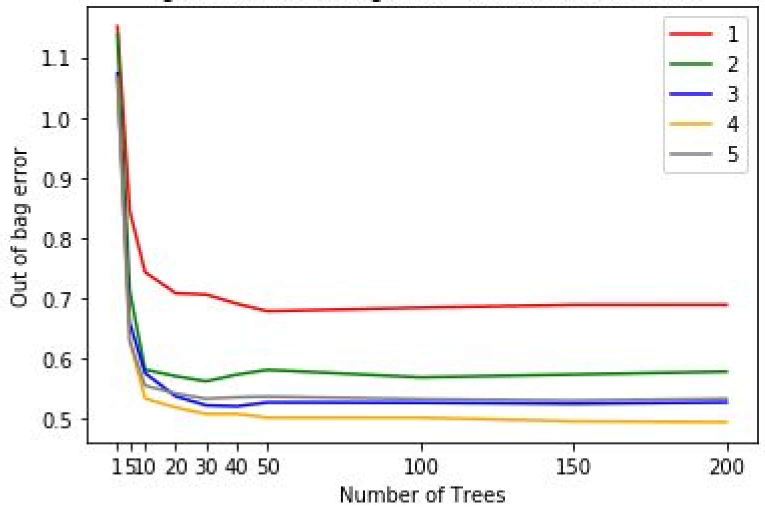
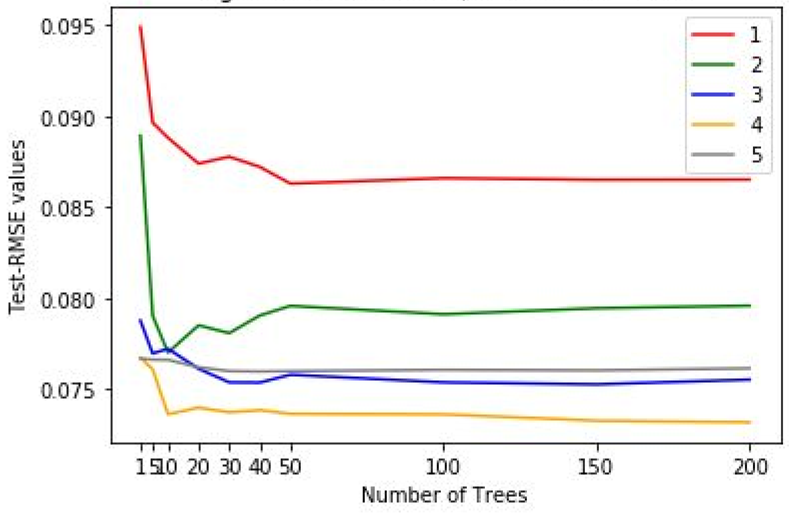


Figure 2: Test-RMSE v/s Number of Trees



Analysis:

- As we increase the number of max features supplied to the regression model from 1 to 4, the Out-of-bag error and Test-RMSE, both decrease for a given number of trees in the forest. Both the errors increase by a slight amount if we set the max features to 5
- The Out-of-bag error drastically reduces as we increase number of trees from 1 to 10. From there on, any increase in the number of trees, decreases the OOB error by a slight amount
- The Test-RMSE also decreases drastically as we increase the number of trees from 1 to 10. Any increase in the number of trees beyond 50, decreases the Test-RMSE only by small amount

Thus, we see that these two factors are determining the predictive power of our model.

➔ Max features: Increasing this parameter means we now have a higher number of options to be considered while making a decision. Thus, it improves the performance. However, this claim is not completely true as we increase the number of max features. This is mainly because this decreases the diversity of an individual tree in our forest which is considered as the USP of the random forest regression model. In our case, this case was observed when we set the max_features to 5 since the oob error and the test RMSE increased as we increased the max_features from 4 to 5. Another observation was that the speed of the algorithm decreased as we increased the number of max features.

➔ Number of trees: We observed that higher number of trees give us better performance. Better prediction results were received for higher number of trees because using more trees reduces the variance and we are using variance reduction as a measure of branching decision. But more number of trees make our code slower. Therefore, it is recommended that one should always choose only that many number of trees as the processor can handle because this makes our predictions more stronger and stable.

These two points are easily verifiable from the table above since we get the least OOB error and least Test-RMSE when we set the maximum number of features as 4 and number of trees as 200.

iii) Now, we monitored the prediction of our model by varying the depth of trees in our forest. At first, we chose a fixed number of trees to check at which depth we get the best results. Then, we increased the number of trees and used this depth to determine which combination gives the best results.

Firstly, we set the number of trees to 50 in our forest and observe the predictions for varying depth of these trees as also by altering the maximum number of features supplied to the regression model.

Observations:

Max Features	Trees	Depth	OOB	Mean training RMSE	Mean testing RMSE
1	50	4	0.67746	0.0856	0.08629
2	50	4	0.58002	0.07899	0.07956
3	50	4	0.52591	0.07524	0.07577
4	50	4	0.50061	0.07341	0.07362
5	50	4	0.53561	0.07575	0.07597
1	50	6	0.47492	0.07232	0.07506
2	50	6	0.29746	0.057	0.05885
3	50	6	0.19911	0.04622	0.04801
4	50	6	0.14217	0.03924	0.03955
5	50	6	0.15255	0.04004	0.04015
1	50	8	0.27647	0.05439	0.06201
2	50	8	0.10801	0.0328	0.03707

3	50	8	0.04182	0.02046	0.02244
4	50	8	0.02041	0.01373	0.01455
5	50	8	0.01606	0.01272	0.01353
1	50	10	0.13172	0.03528	0.04918
2	50	10	0.04222	0.01807	0.0252
3	50	10	0.01795	0.01215	0.01629
4	50	10	0.01531	0.0112	0.0132
5	50	10	0.0151	0.01102	0.0131
1	50	15	0.06036	0.01216	0.04213
2	50	15	0.02195	0.00834	0.02098
3	50	15	0.01854	0.00735	0.0158
4	50	15	0.01748	0.00711	0.01441
5	50	15	0.01776	0.00702	0.01445
1	50	20	0.06371	0.0101	0.0399
2	50	20	0.02592	0.00653	0.02443
3	50	20	0.01928	0.00558	0.017
4	50	20	0.0183	0.00536	0.015
5	50	20	0.01857	0.00541	0.01493

Figure 1: Out-of-bag error v/s Depth of Trees

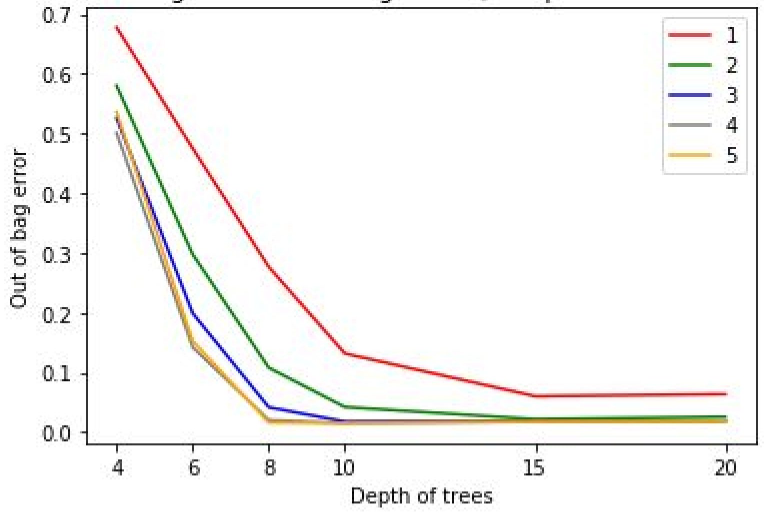
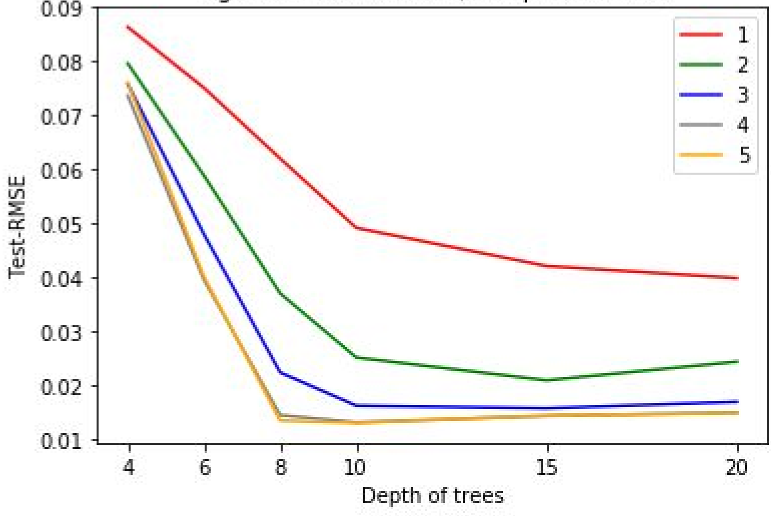


Figure 2: Test-RMSE v/s Depth of Trees



Analysis:

- As we increase the depth from 4 to 8, the OOB error and Test-RMSE decrease drastically. They further decrease from 8 to 10. Beyond 10, the results vary negligibly for each figure as we keep on increasing the depth. **Thus, the best predictive performance is observed at depth=10.**
  The reason: It is observed that the tree depth should be sufficient enough to split each node into desired number of observations. In our case, we observe this at depth=10. By intuition, limiting the depth, makes the ensemble converge a little earlier.

- We also observe that the difference between the errors is very negligible when we increase max_features from 4 to 5.

We now vary the number of trees (all with depth 10) in our forest and identify the parameters which yield the best predictive performance. Here, we also know by the previous analysis that the results obtained with max_features 4 and 5 are better than lower values of max_features supplied to the model. So, now we record observations by varying number of trees from 40 to 200 (we already know that 200 gives the best results), max features 4 and 5 as well as fixing the depth value to 10.

Observations:

Max Features	Trees	Depth	OOB error	Mean training RMSE	Mean testing RMSE
4	40	10	0.01564	0.01121	0.01328
5	40	10	0.01515	0.01102	0.01312
4	50	10	0.01531	0.0112	0.0132
5	50	10	0.0151	0.01102	0.0131
4	100	10	0.01488	0.01116	0.0131

5	100	10	0.015	0.01101	0.01308
4	150	10	0.01483	0.01115	0.0131
5	150	10	0.01498	0.011	0.01308
4	**200**	**10**	**0.01477**	**0.01115**	**0.01307**
5	200	10	0.01495	0.01101	0.01307

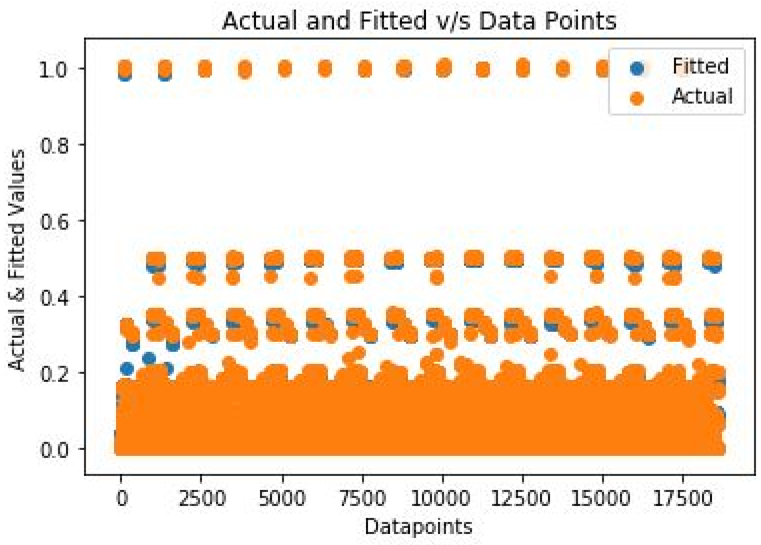
Analysis:

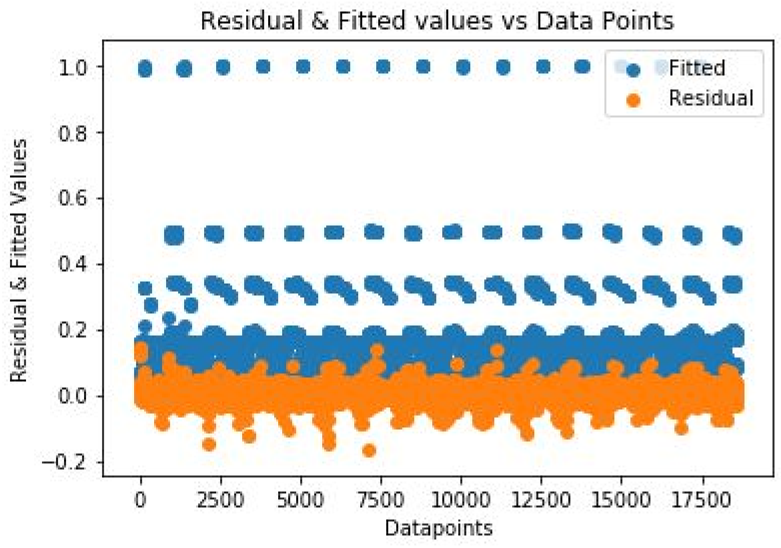
- We see that by fixing number of trees to a particular value and the depth to 10, the results are almost the same for max_features as 4 and 5. The same was observed in the previous analysis also
- By increasing the number of trees, the predictive power of the model increases slightly every time (for depth=10). Thus, the best prediction is observed with a forest of 200 trees
- The Test-RMSE for a forest with 200 trees each with depth 10 is the same (or almost the same since the values have been rounded off to 5 decimal places) for max_features as 4 and 5. But the out-of-bag error is lesser when we supply a maximum of 4 features to the model than when we supply a maximum of 5 features.

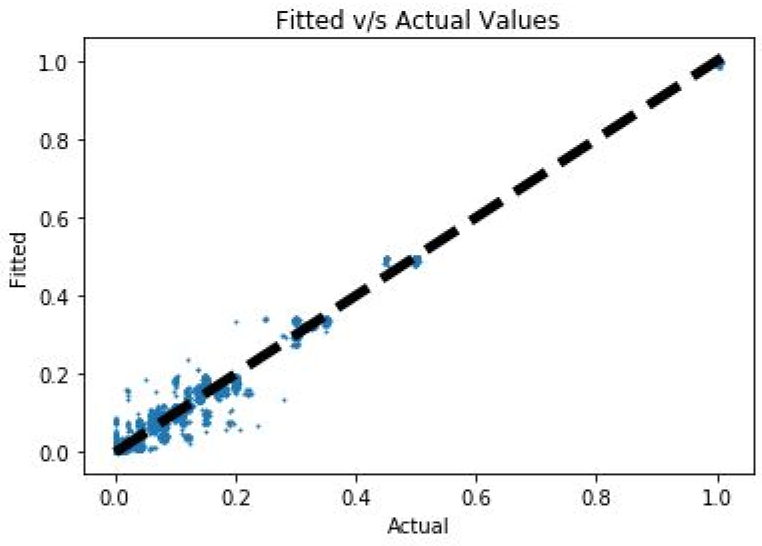
Thus, from all the observations and analysis we made in section (i) , (ii) and (iii), we conclude that the best predictive power of the model is observed for the following parameters:

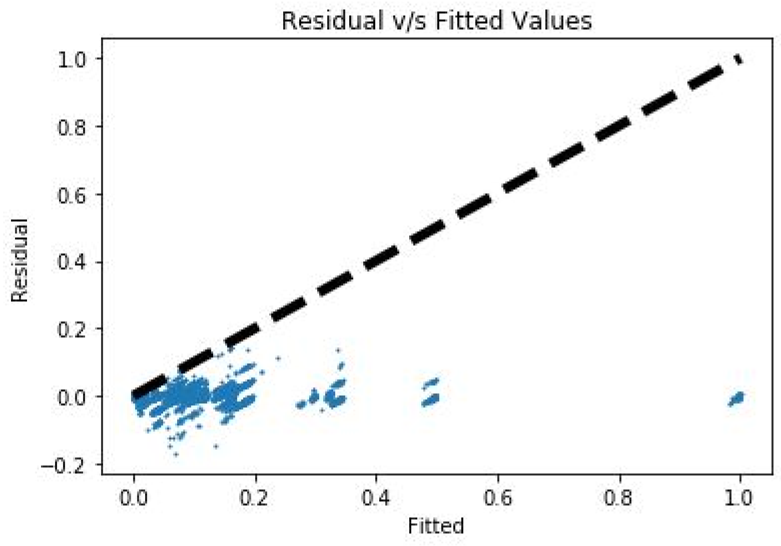
- **n_estimators=200 (Trees in our forest)**
- **max_depth=10 (depth of each tree)**
- **max_features=4 (the maximum number of features which the model uses for prediction)**

The following scatter plots further strengthen our claim that this is the best regression model




Residual & Fitted values vs Data Points
Residual & Fitted Values
Datapoints
Fitted
Residual
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2500
5000
7500
10000
12500
15000
17500


Fitted v/s Actual Values
Fitted
Actual
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



Analysis:

- We observe that there is an almost perfect overlap between the actual and fitted values
- The residual values are also all near and around 0. Most of them are concentrated around 0+=0.1. Which means there is very less difference between the actual and fitted values.

These plots verify that our model is far better than what we plotted earlier. So, we can easily claim that this is the best predictive model.

iv) We now find the feature importances of the best random forest model we received above.

For max_features=4, n_estimators=200 and max_depth=10, we get the following feature importances
[0.00355358, 0.3469408 , 0.33692236, 0.25030568, 0.06227758]

This essentially means the following:

Feature	**Feature Importance**
Week #	0.00355358
Day of Week	0.3469408
Backup Start Time - Hour of Day	0.33692236
Work-Flow-ID	0.25030568
File Name	0.06227758

Analysis:
Of all the five features, we observe that the feature 'Day of Week' gets the most importance in our best random forest regression model. This also supplements our intuition which says that the traffic data on

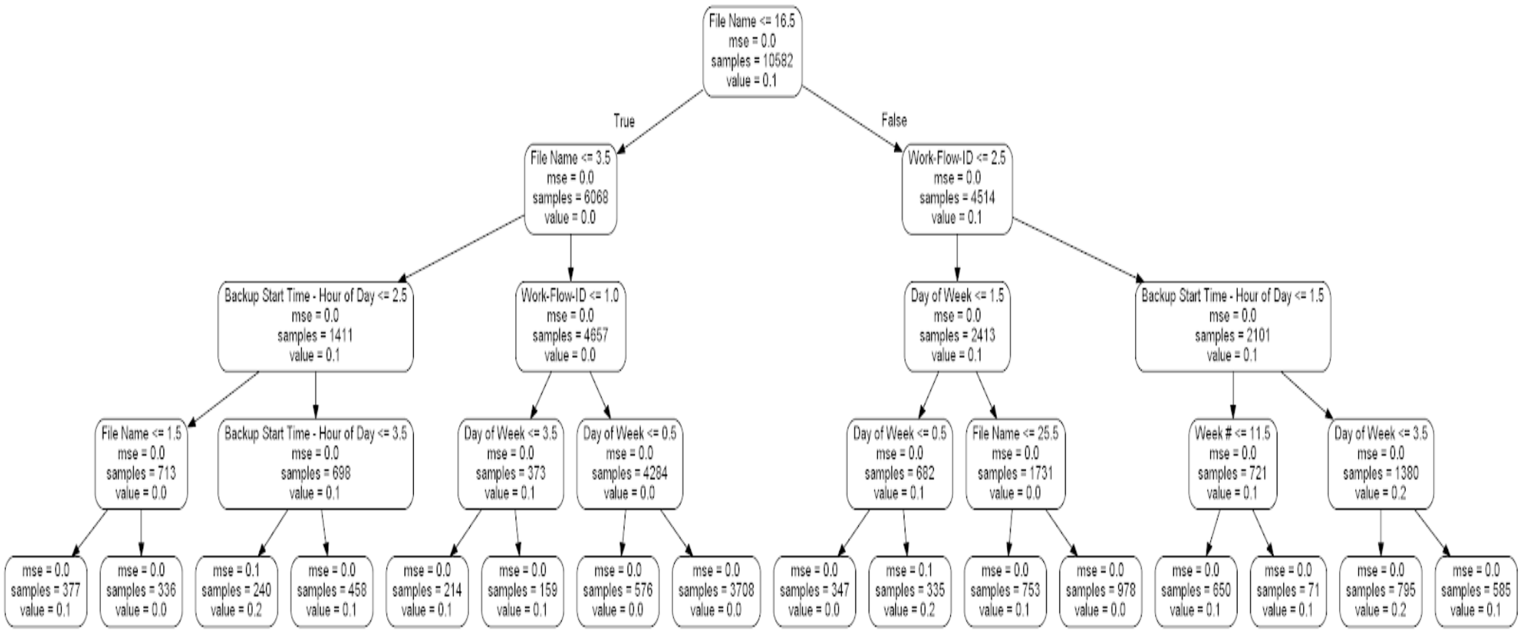
the backup system over the network might be very high on some days of the week while low on others. Thus, the day of week can play an important role in the prediction of backup size.
Here, we also see that the importance of one feature (Week #) is very low. This means that since we have told the model to use a maximum of four features for regression, the other four features would give better results than this one.
For our best model, since the depth is 10, we are not reporting the visualized tree here since the image is very huge and not visible to the naked eye. However, we have visualized with depth=4 (which however does not give the best results)

v) Visualization
In this section, we visualize a tree with depth=4. Since we know that a forest with 200 trees gives better results, we have visualized a tree with depth=4 in this forest.
The feature importances of this forest with 200 trees, max_depth=4 and max_features=4 are:
1.19191814e-04, 3.54656316e-01, 9.17301062e-02, 4.55648246e-01, 9.78461403e-02

This essentially means the following:

Feature	**Feature Importance**
Week #	0.00011919181
Day of Week	0.35465631
Backup Start Time - Hour of Day	0.0917301062
Work-Flow-ID	0.455648246
File Name	0.0978461403

Visualization:



Analysis:

Now that we changed the max depth from 10 to 4, we observe that our best feature for this model is the 'Work-Flow-ID'. This is because as observed before, a max_depth of 10 gives better prediction results than that of 4. Hence we do not get 'Day of Week' as the most important feature here as we get in our best prediction model (with max_depth=10). However, we observe that the 'Week #' still has the least importance out of all the features. The same was observed for our best predictive model.

However, when we visualize one tree from this forest of 200 trees, we get the root node as 'File Name' and not 'Work-Flow-ID'. This is mostly because here, we are visualizing only one tree from our forest while the feature importances retrieved from the model is that of the entire forest and not for one tree in particular.

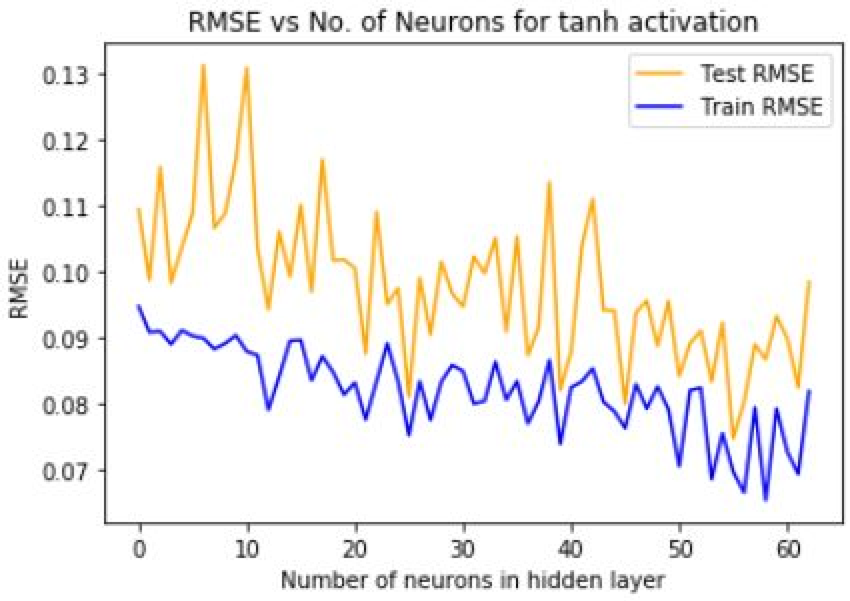
When we add up the feature importances, we get 1. Also, the visualization shows a tree of depth 4. Both these results verify the correctness of the model.

c) Neural Network

After One-hot encoding all the features, each input is of size 63. This means that our input layer will have 63 neurons in it. For getting the best value for number of neurons in hidden layer, we varied the ***hidden_layer_sizes*** from 1 to 63 and calculated the RMSE . We chose the best value to be the one with minimum test RMSE. Following are the outputs for different activation functions: Training and test RMSE is for the best value of neurons for each activation function.

Test and Training RMSE for different activation function			
	TANH	RELU	LOGISTIC
Training RMSE	0.0696	0.0227	0.0884
Test RMSE	0.0747	0.0335	0.0889
No. of neurons	56	37	42

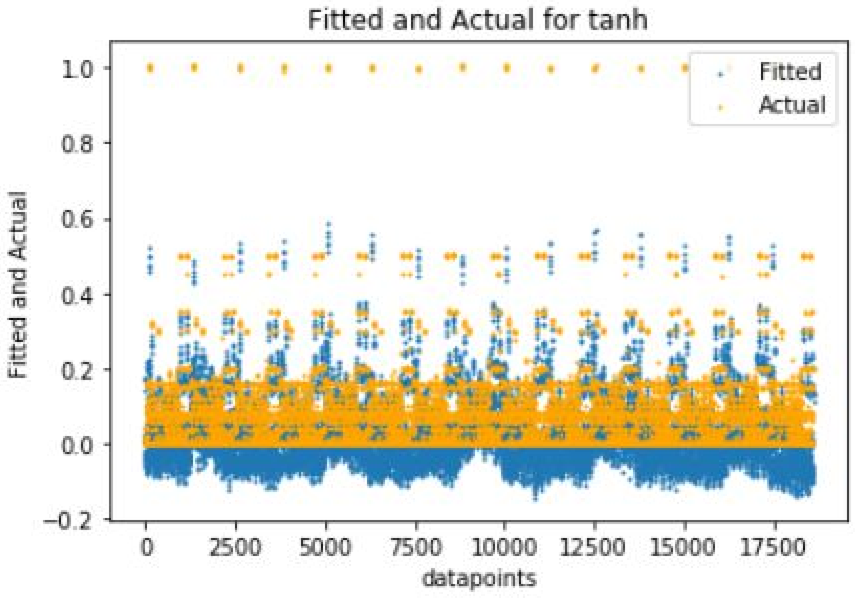
Activation : tanh
f(x) = tanh(x)

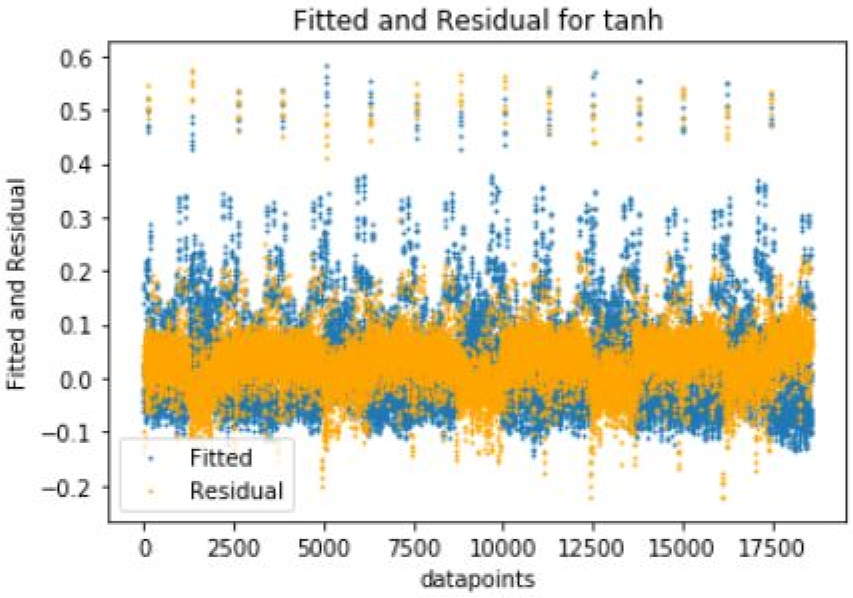


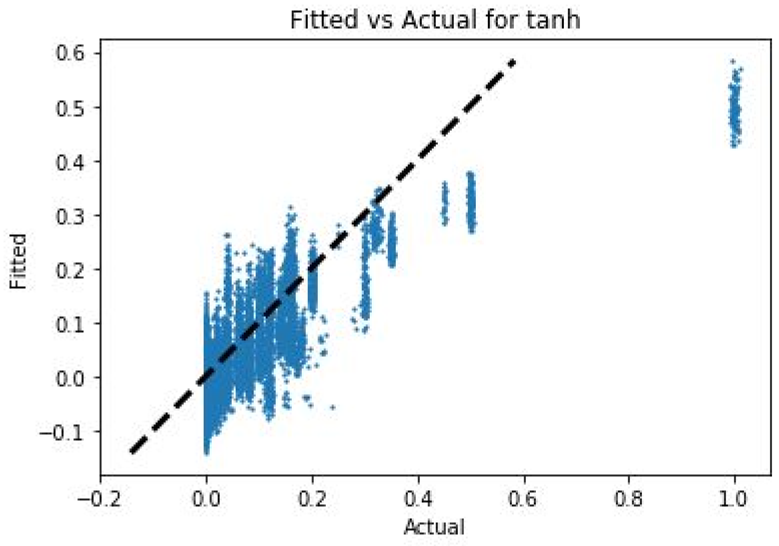
We can observe that as the number of neurons in the hidden layers increases, the error decreases. But the error does not decrease smoothly as the activation function is tan.

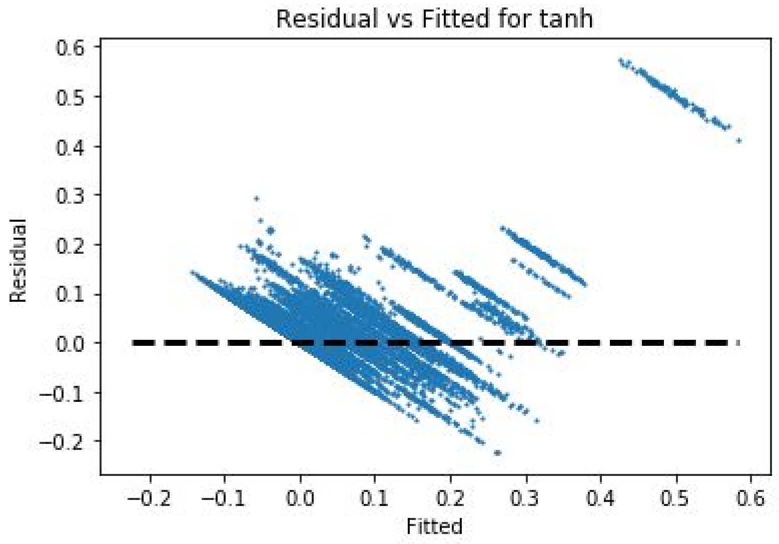
The best value for number of neurons 56.

Following are the graphs of predicted,fitted and residual:


Fitted and Actual for tanh
Fitted
Actual
Fitted and Actual
datapoints
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2500
5000
7500
10000
12500
15000
17500

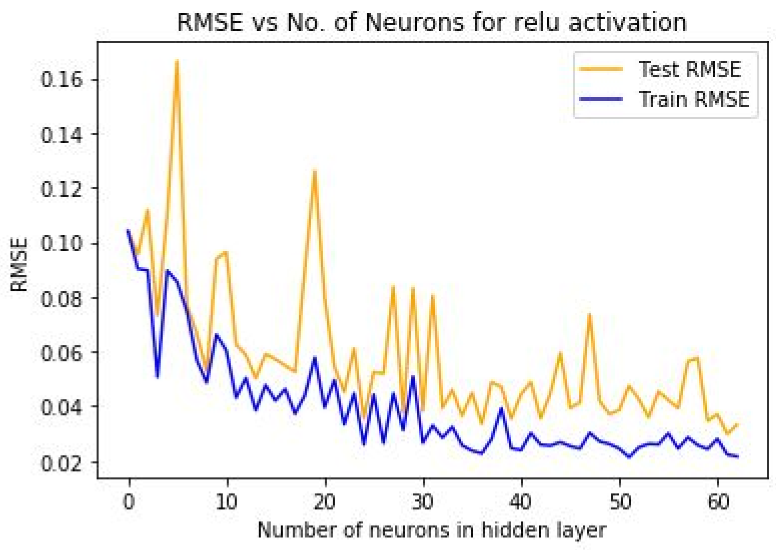

Fitted and Residual for tanh
Fitted
Residual
Fitted and Residual
datapoints
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
0
2500
5000
7500
10000
12500
15000
17500





From the above scatter plots, we can observe that, actual and fitted values are almost close to each other , which means the error is low. From the residual plot, we can see that the residual values are clustered at 0 which implies that the error is low.

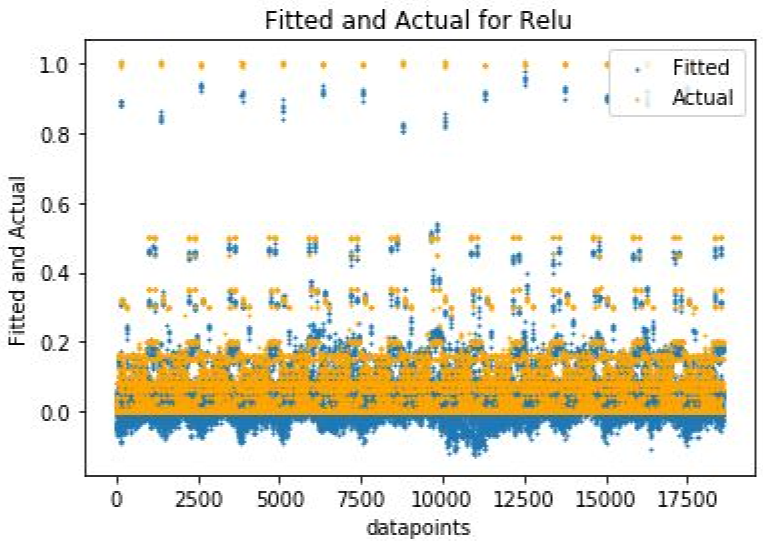
Activation : Relu
f(x) = max(0, x)

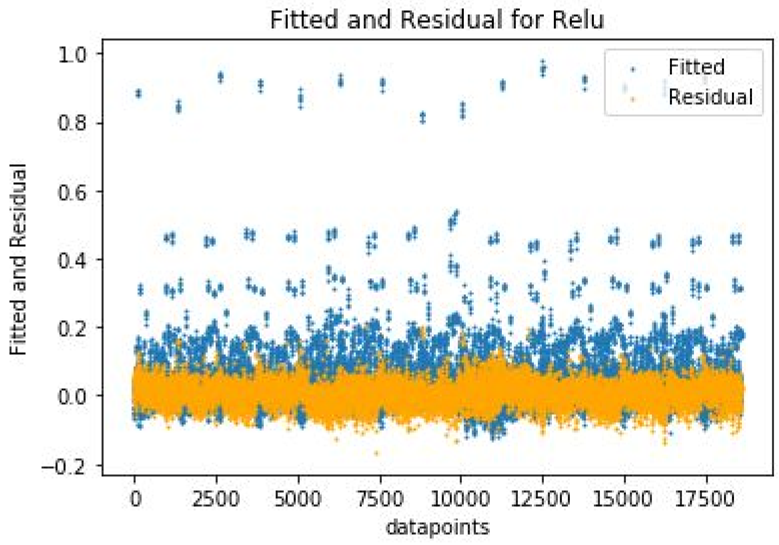


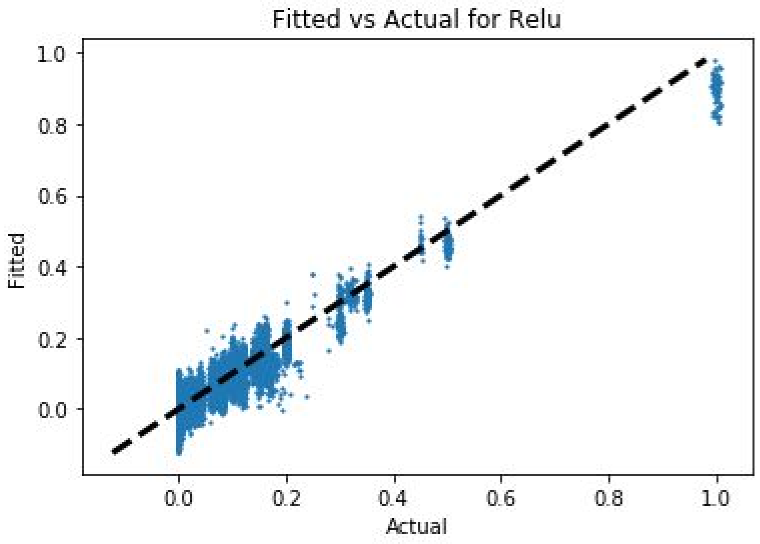
We can observe that as the number of neurons in the hidden layers increases, the error decreases. Relu performs better compared to other activation functions for our dataset.

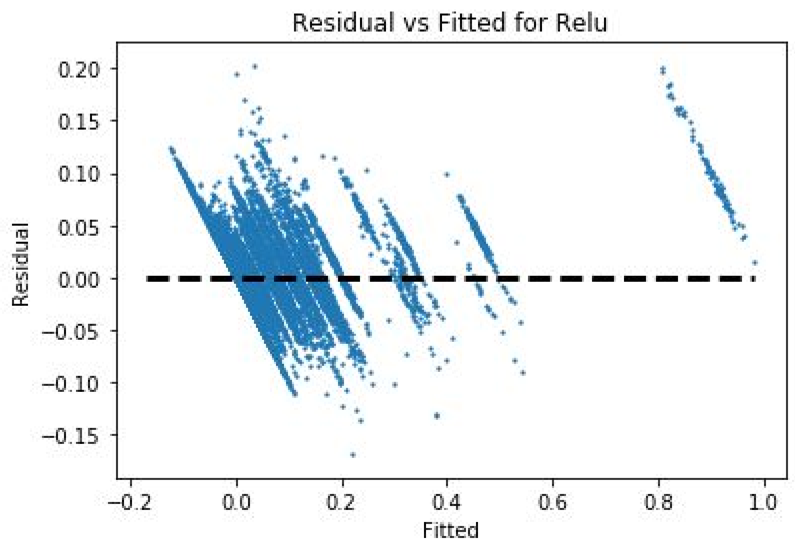
The best value for number of neurons is 37

Following are the graphs of predicted,fitted and residual:



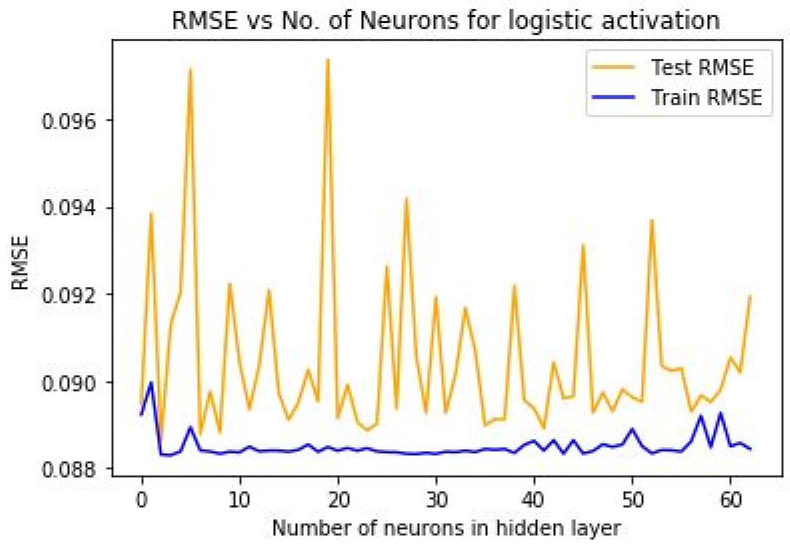

Fitted and Residual for Relu
Fitted
Residual
Fitted and Residual
datapoints
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2500
5000
7500
10000
12500
15000
17500


Fitted vs Actual for Relu
Fitted
Actual
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

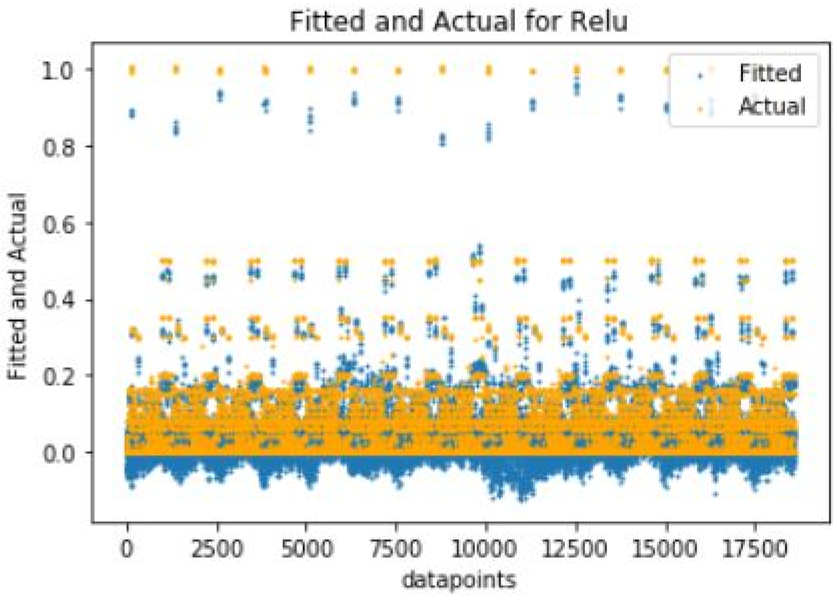


From the above scatter plots, we can observe that, actual and fitted values are almost close to each other , which means the prediction is closer to expected. From the residual plot, we can see that the residual values are clustered at 0 which implies that the error is low.

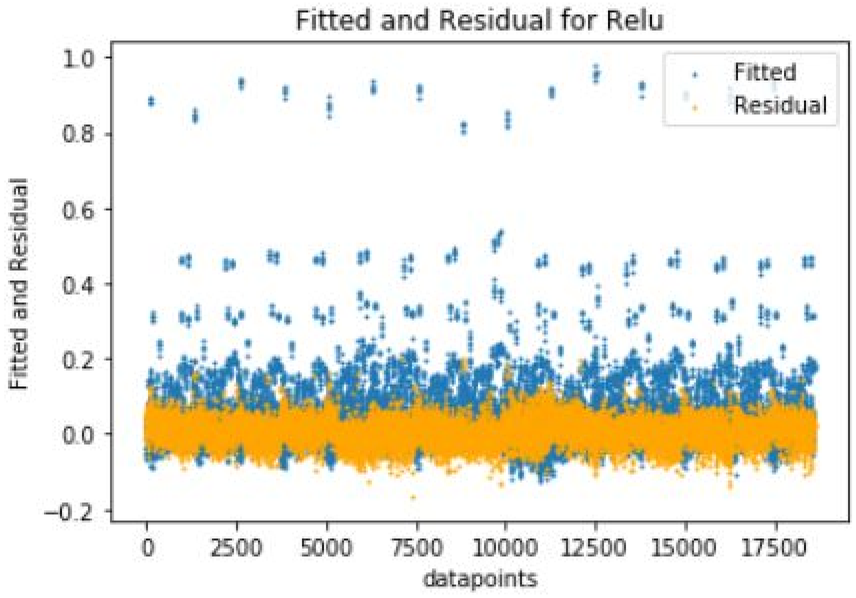
Activation : logistic
f(x) = 1 / (1 + exp(-x))



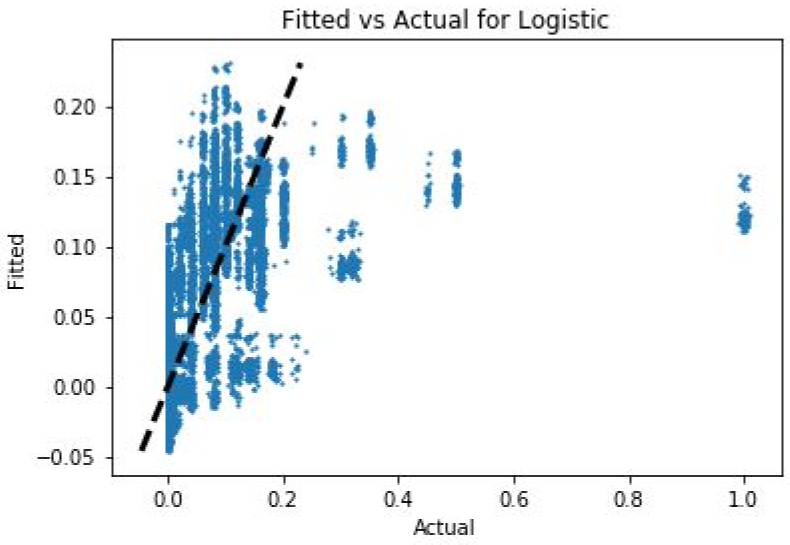
The best value for number of neurons is 42.
Following are the graphs of predicted,fitted and residual:

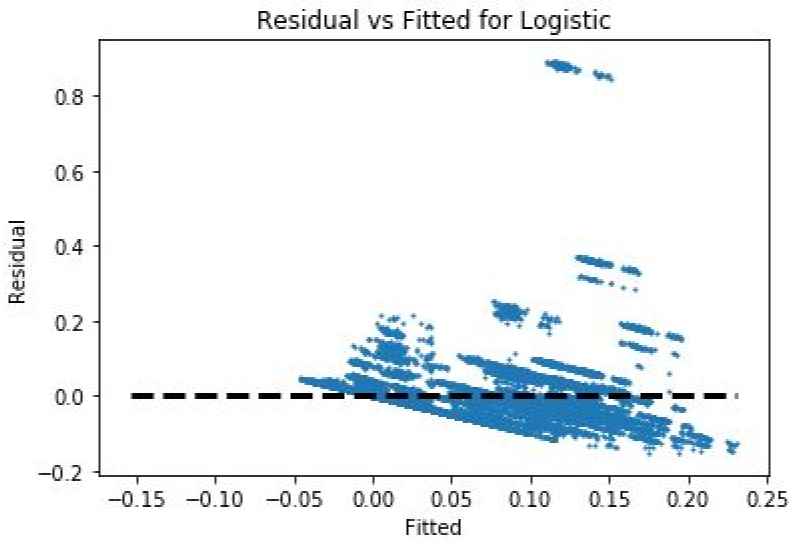


Fitted and Actual for Relu
Fitted
Actual
Fitted and Actual
datapoints
1.0
0.8
0.6
0.4
0.2
0.0
0
2500
5000
7500
10000
12500
15000
17500



Fitted and Residual for Relu
Fitted
Residual
Fitted and Residual
datapoints
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2500
5000
7500
10000
12500
15000
17500





From the above scatter plots, we can observe that, actual and fitted values are almost close to each other , which means the prediction is closer to expected. From the residual plot, we can see that the residual values are clustered at 0 which implies that the error is low.

Observations:

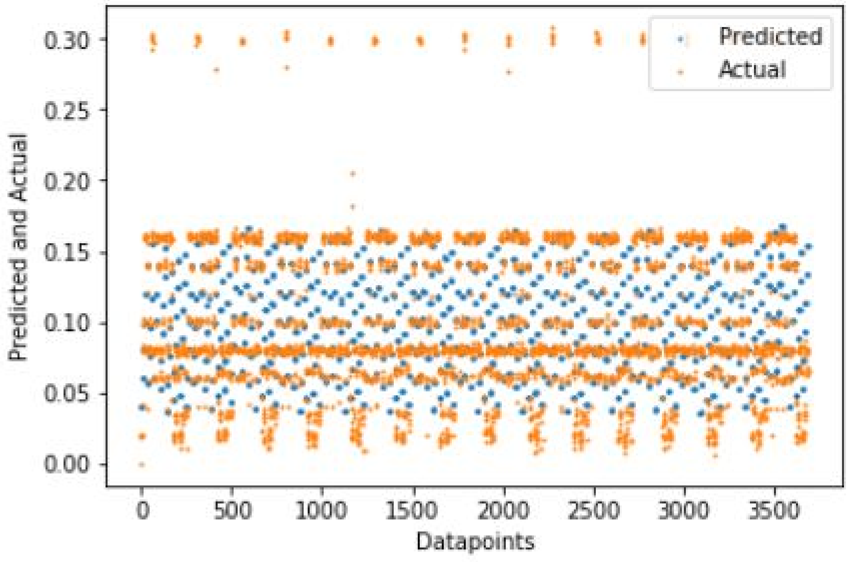
1. Relu performs better than other activation functions from our analysis. We can see that for relu, predicted and actual values have more overlap. Also, all the residual values lies really close to zero which explains the least error od relu.

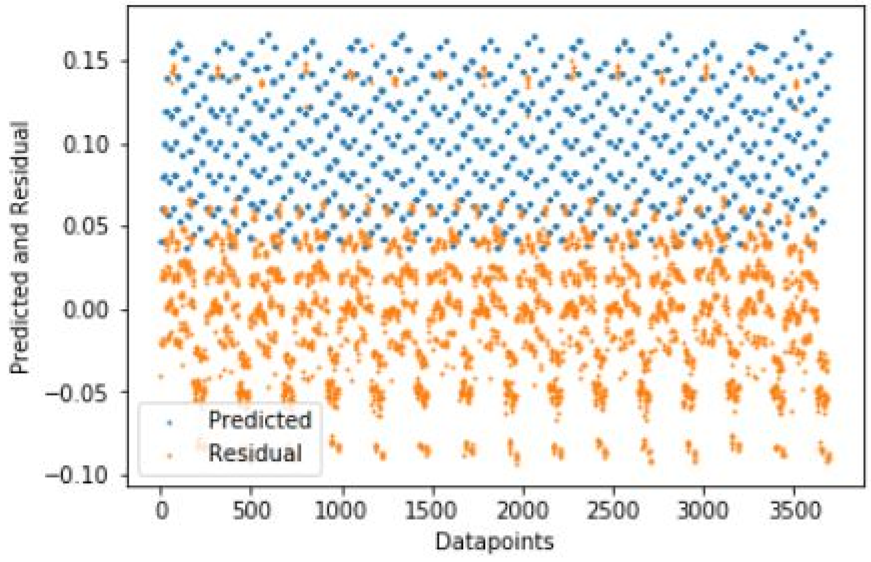
d) Polynomial Regression

(i) Predict the backup size for each of the workflows separately using linear regression model.

In this section, we perform piece-wise linear regression for each of the workflows present in the network dataset. We do this by grouping the data based on workflow ID column and then run 10-fold cross validation on it to test for overfitting.

- **Workflow 0**
  Train RMSE: **0.04297**
  Test RMSE: **0.04328**

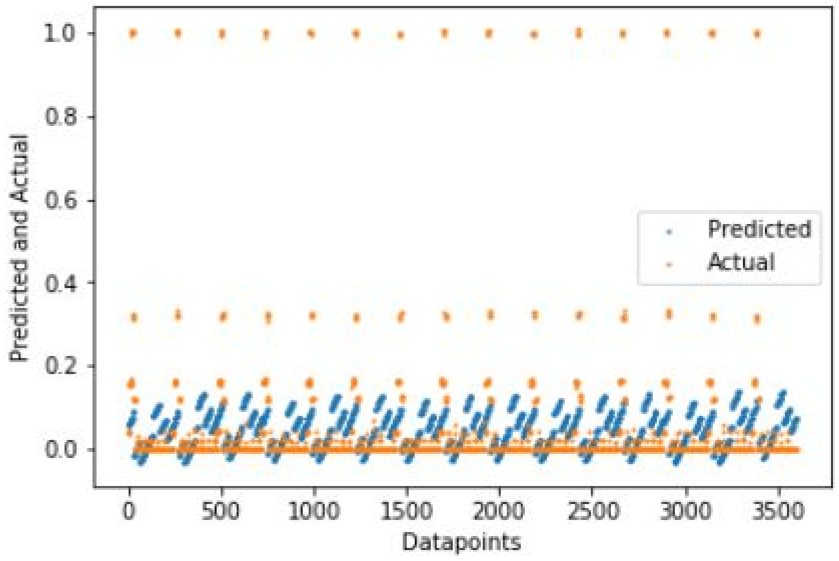


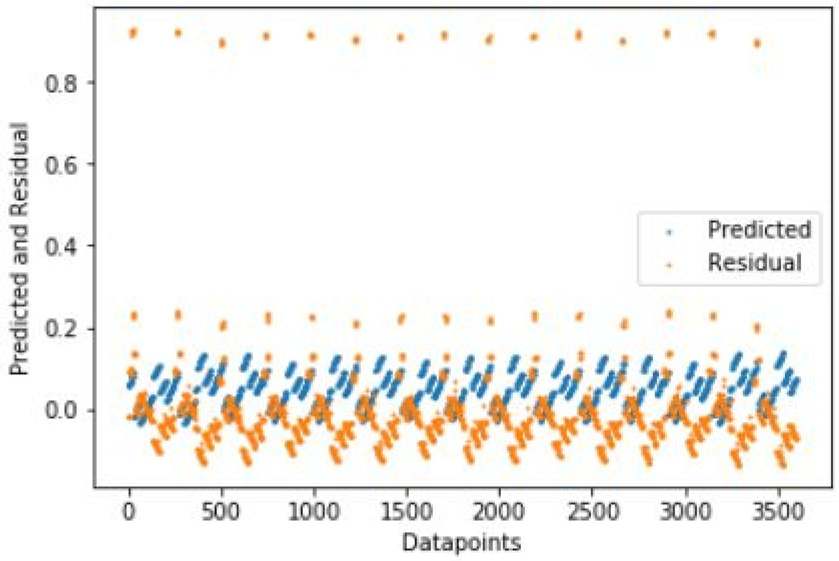


Observations:

The RMSE value for workflow-0 is **much lower** compared to the overall RMSE value of **0.102** for the entire dataset. This indicates there is a huge improvement for workflow-0 if we perform piece-wise linear regression.
From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is also low in the range on **0.10**. This suggests that the individual fit works better than the overall fit.

- **Workflow 1**
  Train RMSE: **0.1596**
  Test RMSE: **0.1609**

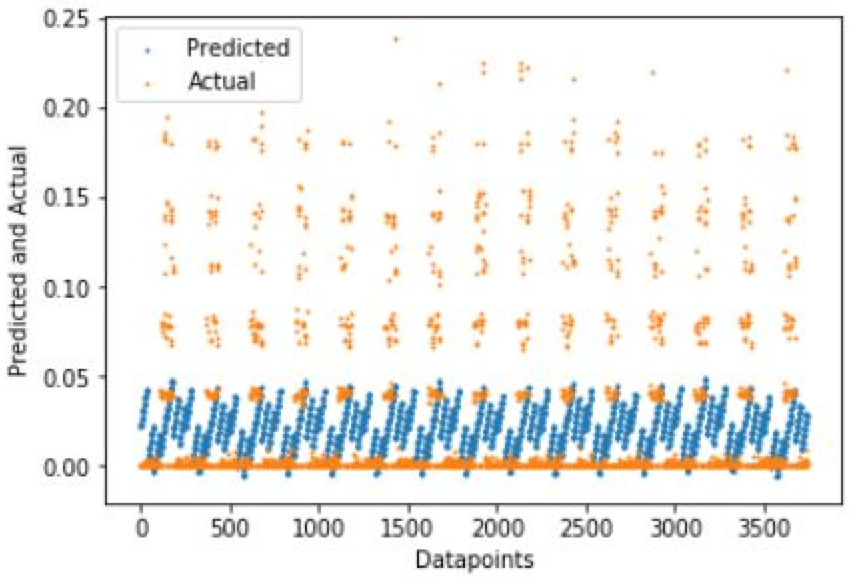


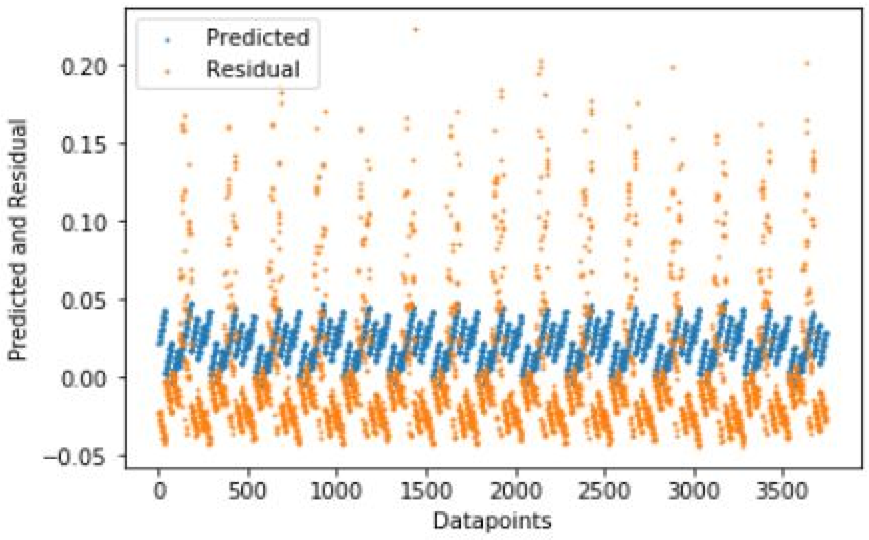


Observations:

The RMSE value for workflow-1 is **much higher** compared to the overall RMSE value of **0.102** for the entire dataset. This indicates that the performance degrades for workflow-1 if we perform piece-wise linear regression.
From the plots it can be seen that the some of the predicted values are far from the actual value. The value of **residual** also varies a lot in the range of more than **0.8**. This suggests that the individual fit does not work better than the overall fit in this case.

- **Workflow 2**
  Train RMSE: **0.04224**
  Test RMSE: **0.04329**

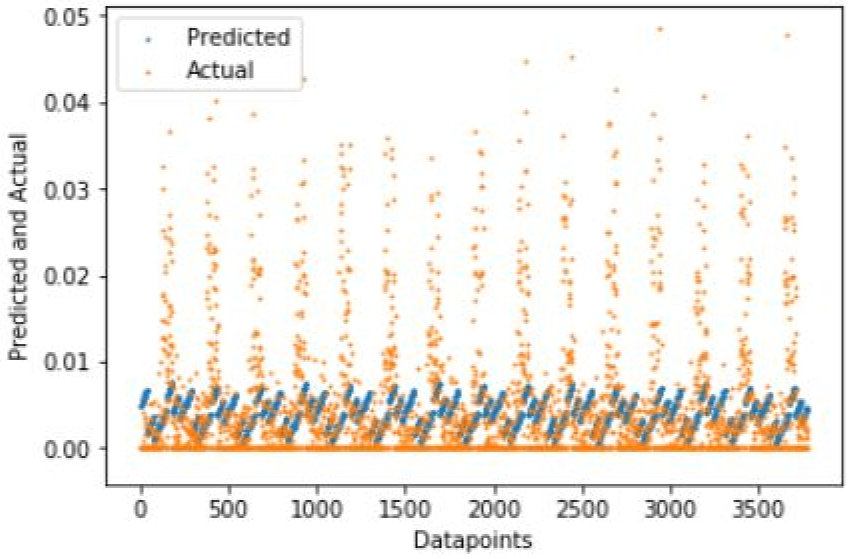


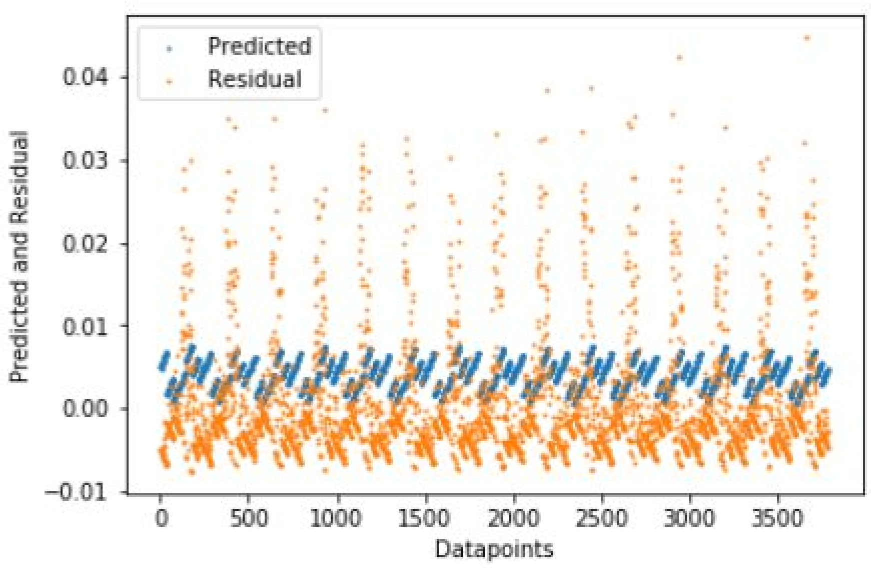


Observations:

The RMSE value for workflow-2 is **much lower** compared to the overall RMSE value of **0.102** for the entire dataset. This indicates there is a huge improvement for workflow-2 if we perform piece-wise linear regression just like in the case of workflow-0.
From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is low in the range on **0.15** except for some outliers. This suggests that the individual fit works better than the overall fit.

- **Workflow 3**
  Train RMSE: **0.00712**
  Test RMSE: **0.00713**

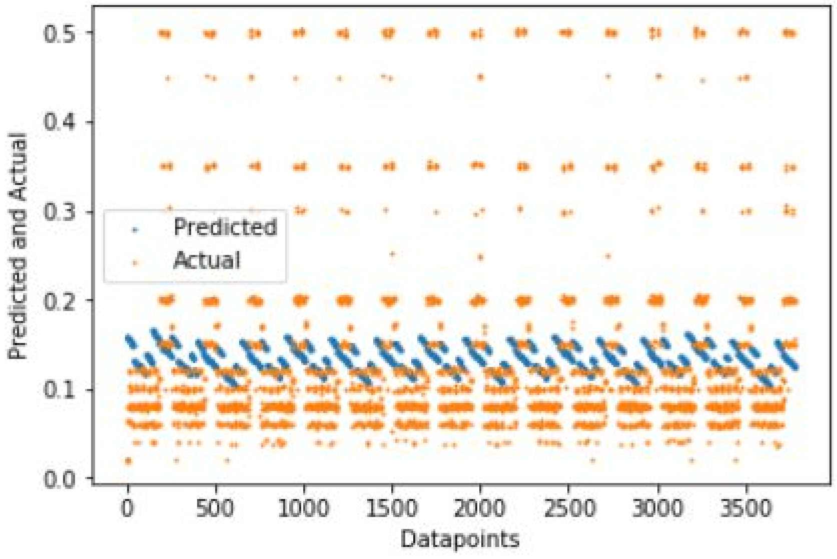


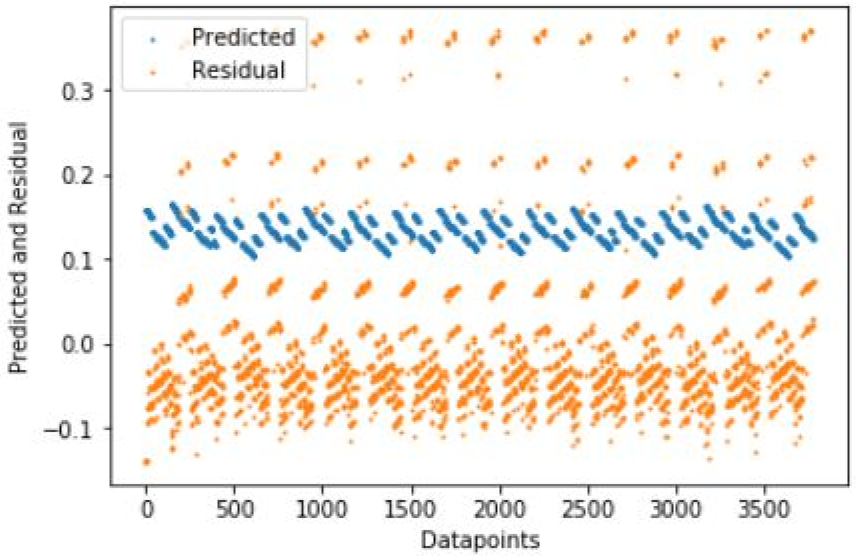


Observations:

The RMSE value for workflow-3 is **much lower** compared to the overall RMSE value of **0.102** for the entire dataset and the value is lowest among all workflows. This indicates the best improvement is observed for workflow-3 if we perform piecewise linear regression.
From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is also low in the range on **0.05**. This suggests that the individual fit works way better than the overall fit.

- **Workflow 4**
  Train RMSE: **0.1029**
  Test RMSE: **0.1036**





Observations:

The RMSE value for workflow-4 is **almost equal** compared to the overall RMSE value of **0.102** for the entire dataset. This indicates there is not really any improvement for workflow-4 if we perform piece-wise linear regression.
The predicted values and the residual values indicated by the plots suggest that the individual fit works almost equally as the overall fit and there is not reasonable improvement.

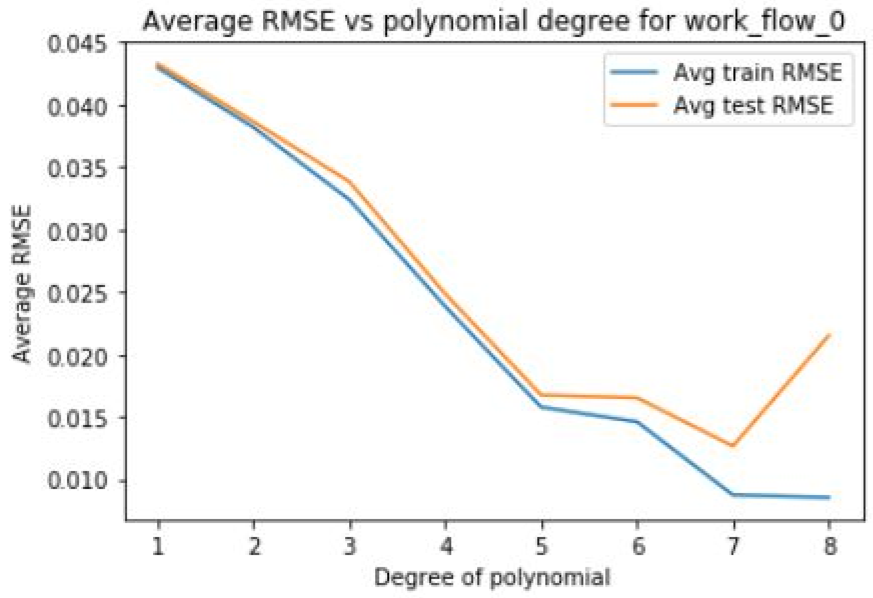
Q: Explain if the fit is improved?

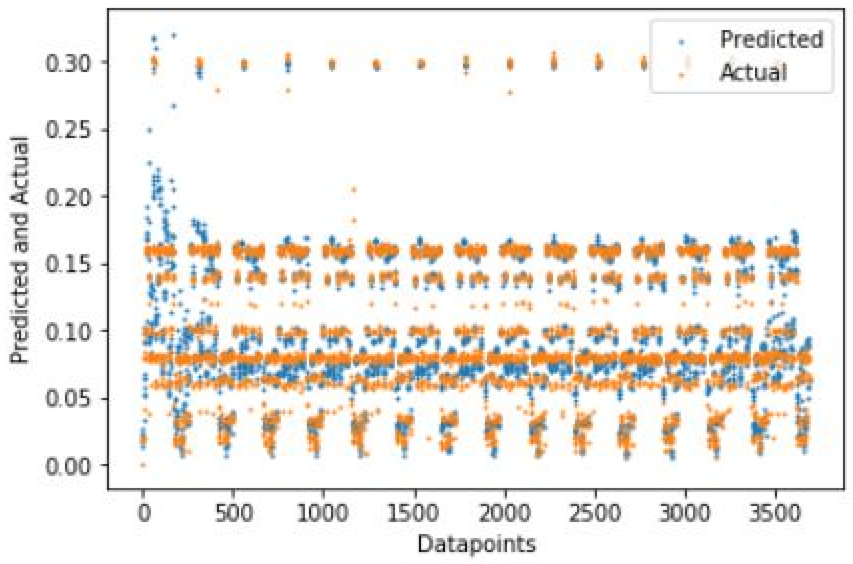
Answer: From the values of RMSE for different workflows, it can be said that the fit has definitely improved for piecewise linear regression compared to the general model. We did note that the performance for workflow-1 is worse and for workflow-4, the performance is similar to general linear regression. However, if we consider the values together for all workflows, the improvement can clearly be seen.

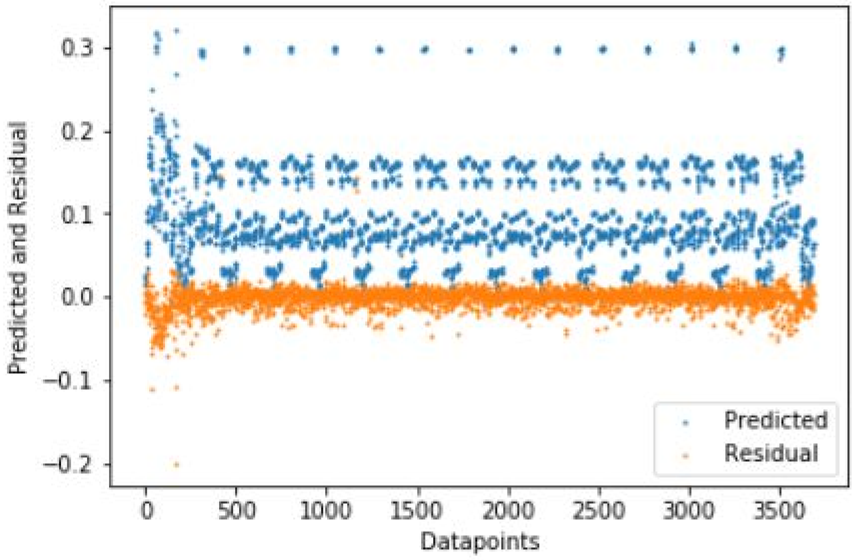
(ii) Try fitting a more complex regression function to your data. You can try a polynomial function of your variables. Try increasing the degree of the polynomial to improve your fit. Again, use a 10 fold cross validation to evaluate your results.

In this section, we use a more complex polynomial function to improve the fit of the variables and in turn, improve the predicted Backup size. The model was evaluated using 10-fold cross validation by varying the degree of polynomial between 1 and 10. When we plot average train and test RMSE of the trained model for each workflow, we get a different optimal value for polynomial degree. The values of optimal degree range from **5 to 8**.

- **Workflow 0**
  Best degree: 7
  Best Train RMSE: **0.00874**
  Best Test RMSE: **0.01265**


Average RMSE vs polynomial degree for work_flow_0
Avg train RMSE
Avg test RMSE
Average RMSE
Degree of polynomial
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
1
2
3
4
5
6
7
8


Predicted
Actual
Predicted and Actual
Datapoints
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
500
1000
1500
2000
2500
3000
3500



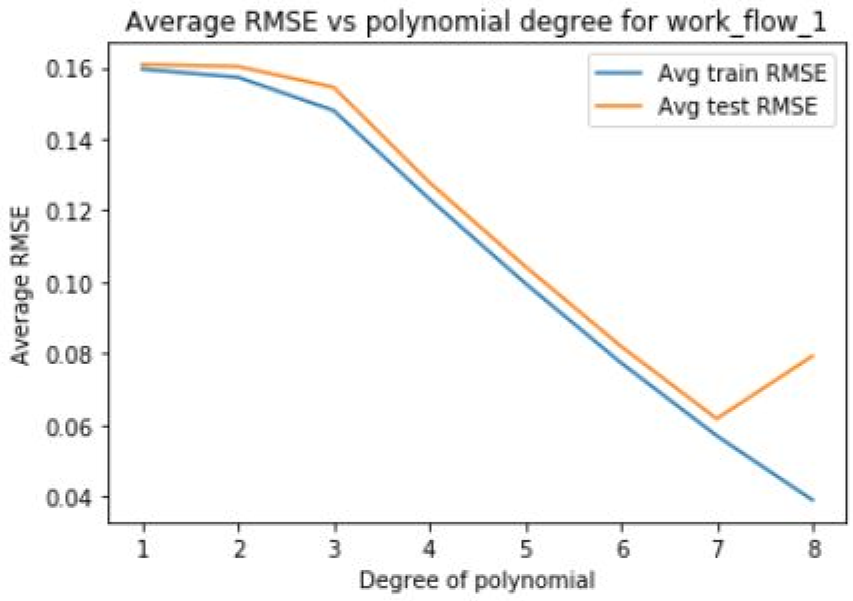
Observations:

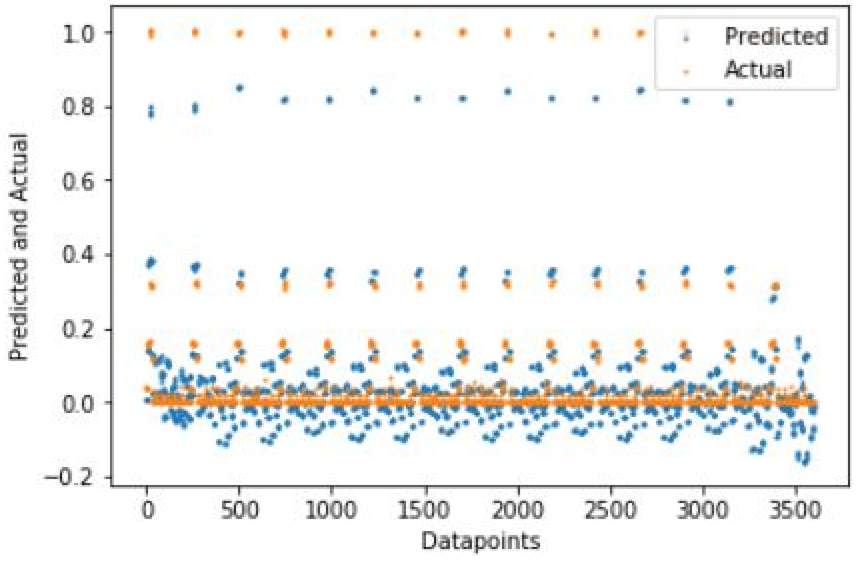
It can be seen from the first plot that the RMSE value for workflow-0 decreases as we increase the degree till the value of 7, after which the RMSE starts to increase.

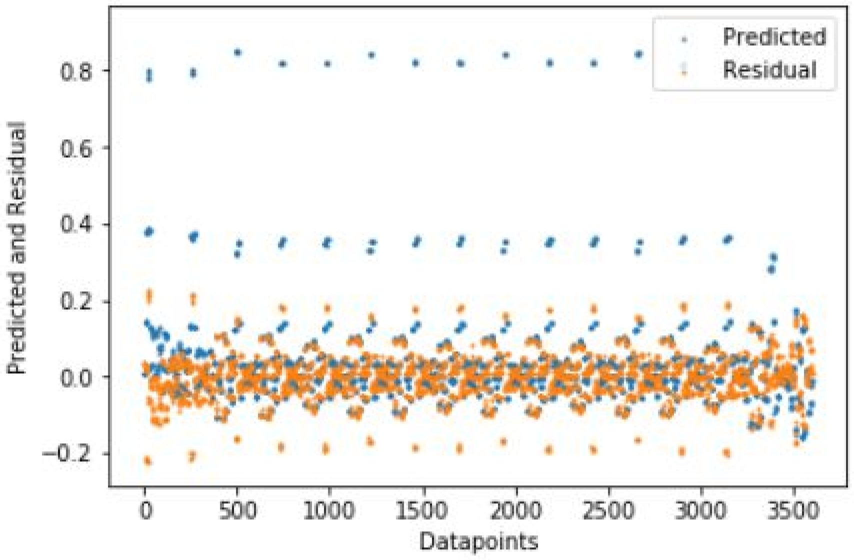
The minimum RMSE value for workflow-0 is **much lower** compared to the corresponding RMSE value obtained for piecewise regression. This indicates there is a huge improvement for workflow-0 if we perform polynomial regression.

From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is very low in the range on **0.10**. This suggests that the polynomial fit works better than the linear one.

- **Workflow 1**
  Best degree: 7
  Best Train RMSE: **0.0569**
  Best Test RMSE: **0.0617**







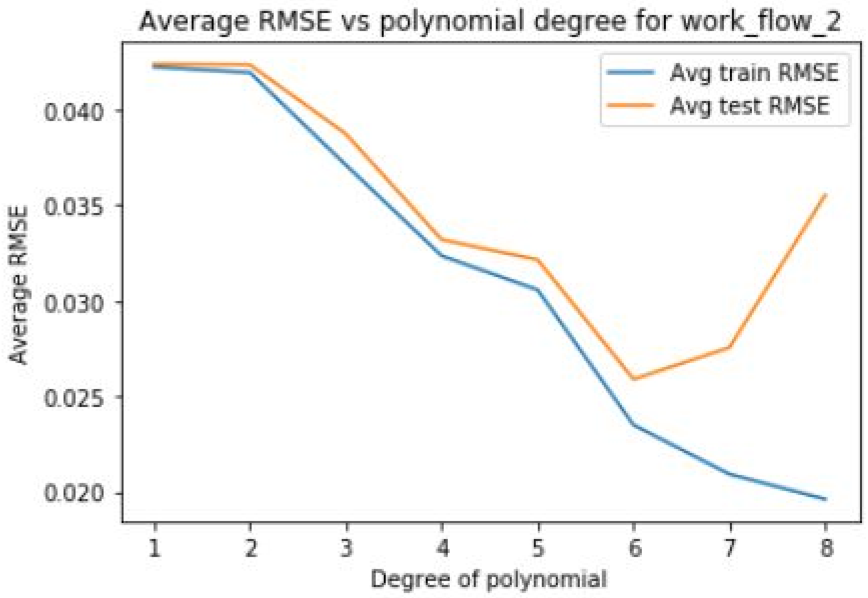
Observations:

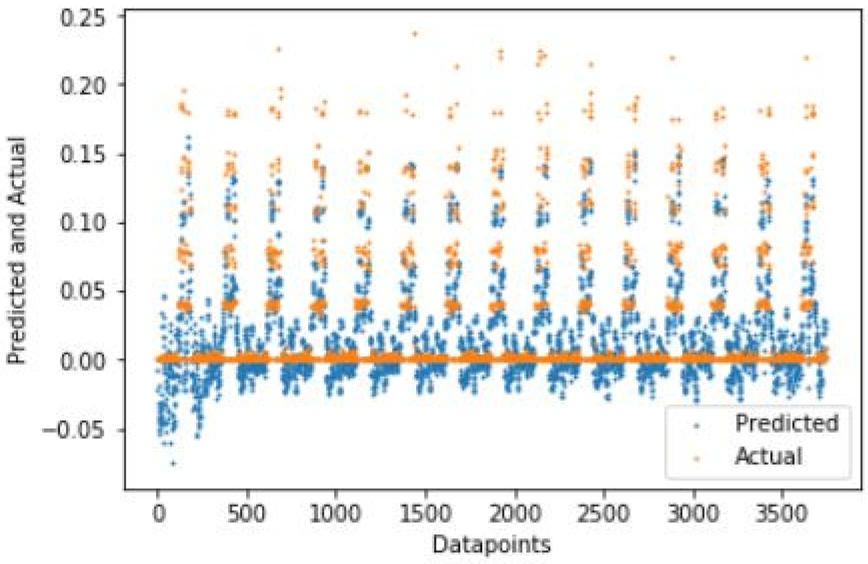
It can be seen from the first plot that the RMSE value for workflow-1 decreases as we increase the degree till the value of 7, after which the RMSE starts to increase.

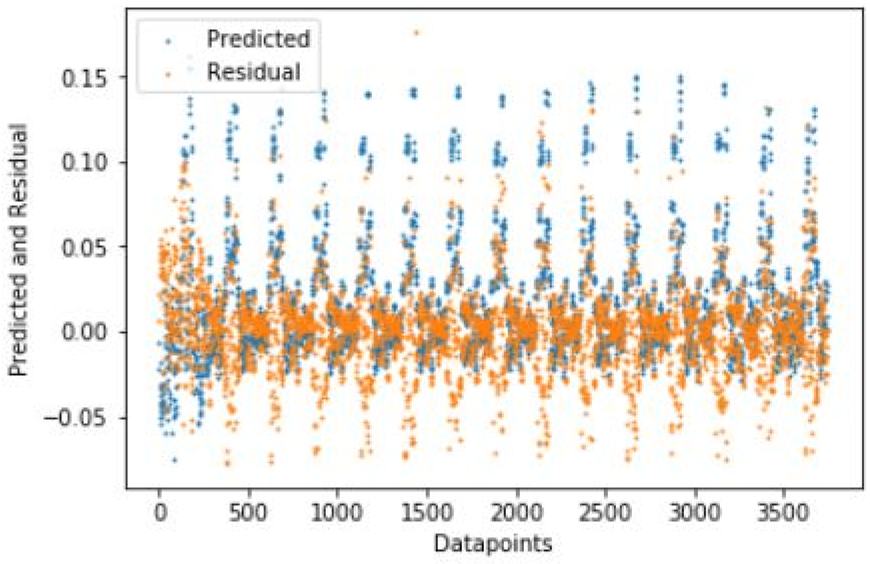
The minimum RMSE value for workflow-1 is **much lower** compared to the corresponding RMSE value obtained for piece-wise regression. This indicates there is a huge improvement for workflow-1 if we perform polynomial regression.

From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is very low in the range on **0.20**. This suggests that the polynomial fit works better than the linear one.

- **Workflow 2**
  Best degree: 6
  Best Train RMSE: **0.02349**
  Best Test RMSE: **0.02588**


Average RMSE vs polynomial degree for work_flow_2
Avg train RMSE
Avg test RMSE
Average RMSE
0.040
0.035
0.030
0.025
0.020
1
2
3
4
5
6
7
8
Degree of polynomial


0.25
0.20
0.15
0.10
0.05
0.00
−0.05
Predicted and Actual
Predicted
Actual
0
500
1000
1500
2000
2500
3000
3500
Datapoints



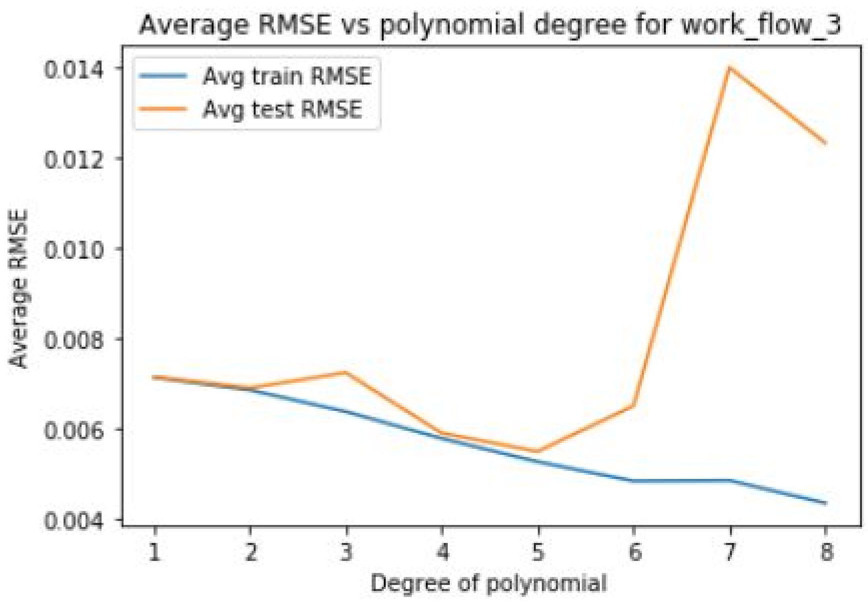
Observations:

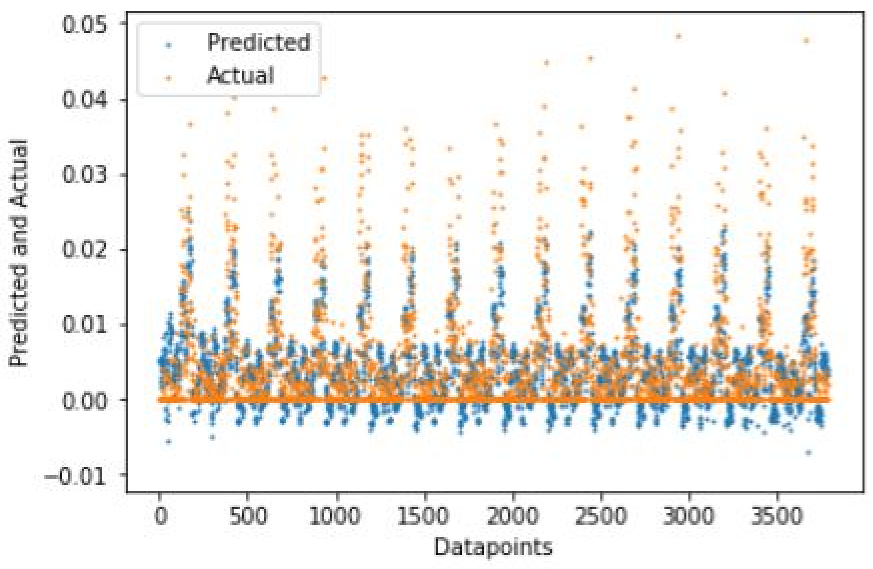
It can be seen from the first plot that the RMSE value for workflow-2 decreases as we increase the degree till the value of 6, after which the RMSE starts to increase.

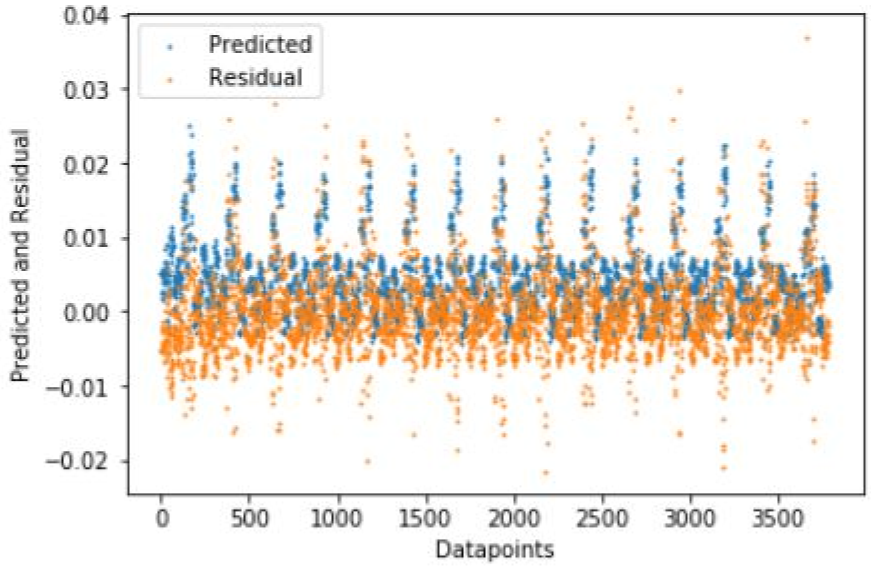
The minimum RMSE value for workflow-2 is **lower** compared to the corresponding RMSE value obtained for piece-wise regression. This indicates there is an improvement for workflow-2 if we perform polynomial regression.

From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations and outliers. The value of **residual** is low in the range on **0.15**. This suggests that the polynomial fit works better than the linear one.

- **Workflow 3**
  Best degree: 5
  Best Train RMSE: **0.00524**
  Best Test RMSE: **0.00547**


Average RMSE vs polynomial degree for work_flow_3
Avg train RMSE
Avg test RMSE
Average RMSE
Degree of polynomial


Predicted
Actual
Predicted and Actual
Datapoints



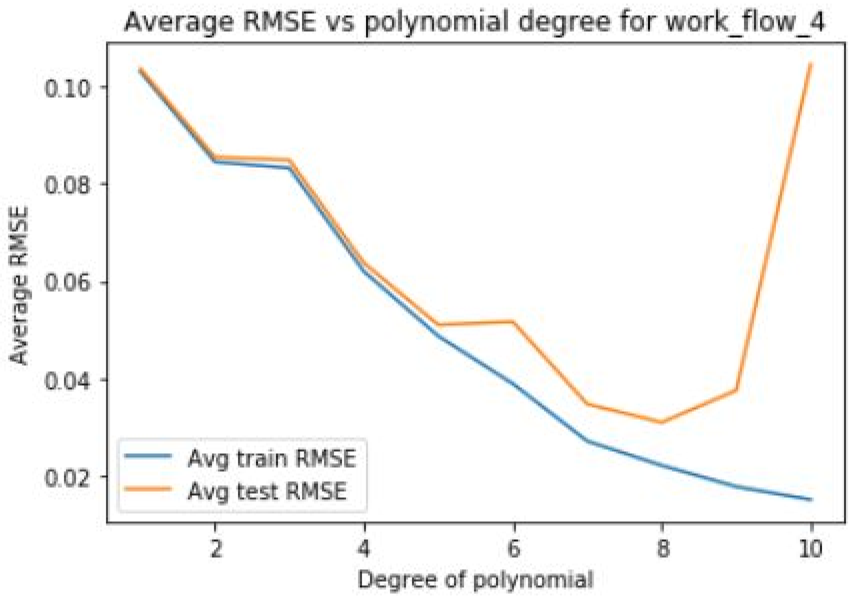
Observations:

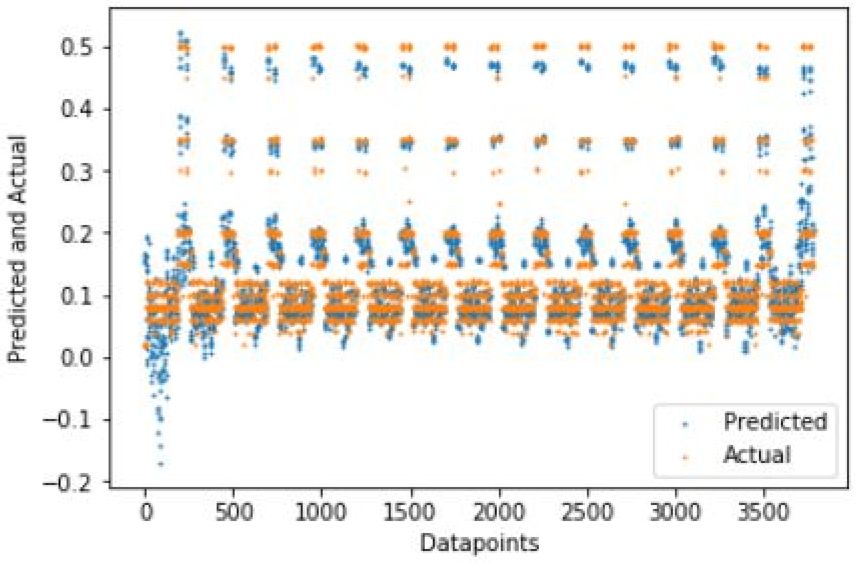
It can be seen from the first plot that the RMSE value for workflow-3 decreases as we increase the degree till the value of 5, after which the RMSE starts to increase.

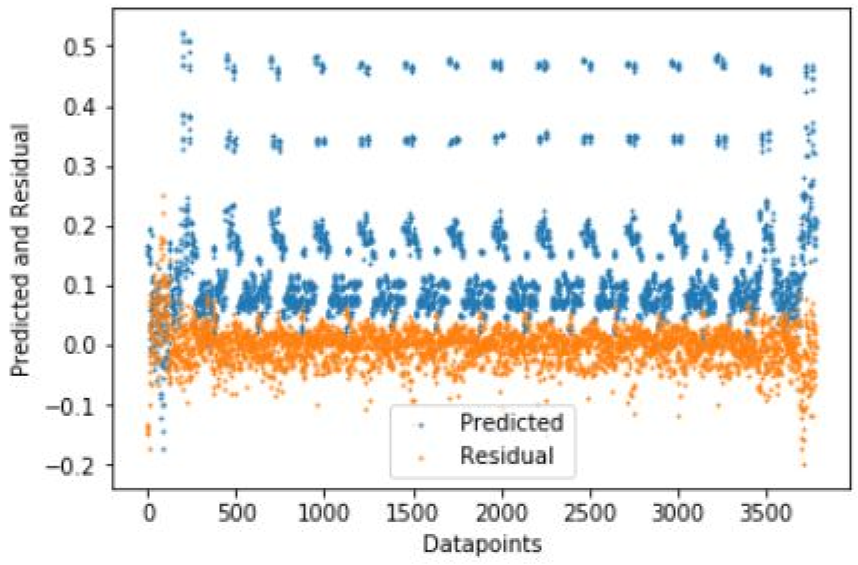
The minimum RMSE value for workflow-3 is **lower** compared to the corresponding RMSE value obtained for piece-wise regression. This indicates there is an improvement for workflow-3 if we perform polynomial regression.

From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is very low in the range on **0.05**. This suggests that the polynomial fit works better than the linear one.

- **Workflow 4**
  Best degree: 8
  Best Train RMSE: **0.0221**
  Best Test RMSE: **0.0309**


Average RMSE vs polynomial degree for work_flow_4
Average RMSE
Degree of polynomial
Avg train RMSE
Avg test RMSE
0.10
0.08
0.06
0.04
0.02
2
4
6
8
10


Predicted and Actual
Datapoints
Predicted
Actual
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
0
500
1000
1500
2000
2500
3000
3500



Observations:

It can be seen from the first plot that the RMSE value for workflow-4 decreases as we increase the degree till the value of 8, after which the RMSE starts to increase.

The minimum RMSE value for workflow-4 is **much lower** compared to the corresponding RMSE value obtained for piecewise regression. This indicates there is a huge improvement for workflow-4 if we perform polynomial regression.

From the plots it can be seen that the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is very low in the range on **0.10**. This suggests that the polynomial fit works better than the linear one.

- **Thus, it can be concluded that if we increase the degree of fitted polynomial beyond the threshold of 6, then the generalization error of our model starts to get worse.**

Q: Can you explain how cross validation helps controlling the complexity of your model?

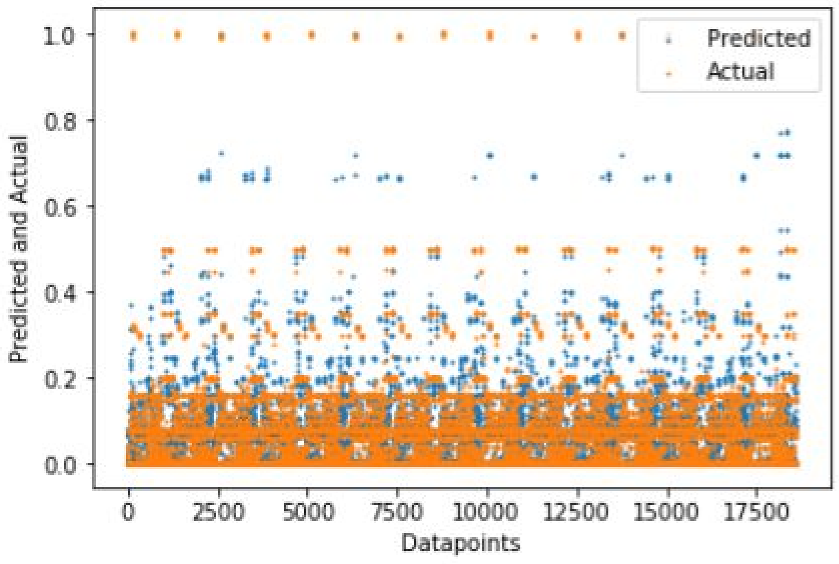
Answer: Cross Validation is a useful measure to evaluate the performance of any predictive model. By using high degree polynomial to fit the features, we might end up overfitting the data. In such scenarios, cross-validation can help in achieving the best fit on data without overfitting the model. A small sized training set can result in underfitting while a larger set can cause overfitting. Thus, it is very important to maintain the right balance in the size of training and testing dataset.

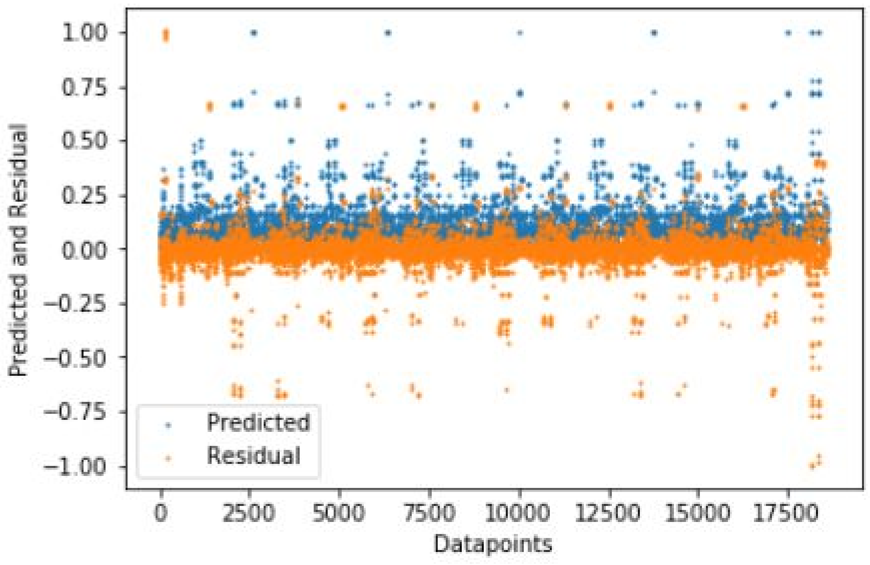
Cross Validation uses a part of training data itself to estimate the data. This makes it less prone to overfitting. Moreover, by averaging the error obtained from different folds of data, we can get a better idea about the generalization performance of the model. Thus, cross validation provides an unbiased measure of performance on unseen data. By trying cross validation on different methods of model training, one can pick the model that generates least cross validation error thereby reducing the complexity of the model.

e) KNN Regression

In this section, we use a k-nearest neighbor regression model to predict the Backup size. The model was evaluated using 10-fold cross validation by varying the number of neighbors between **1 and 20**. When we plot average train and test RMSE of the trained model for each value of k, we get 3 as the best parameter.

- Best k: **3**
- Best train RMSE: **0.0368**
- Best test RMSE: **0.0719**





Observations:

From the plots it can be seen that for k=3, the predicted value is a good approximation of the actual value with minor deviations. The value of **residual** is low in the range on **0.25** except for some datapoints.

It is worth noting that the knn regression model performs better than the linear regression model for which we got RMSE as 0.102.

Conclusion

In this project, we have observed that there are many different types of regression. Ideally, each form of regression has its own importance and a specific area of application.
Regression Analysis is beneficial because it helps us capture the following:

1. Gauge the impact of independent variables (like WorkFlow ID, Week ID,File Name etc.) on the dependent variable(Backup Size).
2. Identify relationships between the dependent and independent variables.

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?

Which model works best at handling categorical features?
We deduce from the above observations and inferences that Random **Forest Regression Model and KNN Regression Model handle categorical data better** than the Linear Regression and Multilayer Perceptron Models.

Why can Random Forest handle categorical data better?
Random Forest model and other ensemble models operate on both categorical and continuous variables. On the other hand, Linear Regression model, Multilayer Perceptron model must transform categorical variables into a numerical analog, which is usually done by one-hot encoding. However, one-hot encoding can lead to a great increase in the dimensionality of the feature representations. Additionally, it also erases important structure in the underlying representation by splitting a single feature into many separate ones. But, the main problems of using one-hot encoding in tree-based models such as Random Forest Regression is that the resulting sparsity virtually ensures that the continuous variables are assigned higher feature importance. Also, a single level of a categorical variable must meet a very high bar so as to be selected for splitting early in the decision tree building. This can degrade the predictive performance of the Random Forest Regression Model. Also, intuitively, we can comment that the less the number of available branches, the better the decision would be. Thus, we see that Random Forest Regression Model can handle categorical variables without one-hot encoding them. In fact, one-hot encoding is not necessary here.

Why can KNN handle categorical data better?
KNN algorithm can be used for both regression and classification. Since our dependent variable is continuous in nature, here we are using KNN regressor model. It should be noted that KNN may ignore a feature if it is more spread out than the others. KNN also suffers from the curse of dimensionality. Thus, as the number of feature increases like in case of one-hot encoding, the performance may get impacted. This is the reason why KNN works better with categorical data.

Now, we compare the results of Random Forest Regression Model and KNN Model to deduce which model works best with categorical data.

- Random Forest Regressor: For 200 trees with max_depth=10 and max_features=4, we get best Test-RMSE as 0.01307
- KNN Regressor: For k=3, we get best Test-RMSE as 0.0719

From these observations, we can conclude that Random Forest Regression Model is the best model to handle categorical data.

Which model works well with sparse data?
For our project, **MLP works well with sparse data** (i.e., when the features are all one-hot encoded.) Generally, simple neural networks does not work well with sparse data as the neural network will not learn anything if most of the inputs are zero. But, if we introduce multiple hidden layers, MLP works well with sparse data as well . This is because, the hidden layers can capture the important features even when the data is sparse. For this data set, the number of input features is 63 after one hot encoding and

just one hidden layer is enough to train the model. But if the input feature vector is of very high dimension, we might need to introduce more dense hidden layers for proper training.

For this dataset, MLP performs better for sparse dataset with relu activation. The test RMSE for relu is **0.0335.**

Which model overall generates the best results?
The best values for our different regression models are:

- **Linear Regression Model:** With ElasticNet Regularization, the best Test-RMSE obtained is **0.088504** for the best combination of Scalar, One-Hot, One-Hot, One-Hot, One-Hot
- **Random Forest Regression Model**: For 200 trees with max_depth=10 and max_features=4, we get best Test-RMSE as **0.01307**
- **Multilayer Perceptron Model**: For 37 neurons in hidden layer using the relu activation, we get the best Test-RMSE as **0.0335**
- **KNN Model**: For k=3, we get best Test-RMSE as **0.0719**

Thus, **Random Forest Regression model gives best prediction result for this dataset,** provided we tune the model with the correct parameters.

We also infer that the Network Backup Dataset is non-linear. This can be verified by the results obtained by Random Forest and KNN Regression Model. Results are better than Linear Regression Model. The Multilayer Perceptron performs better than KNN because we have introduced hidden layer in it. Hidden layer in a perceptron is used to capture non-linearity of the data. A linearly separable data can be classified by a Neural Network without using any hidden layer.