

---

## ECE219 — Large Scale Data Mining

Name: Lu Ren

Name: Tianxue Chen

Name: Yuting Tang

Name: Xiao Peng

Due Date: March 4 2018, 11:59pm

UID: 704706181

UID: 004943548

UID: 304881011

UID: 005033608

Assignment: HW 4

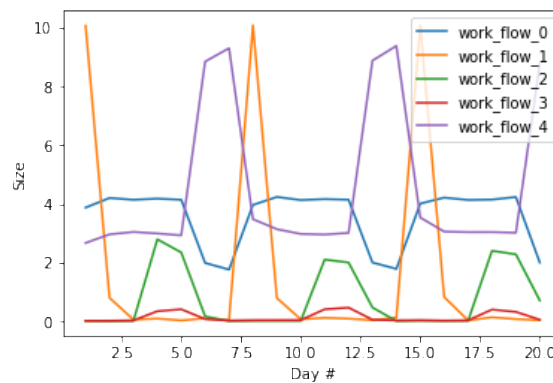
---

### Problem 1

(a)

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

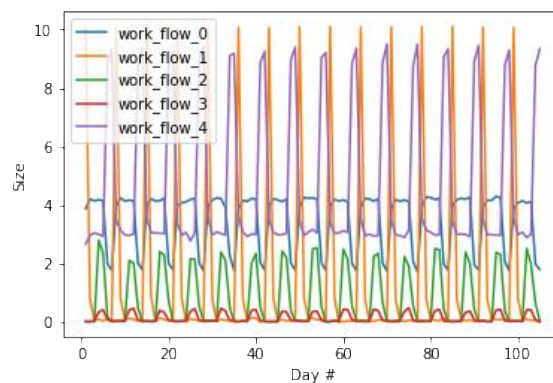
**Plot:**



(b)

**Description:** Do the same plot for the first 105-day period.

**Plot:**



**Discussion:**

It is not difficult to conclude from the two plots that the change of backup size, to some extent, shows periodicity. In other words, the plot of each workflow repeats every week (every seven days). Therefore, it is reasonable to assume that the feature Week # would not make a huge influence on the backup size.

## Problem 2

(a)

i

**Description:** First convert each categorical feature into one dimensional numerical values using scalar encoding (e.g. Monday to Sunday can be mapped to 1-7), and then directly use them to fit a basic linear regression model.

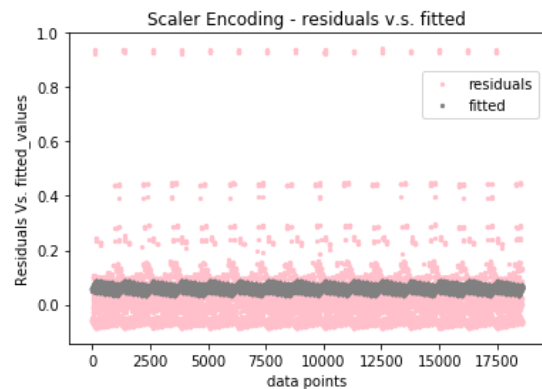
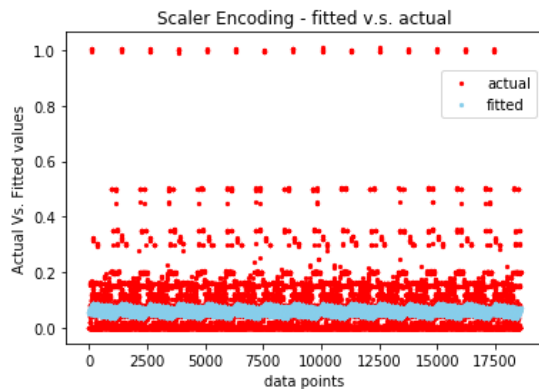
**Results:**

Time RMSE	1	2	3	4	5	6	7	8	9	10
Train	0.1032	0.1040	0.1032	0.1039	0.1032	0.1039	0.1032	0.1039	0.1032	0.1040
Test	0.1067	0.1002	0.1068	0.1004	0.1071	0.1004	0.1071	0.1005	0.1071	0.0999

The average Training RMSE is 0.1036

The average Test RMSE is 0.1036

**Plot:**



ii

**Description: Data Preprocessing:** Standardize (see the Useful Functions Section) all these numerical features, then fit and test the model. How does the fitting result change as shown in the plots?

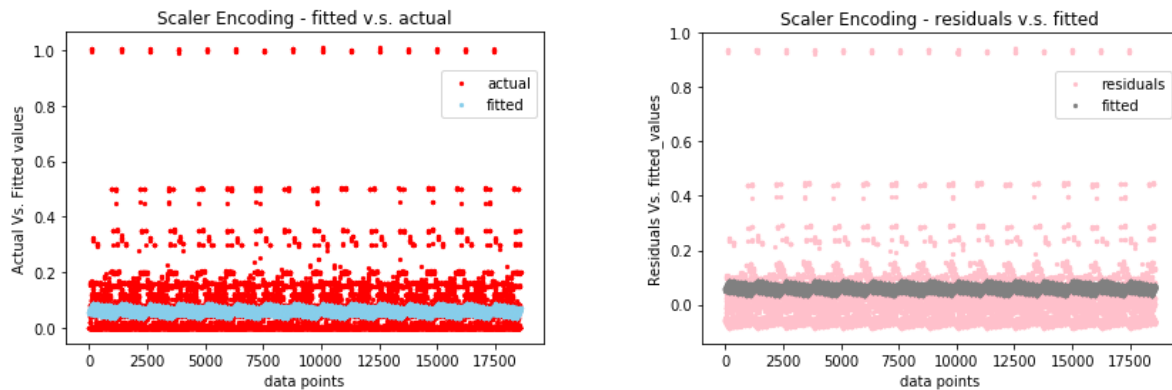
**Results:**

Time RMSE	1	2	3	4	5	6	7	8	9	10
Train	0.1032	0.1040	0.1032	0.1039	0.1032	0.1039	0.1032	0.1039	0.1032	0.1040
Test	0.1067	0.1002	0.1068	0.1004	0.1071	0.1004	0.1071	0.1005	0.1071	0.0999

The average Training RMSE is 0.1036

The average Test RMSE is 0.1036

**Plot:**



### Discussion:

The fitting result has almost no change after the process of standardization. In other words, when we choose to keep four decimals, the results are the same whether the standardization is applied or not, which means standardizing all the numerical features does not bring great benefits to the results.

### iii

**Description:** Feature Selection: Use f regression and mutual information regression measure to select three most important variables respectively. Report the three most important variables you find. Use those three most important variables to train a new linear model, does the performance improve?

### Results:

feature \ RMSE	Week #	Day of Week	Backup Start Time	Work-Flow-ID	File Name
F score	8.4501e-03	3.8816e+01	1.5074e+02	2.6139e+01	2.5320e+01
P value	9.2676e-01	4.7561e-10	1.6247e-34	3.2091e-07	4.9015e-07
Mutual info	0	0.22463089	0.2334576	0.27349208	0.43095612

Considering both the results of f regression and mutual information regression, the three most important variables are Day of Week, Backup Start Time - Hour of Day, File name.

Use those three most important variables to train a new linear model.

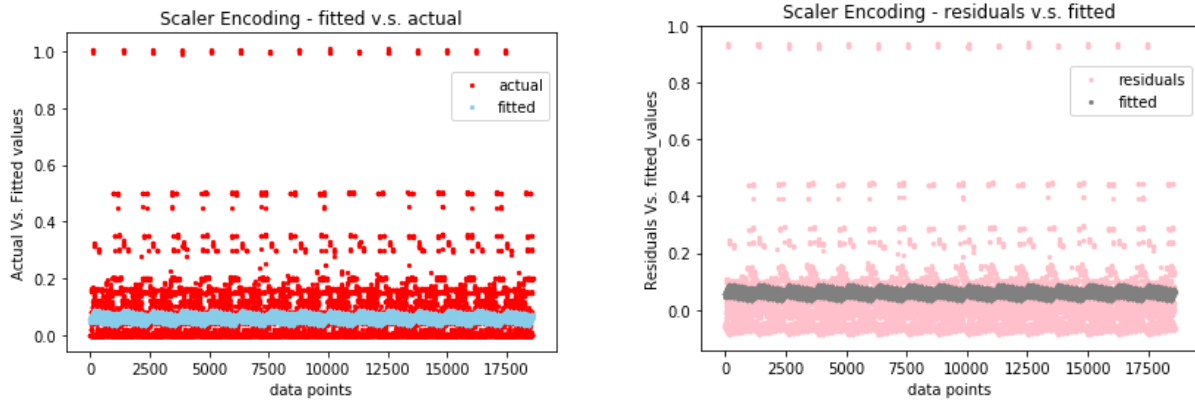
### Results:

Time \ RMSE	1	2	3	4	5	6	7	8	9	10
Train	0.1032	0.1040	0.1032	0.1039	0.1032	0.1039	0.1032	0.1039	0.1032	0.1040
Test	0.1067	0.1002	0.1068	0.1004	0.1071	0.1005	0.1070	0.1005	0.1071	0.0999

The average Training RMSE is 0.1036

The average Test RMSE is 0.1036

Plot:



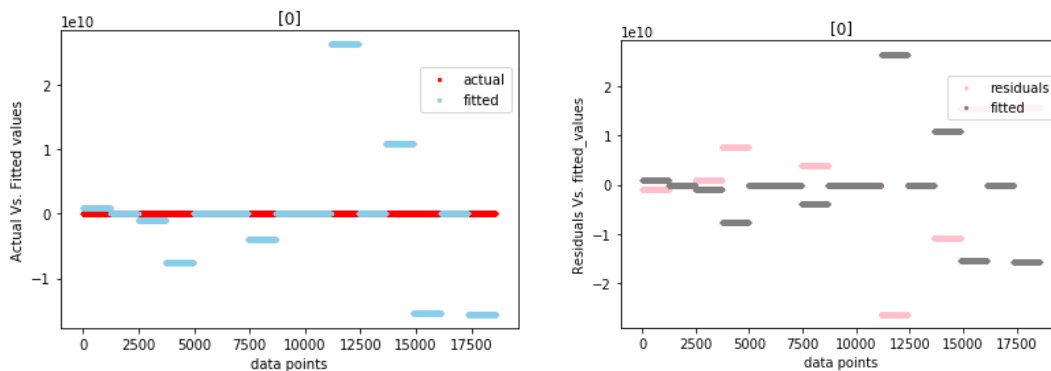
#### Discussion:

The three most important variables are Day of Week, Backup Start Time - Hour of Day and File name. However, the performance by using those three most important variables to train a new linear model does not improve much. But we could tell that the results have been improved due to the 7<sup>th</sup> set of data which has been improved. The possible reason could be that we only keep 4 decimal digits which might be too few to tell the difference.

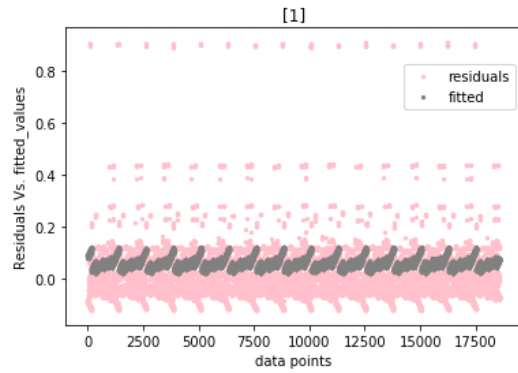
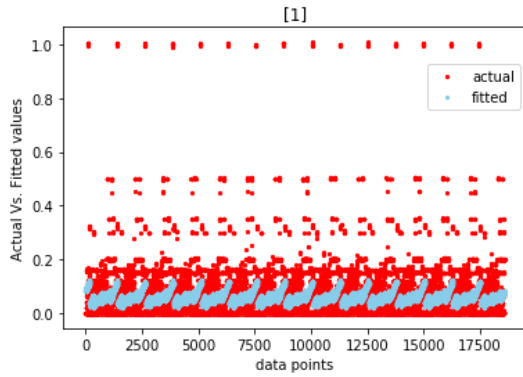
#### iv

**Description: Feature Encoding:** As explained in the preceding discussions, there are 32 possible combinations of encoding the five categorical variables. Plot the average training RMSE and test RMSE for each combination (in range 1 to 32). Which combinations achieve best performance? Can you provide an intuitive explanation?

Plot:

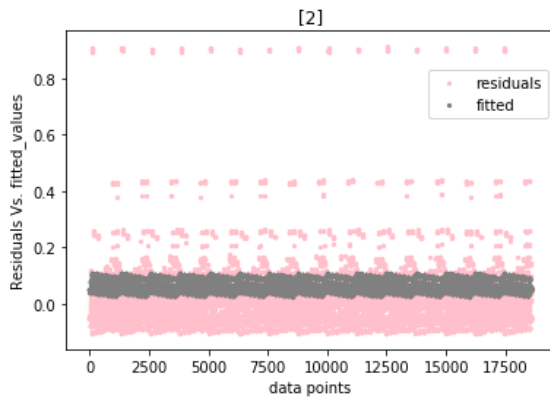
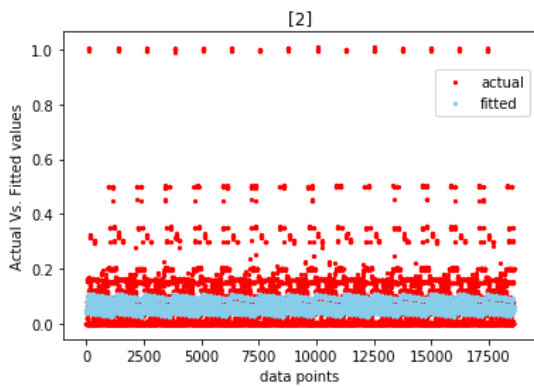


The average Test RMSE is 6.674e+09  
The average Training RMSE is 0.1036



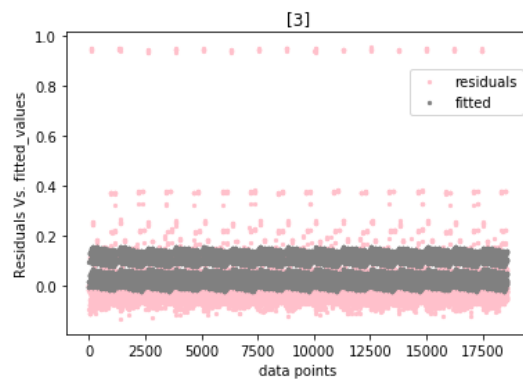
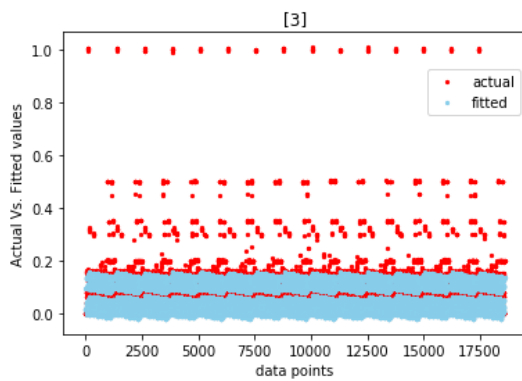
The average Test RMSE is 0.1022  
The average Training RMSE is 0.1022

---



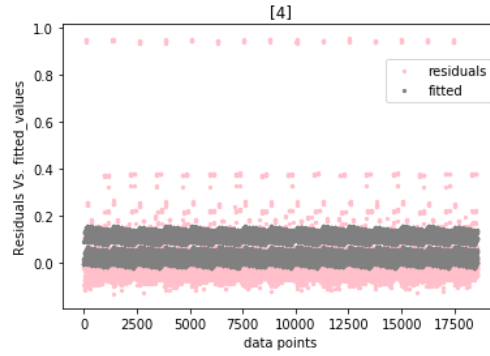
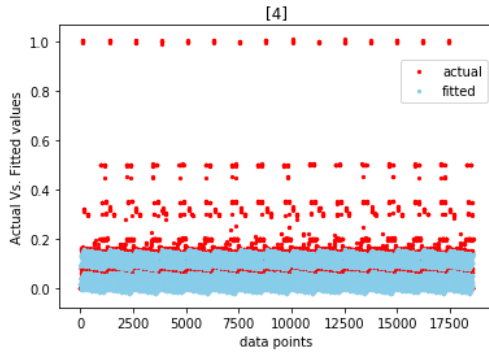
The average Test RMSE is 0.1024  
The average Training RMSE is 0.1024

---



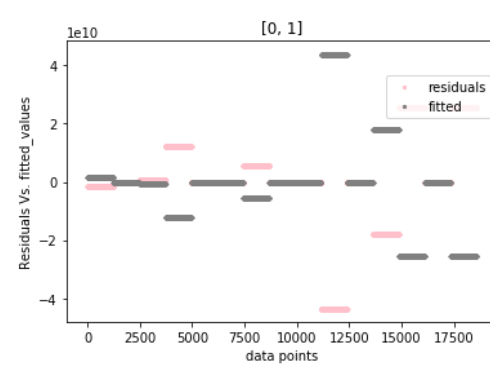
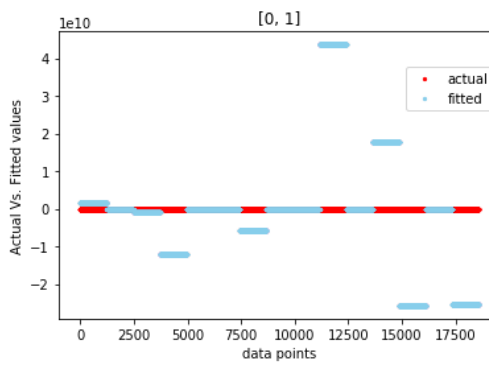
The average Test RMSE is 0.09133  
The average Training RMSE is 0.09134

---



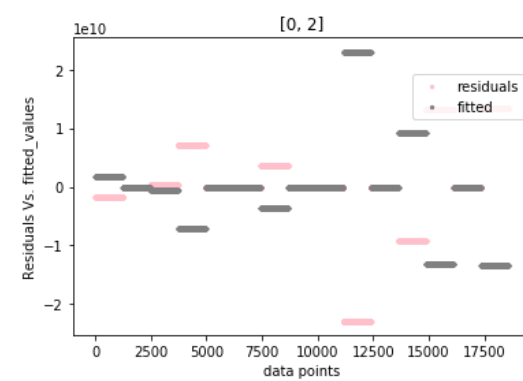
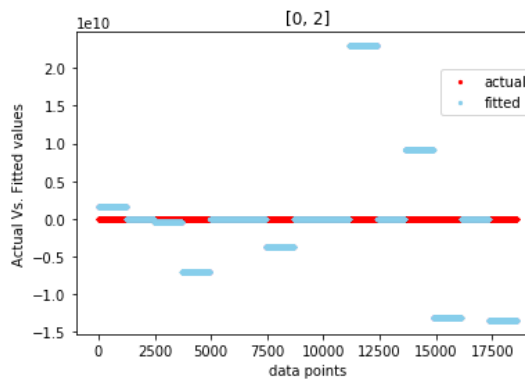
The average Test RMSE is 0.09134  
The average Training RMSE is 0.09134

---



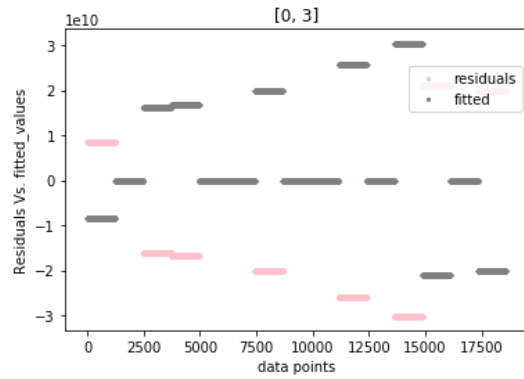
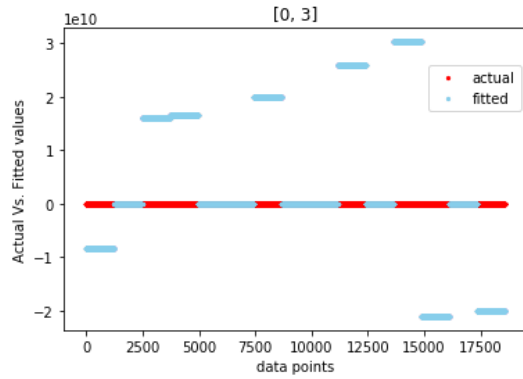
The average Test RMSE is 1.082e+10  
The average Training RMSE is 0.1021

---



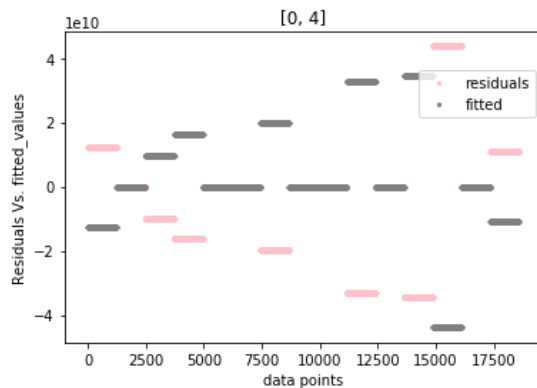
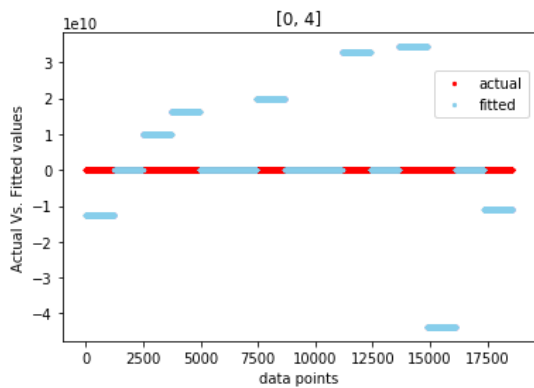
The average Test RMSE is 5.849e+09  
The average Training RMSE is 0.1024

---



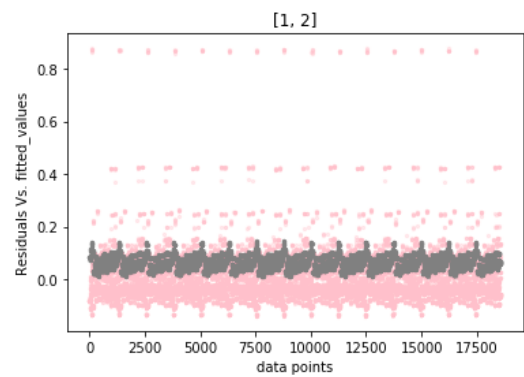
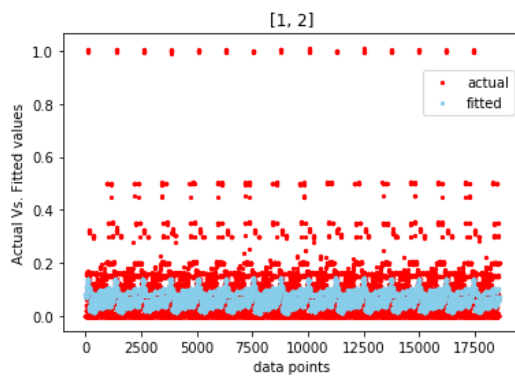
The average Test RMSE is 1.293e+10  
The average Training RMSE is 0.09133

---



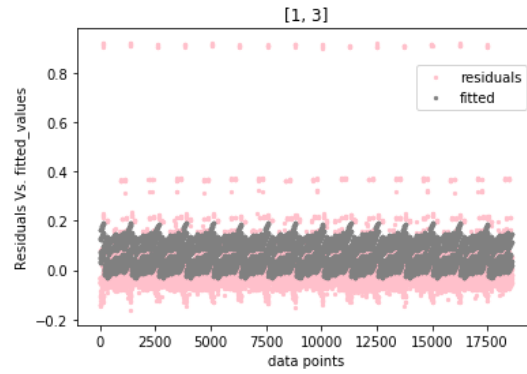
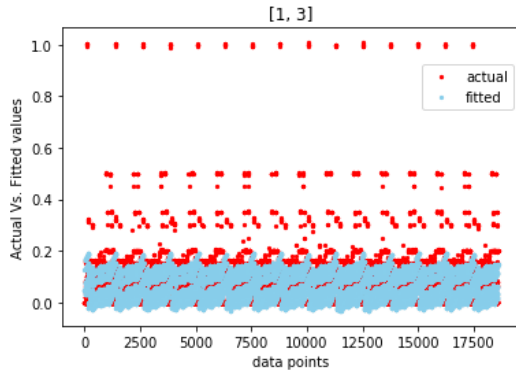
The average Test RMSE is 1.472e+10  
The average Training RMSE is 0.09133

---



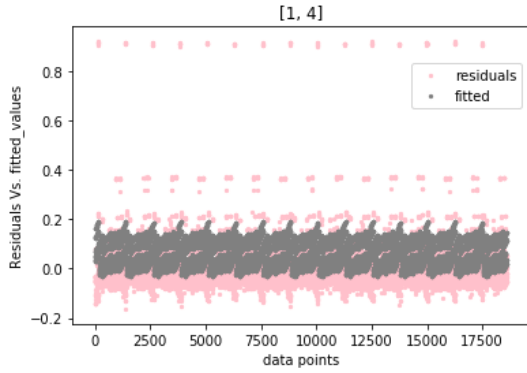
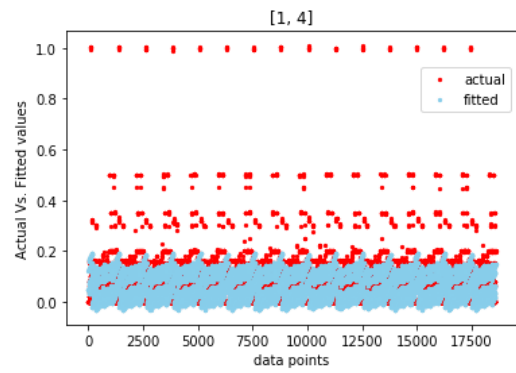
The average Test RMSE is 0.101  
The average Training RMSE is 0.1009

---



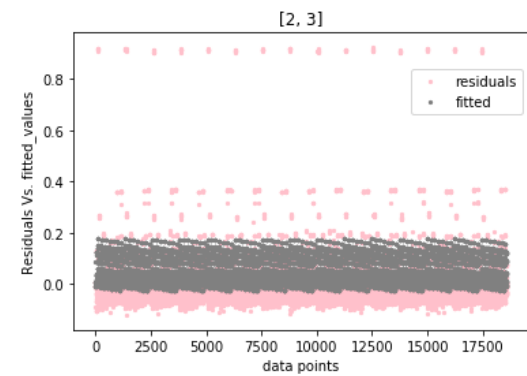
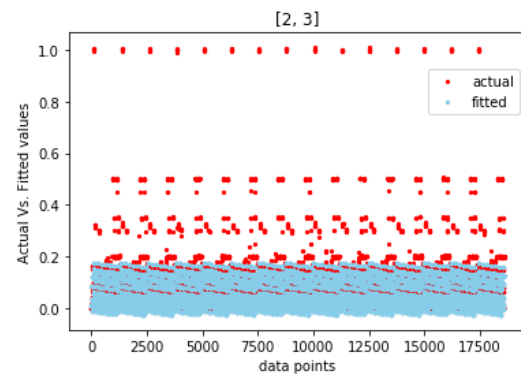
The average Test RMSE is 0.08977  
The average Training RMSE is 0.08976

---



The average Test RMSE is 0.08977  
The average Training RMSE is 0.08976

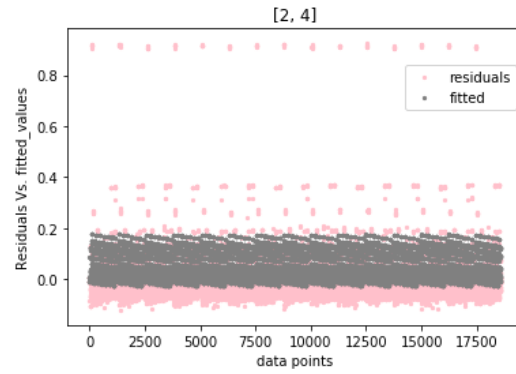
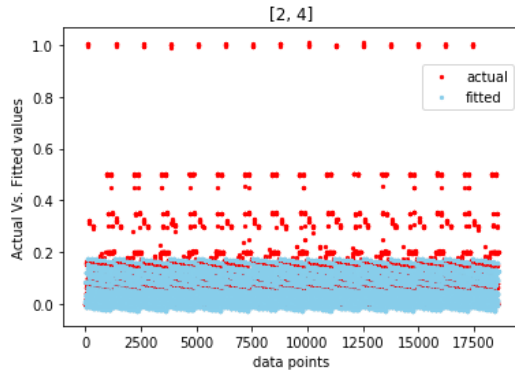
---



The average Test RMSE is 0.08997  
The average Training RMSE is 0.08995

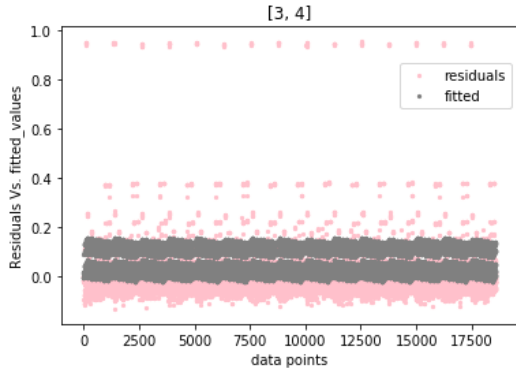
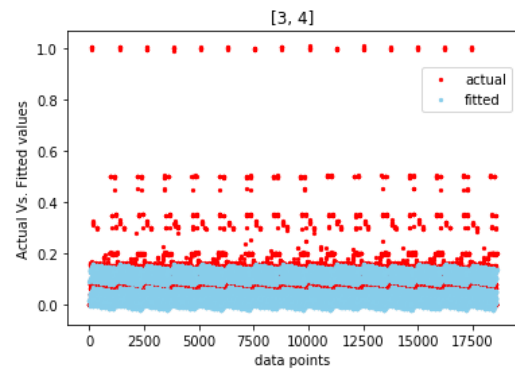
---





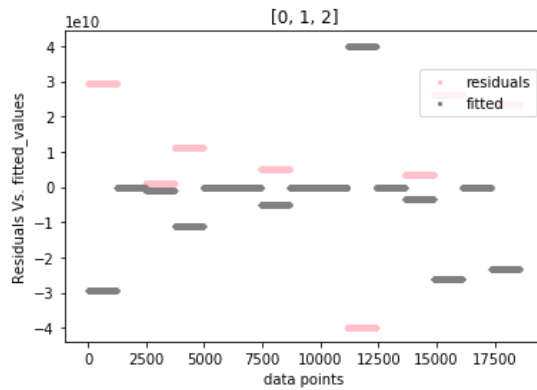
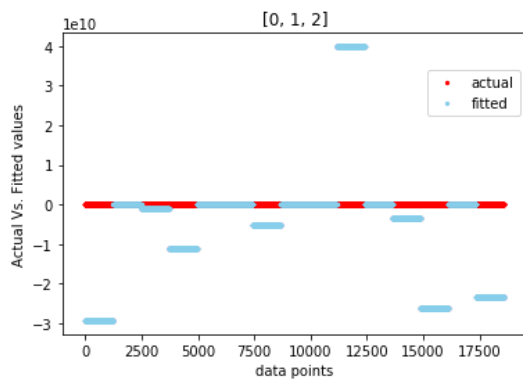
The average Test RMSE is 0.08997  
The average Training RMSE is 0.08995

---



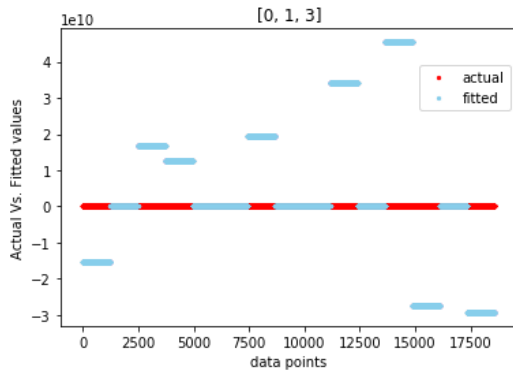
The average Test RMSE is 0.09134  
The average Training RMSE is 0.09134

---



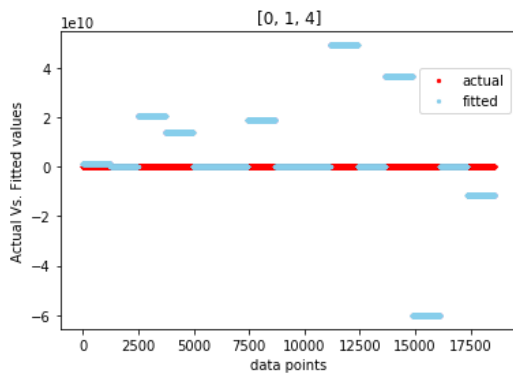
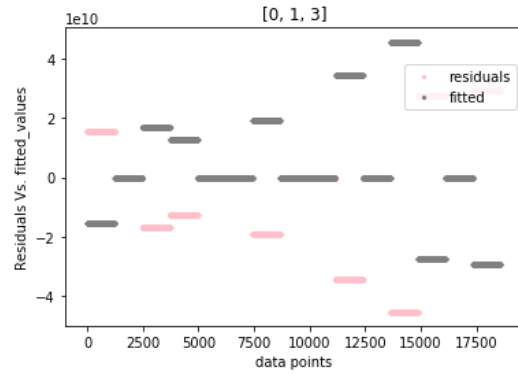
The average Test RMSE is 1.138e+10  
The average Training RMSE is 0.1009

---



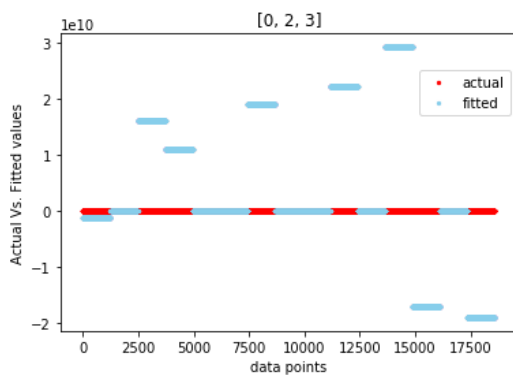
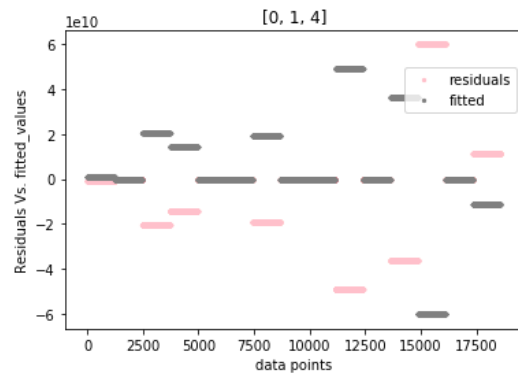
The average Test RMSE is 1.637e+10  
The average Training RMSE is 0.08975

---



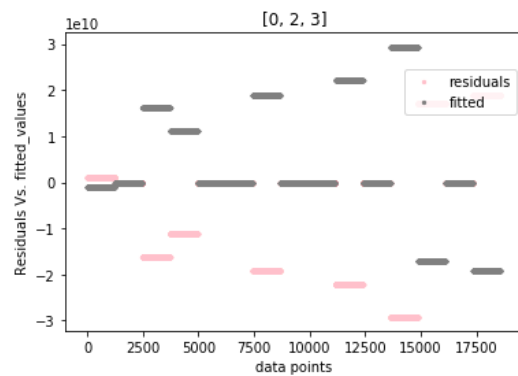
The average Test RMSE is 1.731e+10  
The average Training RMSE is 0.08975

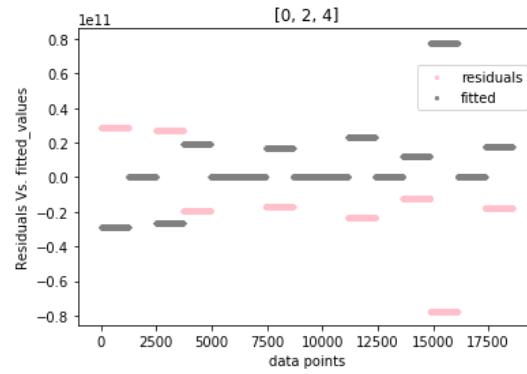
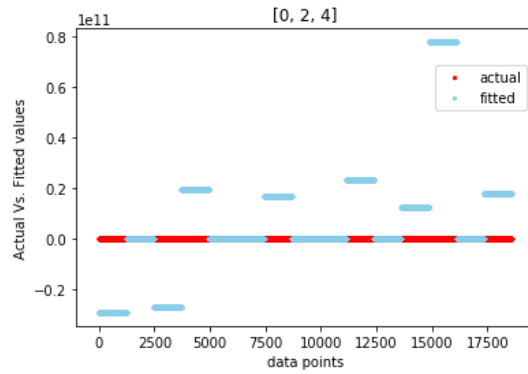
---



The average Test RMSE is 1.1e+10  
The average Training RMSE is 0.08994

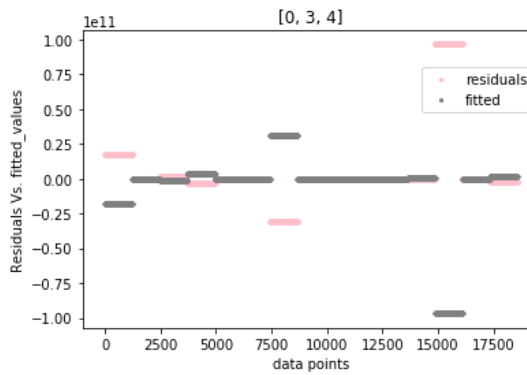
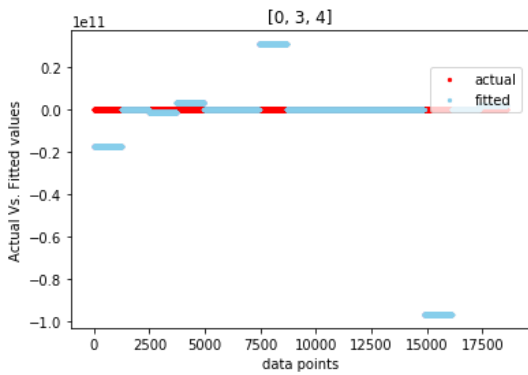
---





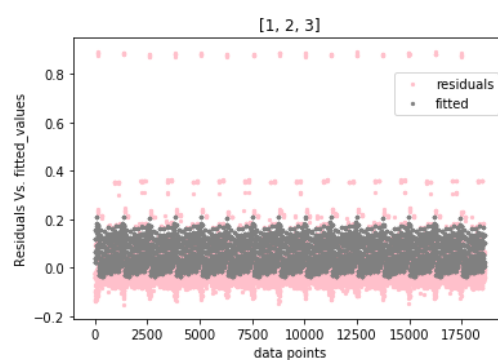
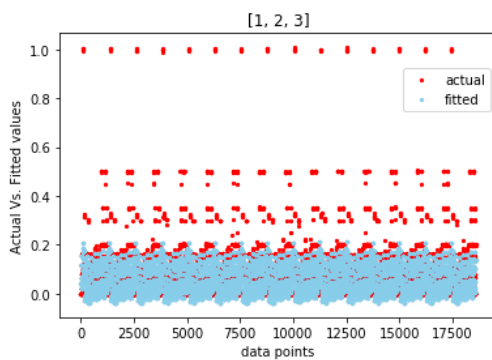
The average Test RMSE is 1.825e+10  
The average Training RMSE is 0.08995

---



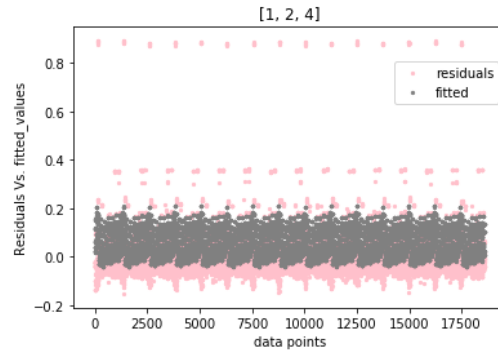
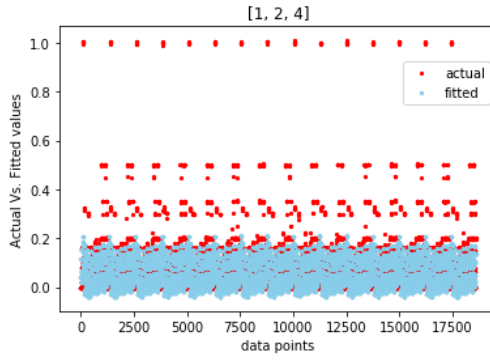
The average Test RMSE is 1.241e+10  
The average Training RMSE is 0.09134

---



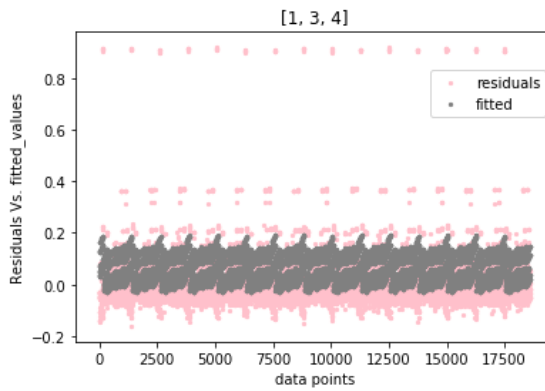
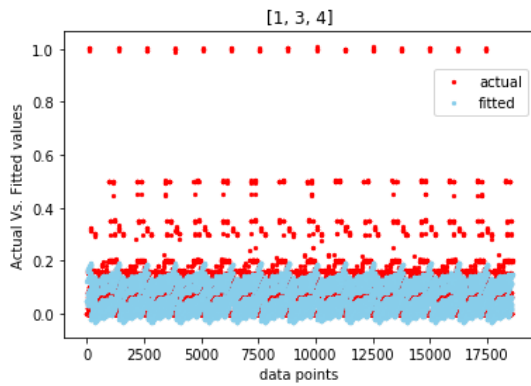
The average Test RMSE is 0.08837  
The average Training RMSE is 0.08834

---



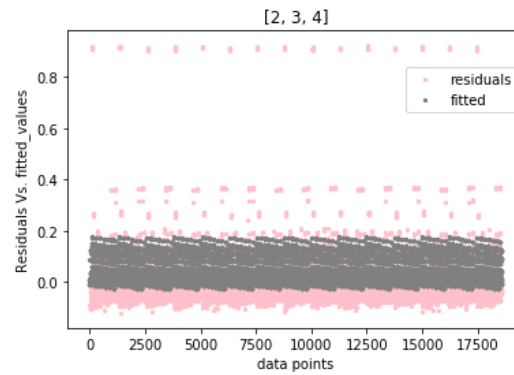
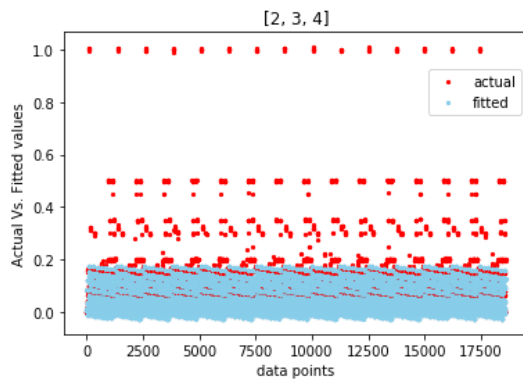
The average Test RMSE is 0.08837  
The average Training RMSE is 0.08834

---



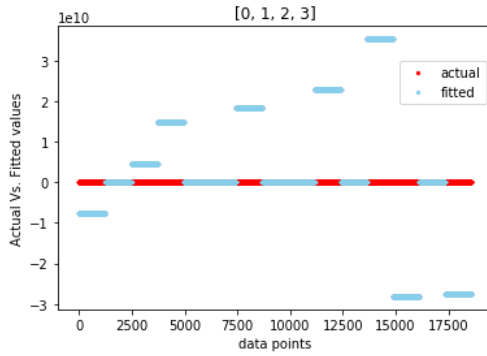
The average Test RMSE is 0.08978  
The average Training RMSE is 0.08975

---



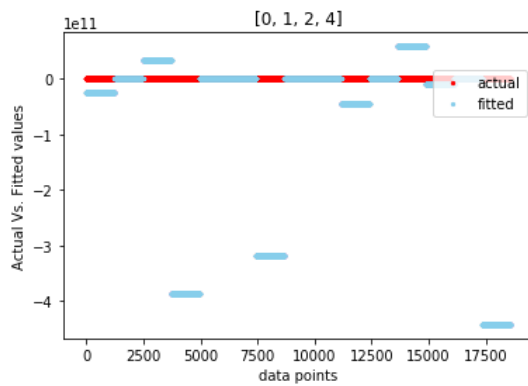
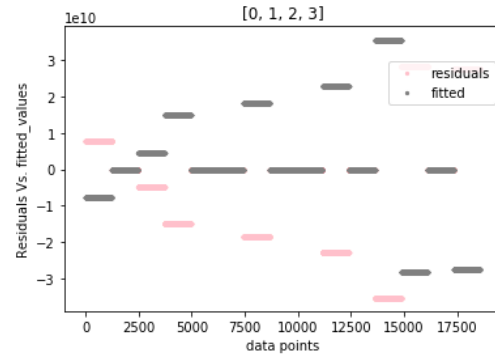
The average Test RMSE is 0.08997  
The average Training RMSE is 0.08995

---



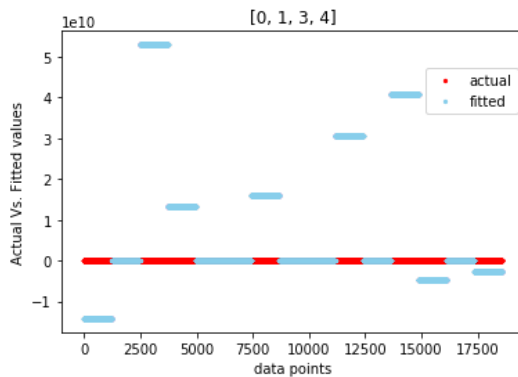
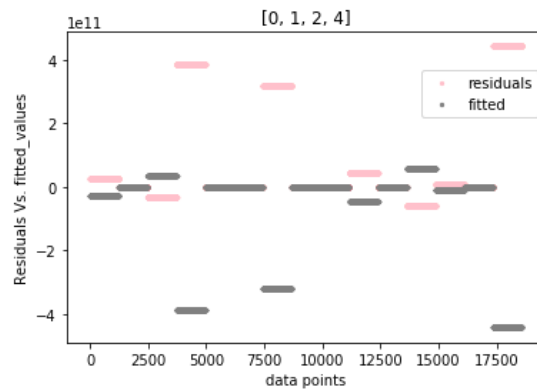
The average Test RMSE is 1.301e+10  
The average Training RMSE is 0.08834

---



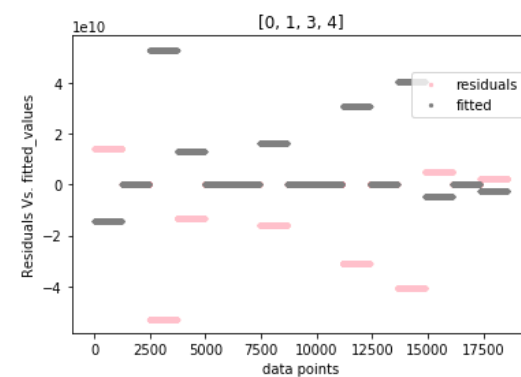
The average Test RMSE is 1.076e+11  
The average Training RMSE is 0.08834

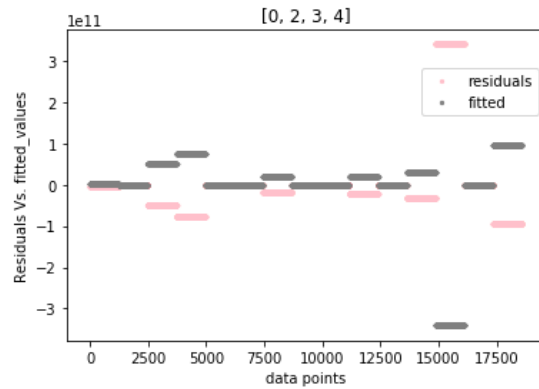
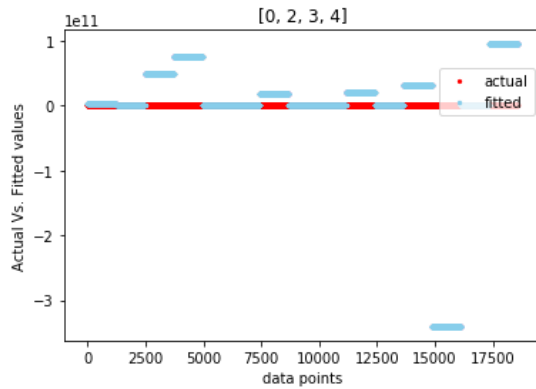
---



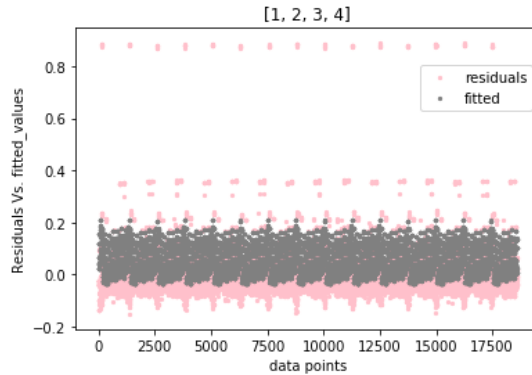
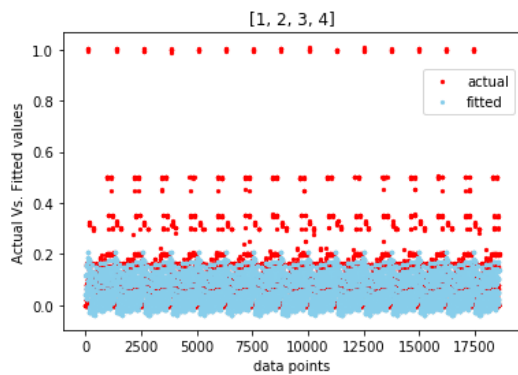
The average Test RMSE is 1.432e+10  
The average Training RMSE is 0.08975

---

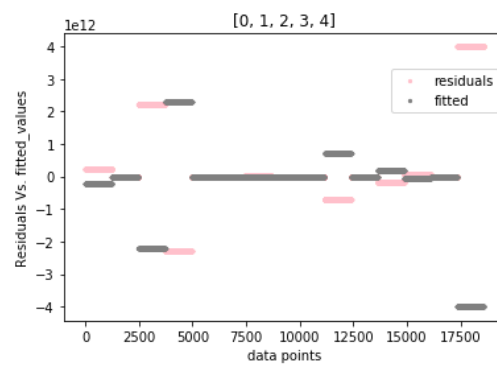
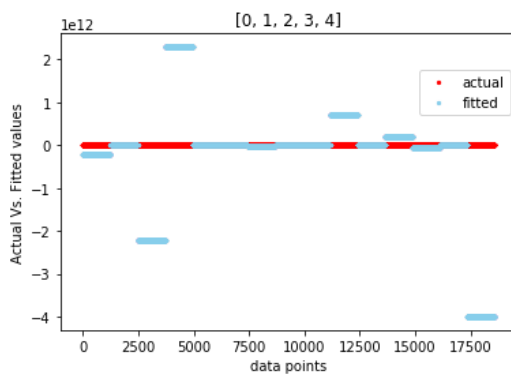




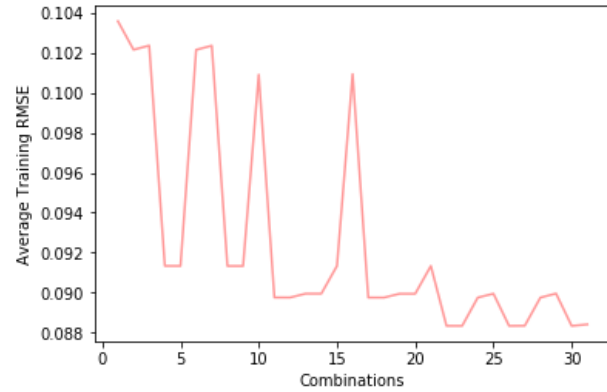
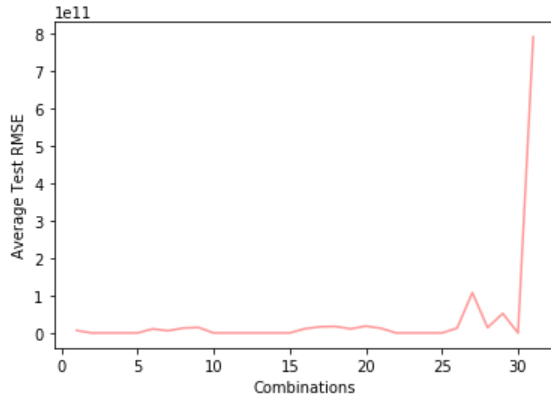
The average Test RMSE is 5.192e+10  
 The average Training RMSE is 0.08995  
 -----



The average Test RMSE is 0.08837  
 The average Training RMSE is 0.08834  
 -----



The average Test RMSE is 7.901e+11  
 The average Training RMSE is 0.08841  
 -----



### Result:

The numbers in the title of the plots 0, 1, 2, 3, 4 and their combinations represent the five features that has applied One-Hot-Encoding. According to the results, we could find that the best combination which has the lowest average RMSE is [1, 2, 4]. In other words, the combination of Day of Week, Backup Start Time – Hour of Day and File name achieves best.

### Discussion:

The combination of Day of Week, Backup Start Time – Hour of Day and File name achieves best, which is the same as the result of the problem iii. One of the intuitive reason is that 'Day of Week' and 'Backup Start Time – Hour of Day' are more correlated with backup size according to the previous problems and results. And it is easy to understand that 'File name' is also related to backup size, compared with 'Week #', which could also be concluded from the f-regression and mutual information.

What is more, as concluded in the problem i, Week # would not make a huge influence on the backup size since the change of backup size is almost the same for each week. The result shown in this part also proves the conclusion. Each combination with Week # applied One-Hot-Encoding has quite a bad performance compared with those without Week # applied. In addition, the effect of Work-Flow-ID is similar to the effect of File Name. We suppose that one possible reason is that the Work-Flow-ID is related with the File Name, which we could infer this conclusion from the .csv file.

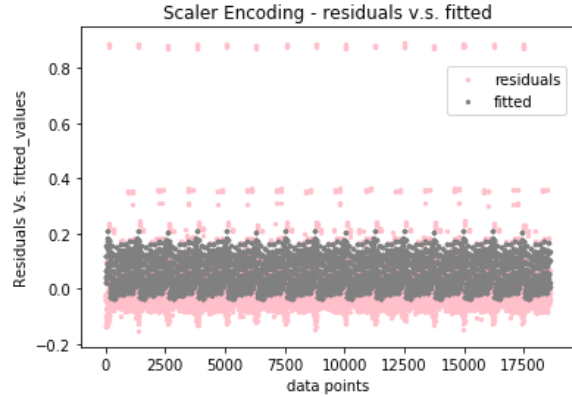
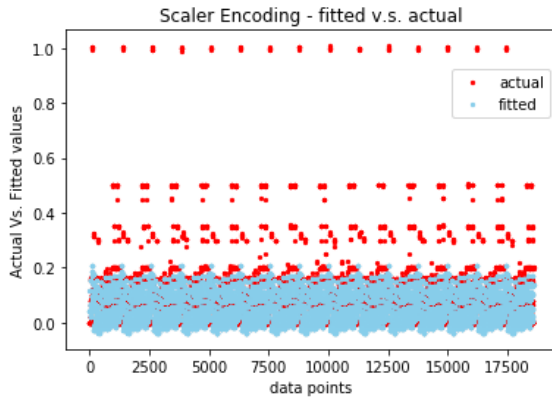
## V

**Description:** For any choice of the hyper-parameter(s) (i.e.,  $\alpha$ ,  $\lambda_1$ ,  $\lambda_2$ ) you will find one of the 32 models with the lowest Test-RMSE. Optimize over choices of  $\alpha$ ,  $\lambda_1$ ,  $\lambda_2$  to pick one good model. Compare the values of the estimated coefficients for these regularized good models, with the un-regularized best model

**Result:**

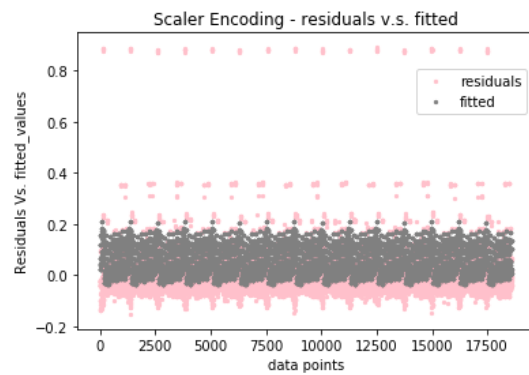
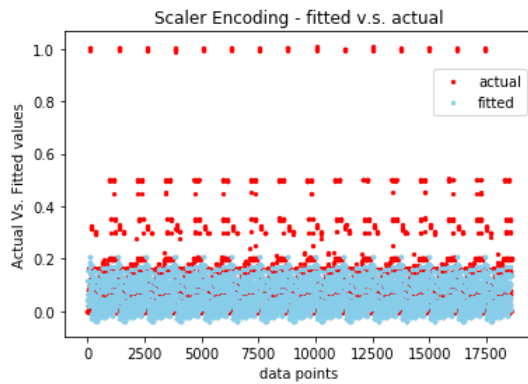
**1. Ridge Regularizer**

Alpha = 0.001000



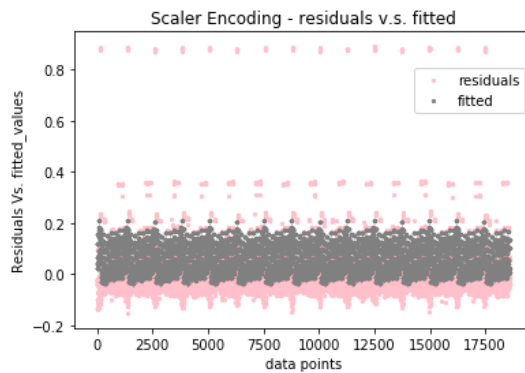
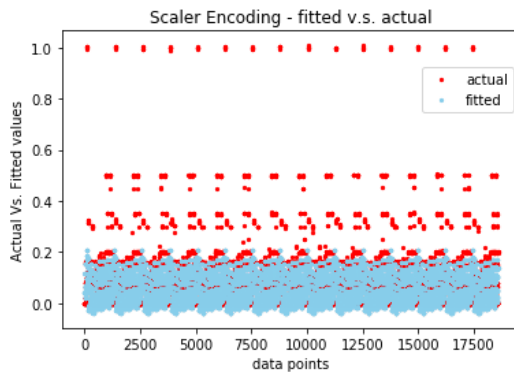
The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883358  
-----

Alpha = 0.010000



The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883358  
-----

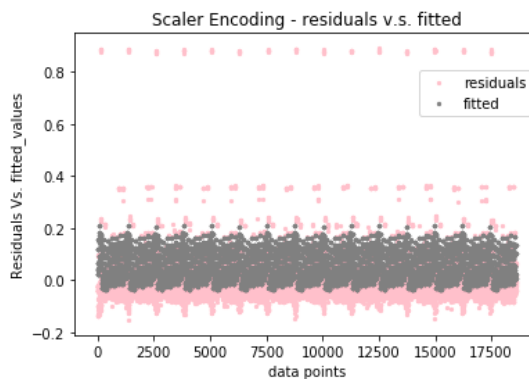
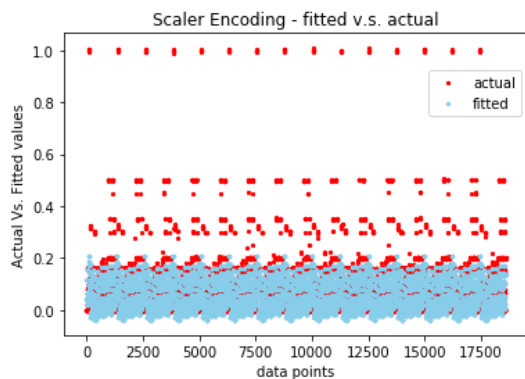
Alpha = 0.100000



The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883358  
-----



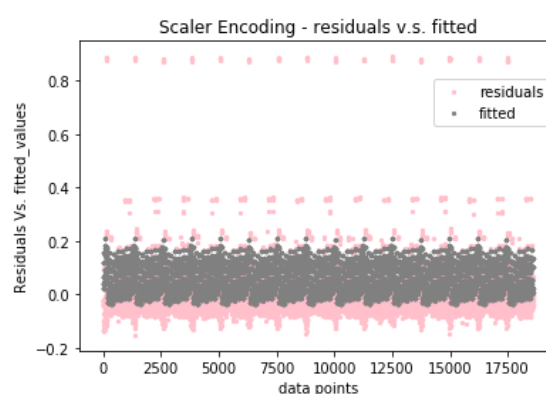
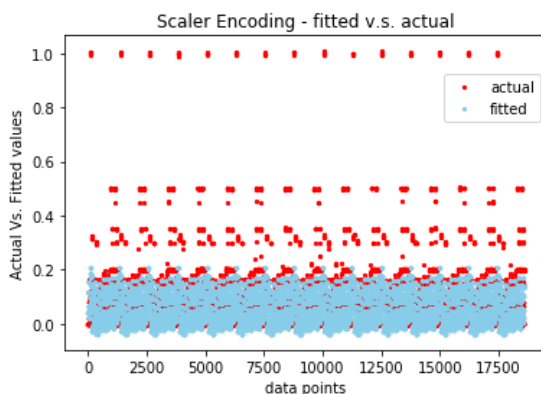
Alpha = 1.000000



The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883358

---

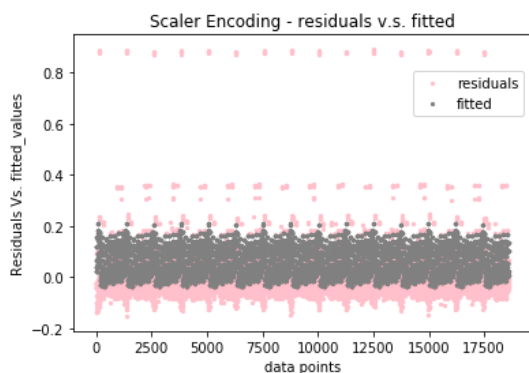
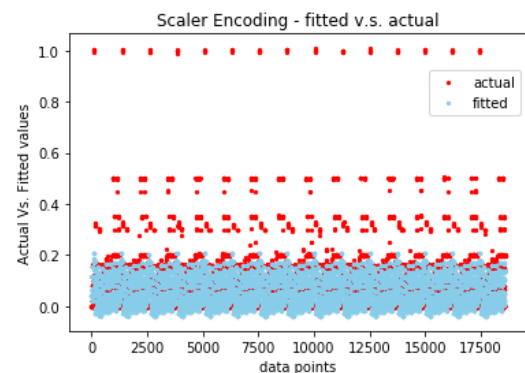
Alpha = 5.000000



The average Test RMSE is 0.0883677  
The average Training RMSE is 0.0883358

---

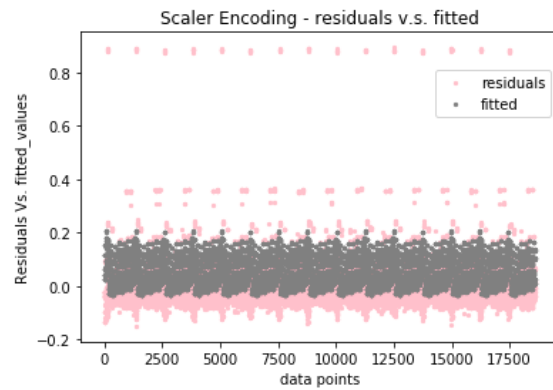
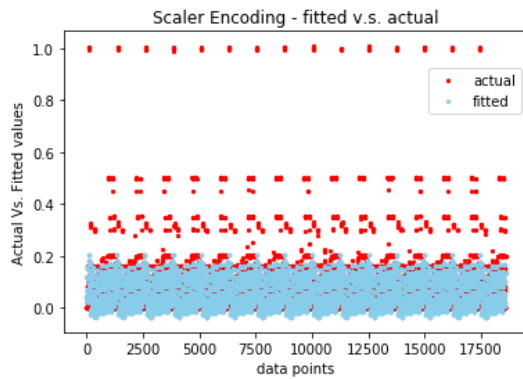
Alpha = 10.000000



The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883359

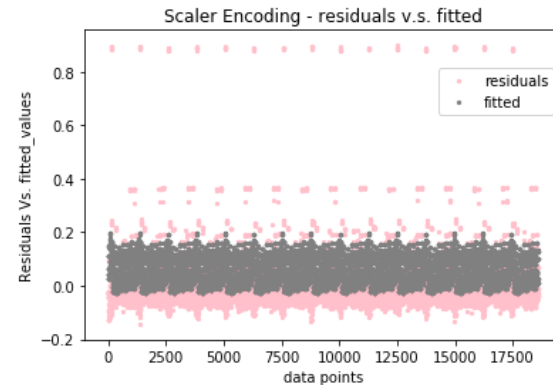
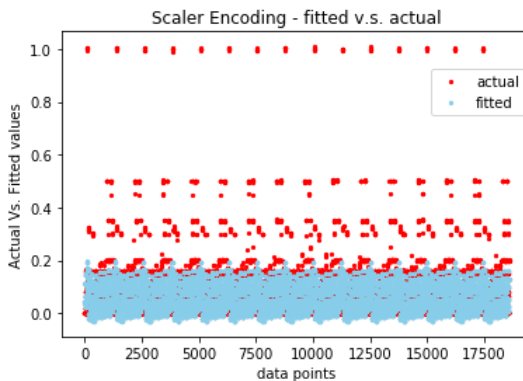
---

Alpha = 80.000000



The average Test RMSE is 0.0883778  
The average Training RMSE is 0.0883467

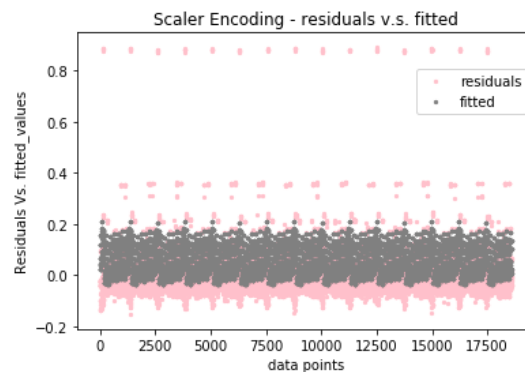
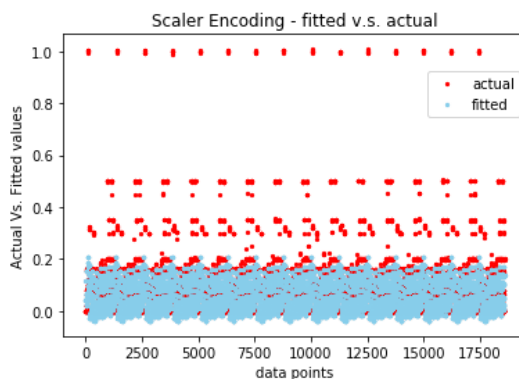
Alpha = 300.000000



The average Test RMSE is 0.0884972  
The average Training RMSE is 0.0884665

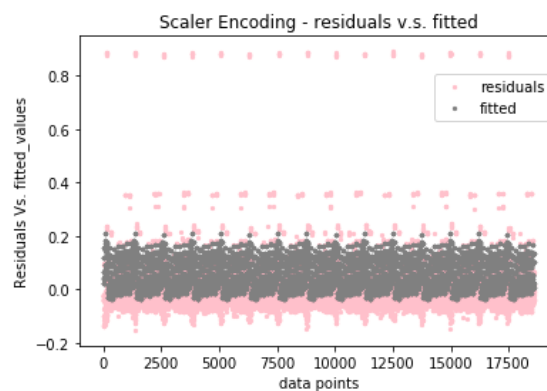
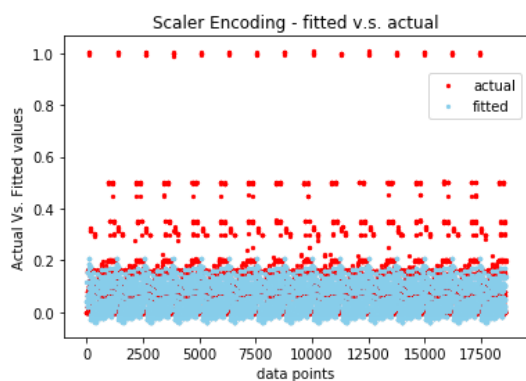
## 2. Lasso Regularizer

Alpha = 0.000001



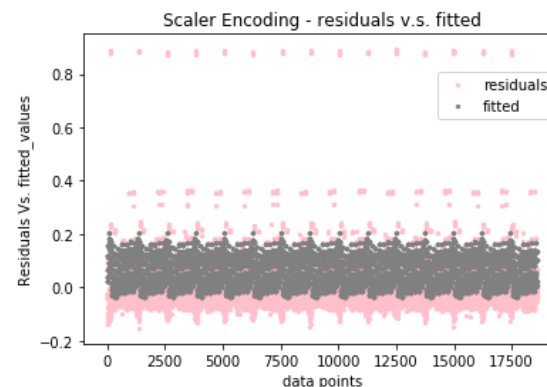
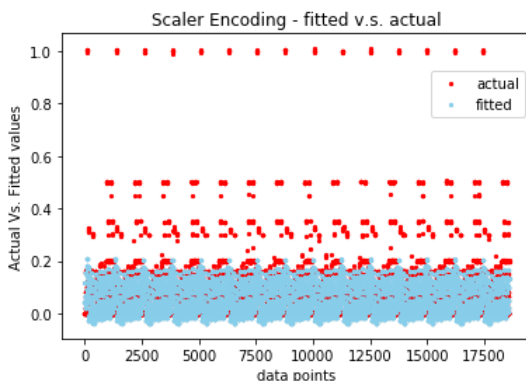
The average Test RMSE is 0.0883677  
The average Training RMSE is 0.0883358

Alpha = 0.000010



The average Test RMSE is 0.0883678  
The average Training RMSE is 0.0883358  
-----

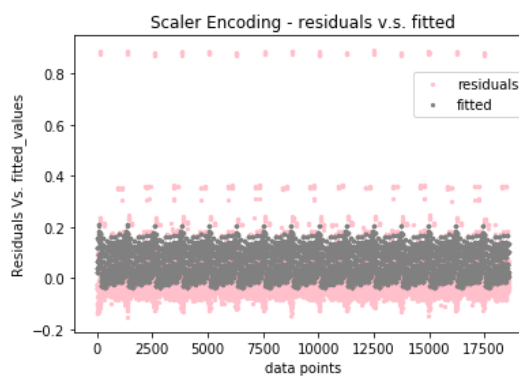
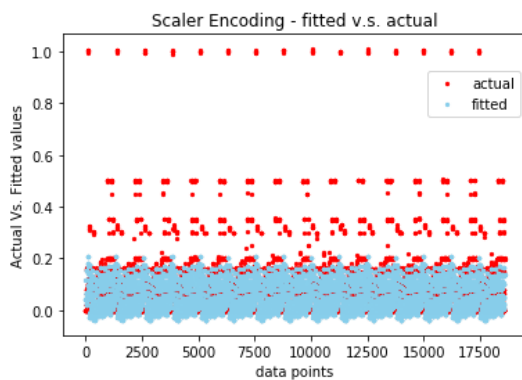
Alpha = 0.000100



The average Test RMSE is 0.0883728  
The average Training RMSE is 0.0883416  
-----

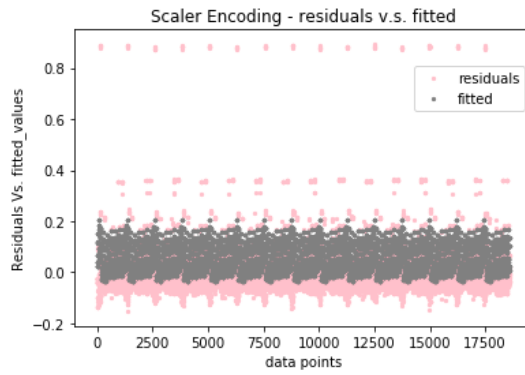
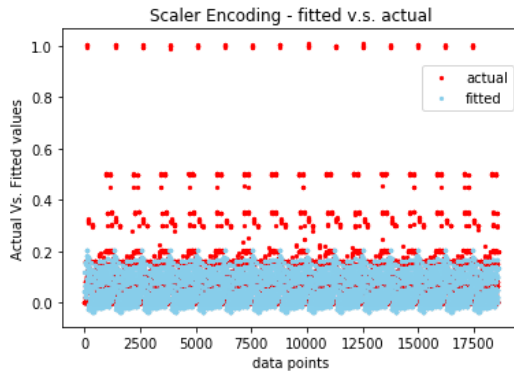
### 3. Elastic Net Regularizer

lambda1 = 0.000001, lambda2 = 0.000500



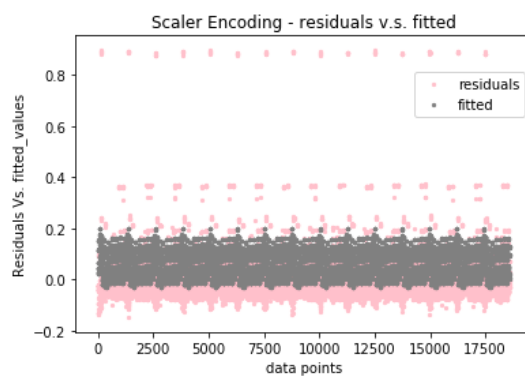
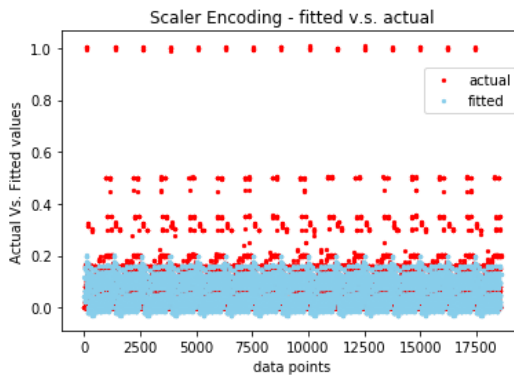
The average Test RMSE is 0.0883681  
The average Training RMSE is 0.0883363  
-----

lambda1 = 0.000100, lambda2 = 0.000500



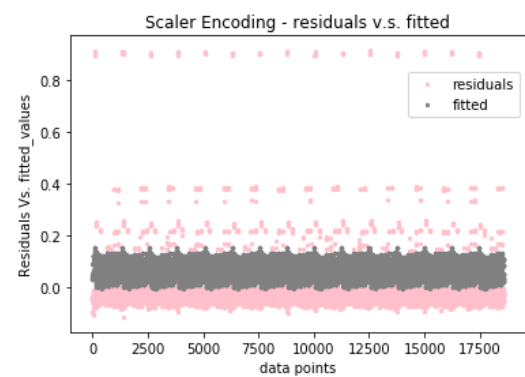
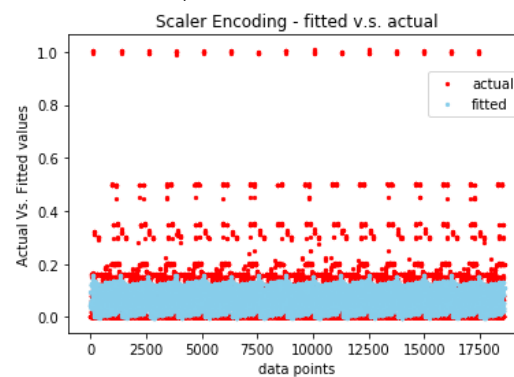
The average Test RMSE is 0.0883743  
The average Training RMSE is 0.0883439

lambda1 = 0.000500, lambda2 = 0.000500



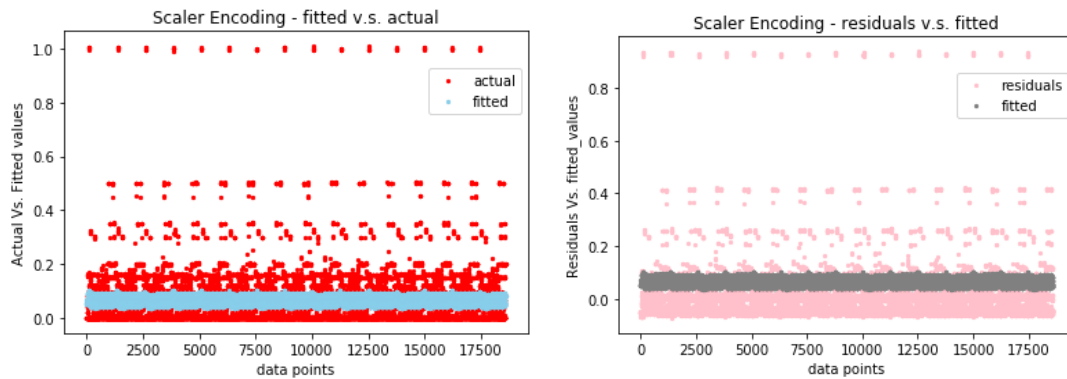
The average Test RMSE is 0.088501  
The average Training RMSE is 0.0884734

lambda1 = 0.000100, lambda2 = 0.049995



The average Test RMSE is 0.0904928  
The average Training RMSE is 0.0904604

lambda1 = 0.000500, lambda2 = 0.249875



The average Test RMSE is 0.0973349  
The average Training RMSE is 0.097309

### Discussion:

As we could see from the results, those three regularizers could all improve the performances. And the hyper-parameters like  $\alpha$ ,  $\lambda_1$ ,  $\lambda_2$  would all bring greater benefits when they are smaller for the three regularizers.

(b)

i

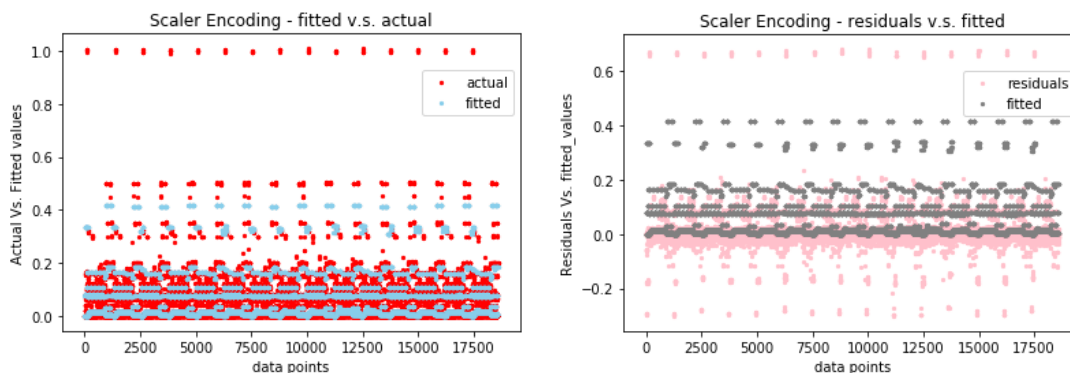
**Description:** Report Training and average Test RMSE from 10 fold cross validation (sum up each fold's square error, divide by total number of data then take square root) and Out Of Bag error you get from this initial model.

### Results:

Time RMSE \	1	2	3	4	5	6	7	8	9	10
Train	0.0601	0.0606	0.0601	0.0609	0.0597	0.0598	0.0601	0.0604	0.0601	0.0616
Test	0.0675	0.0524	0.0674	0.0527	0.0669	0.0530	0.0677	0.0520	0.0673	0.0534
OOBE	0.3350	0.3414	0.3367	0.3414	0.3325	0.3326	0.3367	0.3363	0.3361	0.3485

The average Training RMSE is 0.0603  
The average Test RMSE is 0.0600  
The average Out of Bag error is 0.3377

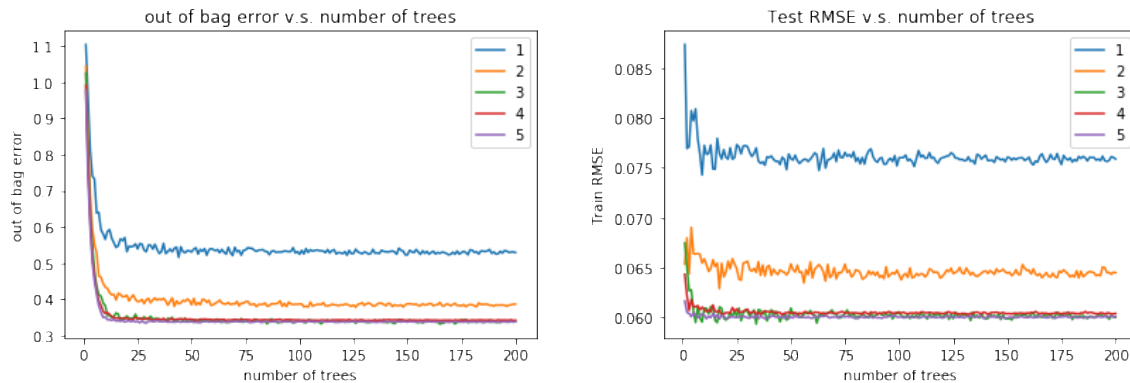
### Plot:



## ii

**Description:** Report Training and average Test RMSE from 10-fold cross validation (sum up each fold's square error, divide by total number of data then take square root) and Out of Bag error you get from this initial model.

**Plot:**



### Discussion:

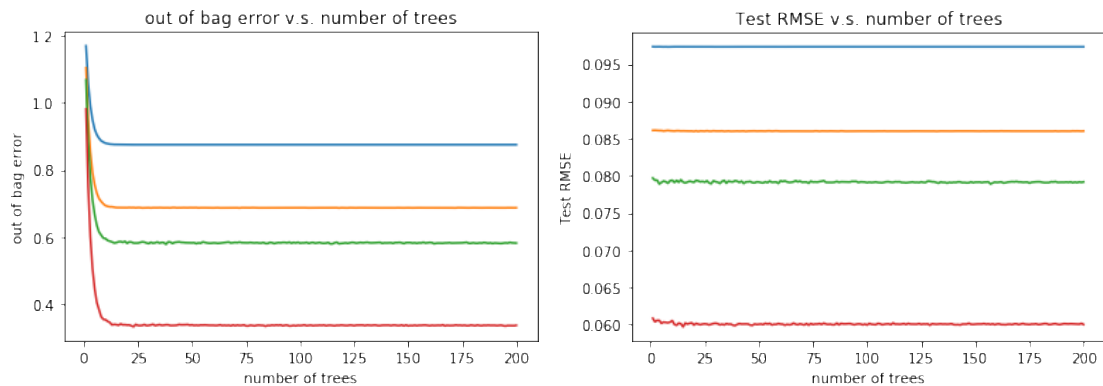
It is easy to make a conclusion that the greatest maximum number of features (5) performs best when we pay attention to the maximum number of features according to the out of bag error and RMSE, since the curve of out of bag error has slighter fluctuation and the range of the RMSE becomes smaller.

And the out of bag error would drop to a certain level and tend to be steady when number of trees increases to 20 whose RMSE is also relatively low. Combined with RMSE, we could conclude that the performance would be better when the maximum number of features is 5 and the number of trees is 20.

## iii

**Description:** Pick another parameter you want to experiment on. Plot similar figure 1 and figure 2 as above. What parameters would you pick to achieve the best performance?

**Plot:**



It is easy to make a conclusion that the greatest maximum depth (4) performs best when we pay attention to the maximum depth according to the out of bag error and RMSE, since the curve of out of bag error has slighter fluctuation and the range of the RMSE becomes smaller.

iv

### Results:

```

Feature ranking:
1. feature 4 (0.411561)
2. feature 1 (0.274410)
3. feature 3 (0.160253)
4. feature 2 (0.153769)
5. feature 0 (0.000006)

```

The feature 0, 1, 2, 3, 4 respectively represent 'Week #', 'Day of Week', 'Backup Start Time – Hour of Day', 'Work-Flow-ID', 'File Name'. And the result is consistent with the previous results except that 'File Name' seems to be more important than 'Work-Flow-ID'. And as supposed before, those two features is related, which, to some extent, could be understood. The most important feature is File Name, while the Week # could not make a huge difference to the result.

**Description:** Visualize your decision trees. Pick any tree (estimator) in best random forest (with max depth=4) and plot its structure, which is the root node in this decision tree? Is it the most important feature according to the feature importance reported by the regressor?

```

graph TD
    Root["X[4] ≤ 23.5  
mse = 0.012  
samples = 11672  
value = 0.062"]
    Root -- True --> Node1["X[3] ≤ 0.5  
mse = 0.01  
samples = 9311  
value = 0.043"]
    Root -- False --> Node2["X[1] ≤ 5.5  
mse = 0.011  
samples = 2361  
value = 0.135"]
    Node1 -- True --> Node3["X[2] ≤ 11.0  
mse = 0.003  
samples = 2295  
value = 0.101"]
    Node1 -- False --> Node4["X[1] ≤ 1.5  
mse = 0.011  
samples = 7016  
value = 0.025"]
    Node2 -- True --> Node5["X[2] ≤ 15.0  
mse = 0.001  
samples = 1688  
value = 0.086"]
    Node2 -- False --> Node6["X[2] ≤ 7.0  
mse = 0.017  
samples = 673  
value = 0.255"]
    Node3 -- True --> Node7["X[2] ≤ 7.0  
mse = 0.001  
samples = 1175  
value = 0.064"]
    Node3 -- False --> Node8["X[1] ≤ 5.5  
mse = 0.001  
samples = 1175  
value = 0.138"]
    Node4 -- True --> Node9["X[4] ≤ 11.5  
mse = 0.065  
samples = 974  
value = 0.109"]
    Node4 -- False --> Node10["X[1] ≤ 5.5  
mse = 0.001  
samples = 6042  
value = 0.012"]
    Node5 -- True --> Node11["X[2] ≤ 7.0  
mse = 0.001  
samples = 1109  
value = 0.093"]
    Node5 -- False --> Node12["X[0] ≤ 1.5  
mse = 0.001  
samples = 579  
value = 0.072"]
    Node6 -- True --> Node13["X[2] ≤ 3.0  
mse = 0.002  
samples = 216  
value = 0.161"]
    Node6 -- False --> Node14["X[2] ≤ 15.0  
mse = 0.001  
samples = 457  
value = 0.297"]
    Node7 -- True --> Leaf1["mse = 0.001  
samples = 778  
value = 0.057"]
    Node7 -- False --> Leaf2["mse = 0.001  
samples = 397  
value = 0.077"]
    Node8 -- True --> Leaf3["mse = 0.002  
samples = 785  
value = 0.166"]
    Node8 -- False --> Leaf4["mse = 0.0  
samples = 335  
value = 0.36"]
    Node9 -- True --> Leaf5["mse = 0.124  
samples = 282  
value = 0.36"]
    Node9 -- False --> Leaf6["mse = 0.0  
samples = 692  
value = 0.091"]
    Node10 -- True --> Leaf7["mse = 0.001  
samples = 4042  
value = 0.016"]
    Node10 -- False --> Leaf8["mse = 0.0  
samples = 2000  
value = 0.081"]
    Node11 -- True --> Leaf9["mse = 0.001  
samples = 563  
value = 0.081"]
    Node11 -- False --> Leaf10["mse = 0.0  
samples = 546  
value = 0.107"]
    Node12 -- True --> Leaf11["mse = 0.0  
samples = 34  
value = 0.076"]
    Node12 -- False --> Leaf12["mse = 0.0  
samples = 545  
value = 0.172"]
    Node13 -- True --> Leaf13["mse = 0.002  
samples = 109  
value = 0.139"]
    Node13 -- False --> Leaf14["mse = 0.001  
samples = 107  
value = 0.183"]
    Node14 -- True --> Leaf15["mse = 0.007  
samples = 228  
value = 0.416"]
    Node14 -- False --> Leaf16["mse = 0.001  
samples = 229  
value = 0.183"]
  
```

**Discussion:**

Firstly we achieved the .dot file and then input the 'dot -Tpdf DecisionTree.dot -o DecisionTree.pdf' in the command window to transfer the .dot file into .pdf file to get the result. And we could see the structure of the decision tree from the result above.

**(c)**

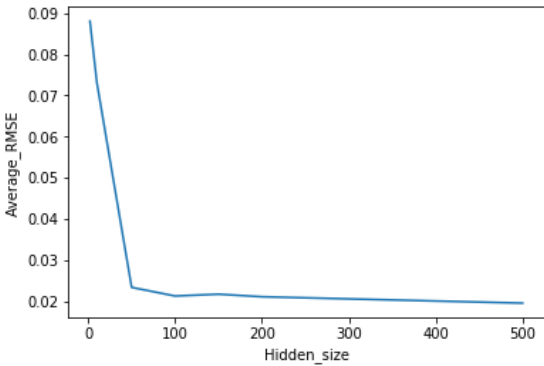
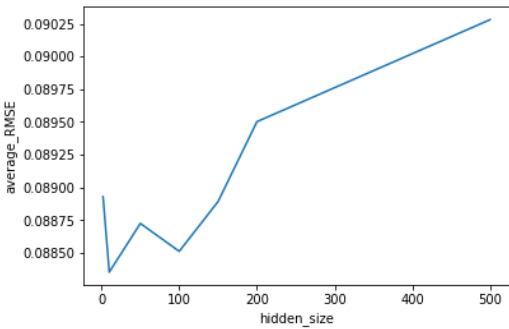
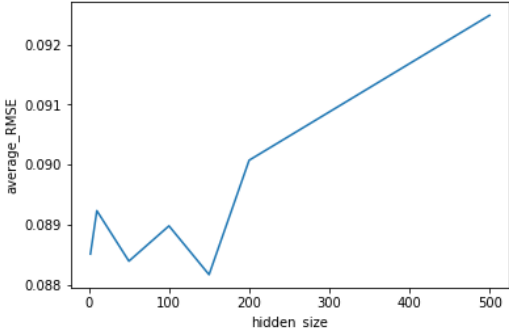
Description: Now use a neural network regression model (one hidden layer) with all features one-hot encoded. Parameters:

- Number of hidden units
- Activity Function (relu, logistic, tanh)

Plot Test-RMSE as a function of the number of hidden units for different activity functions. Report the best combination.

**Results:**

Given the hidden units = [2,10,50,100,150,200,500], average testing RMSE results for different activity functions are given below:

Activity	ReLU	
Results		
Best param	500	
Test_RMSE	0.0196	
Activity	Logistic	tanh
Result		
Best param	10	200
Test_RMSE	0.08835	0.08817



**Discussion:**

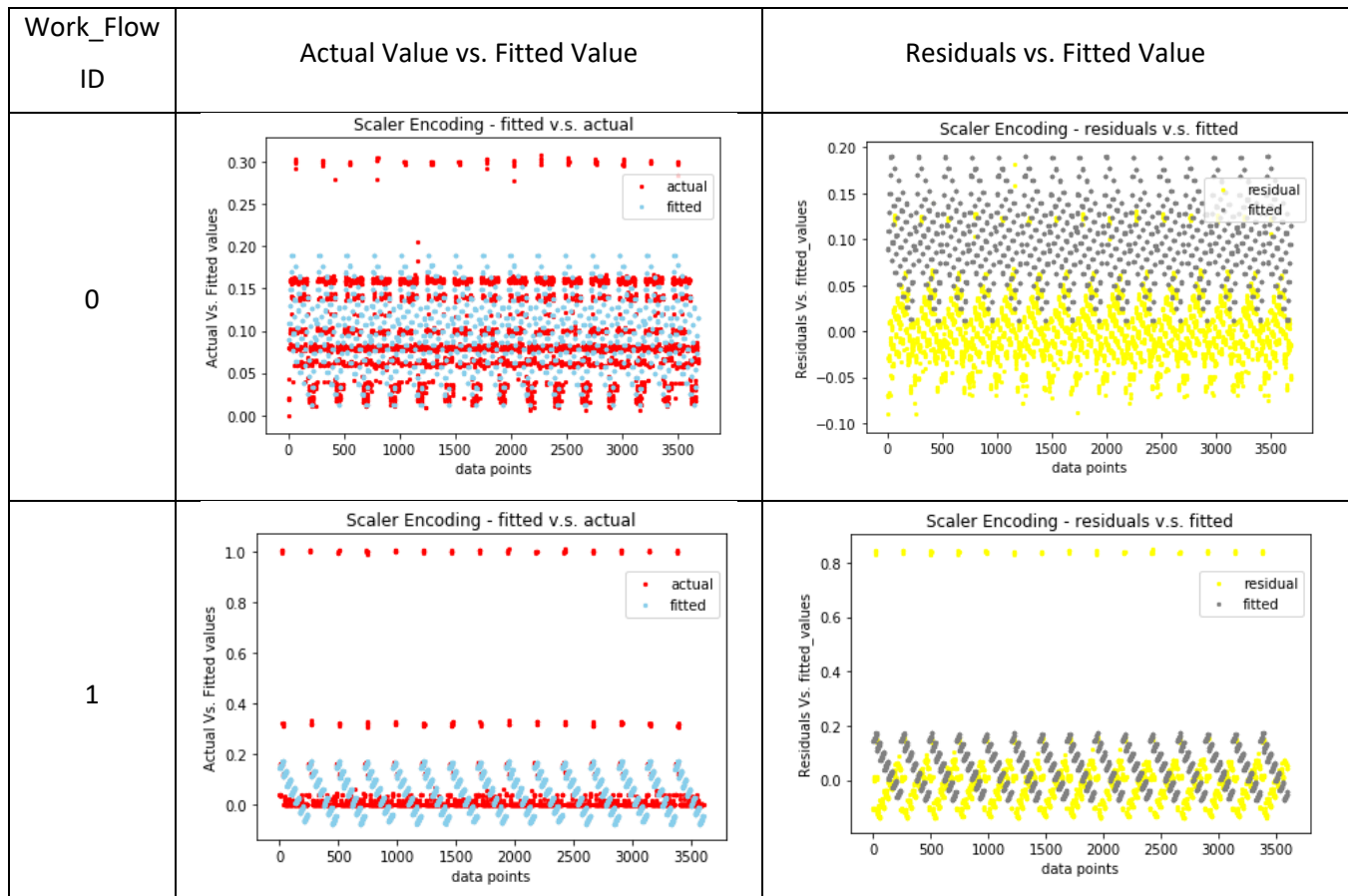
From the plot we can tell, the best combination is using large hidden units with activity ReLU. Even though we cut the number of hidden units at 500, we can anticipate the tendency that with more hidden units, we will have lower testing RMSE using ReLU. But for logistic and tanh activity, testing RMSE will be higher with larger hidden units.

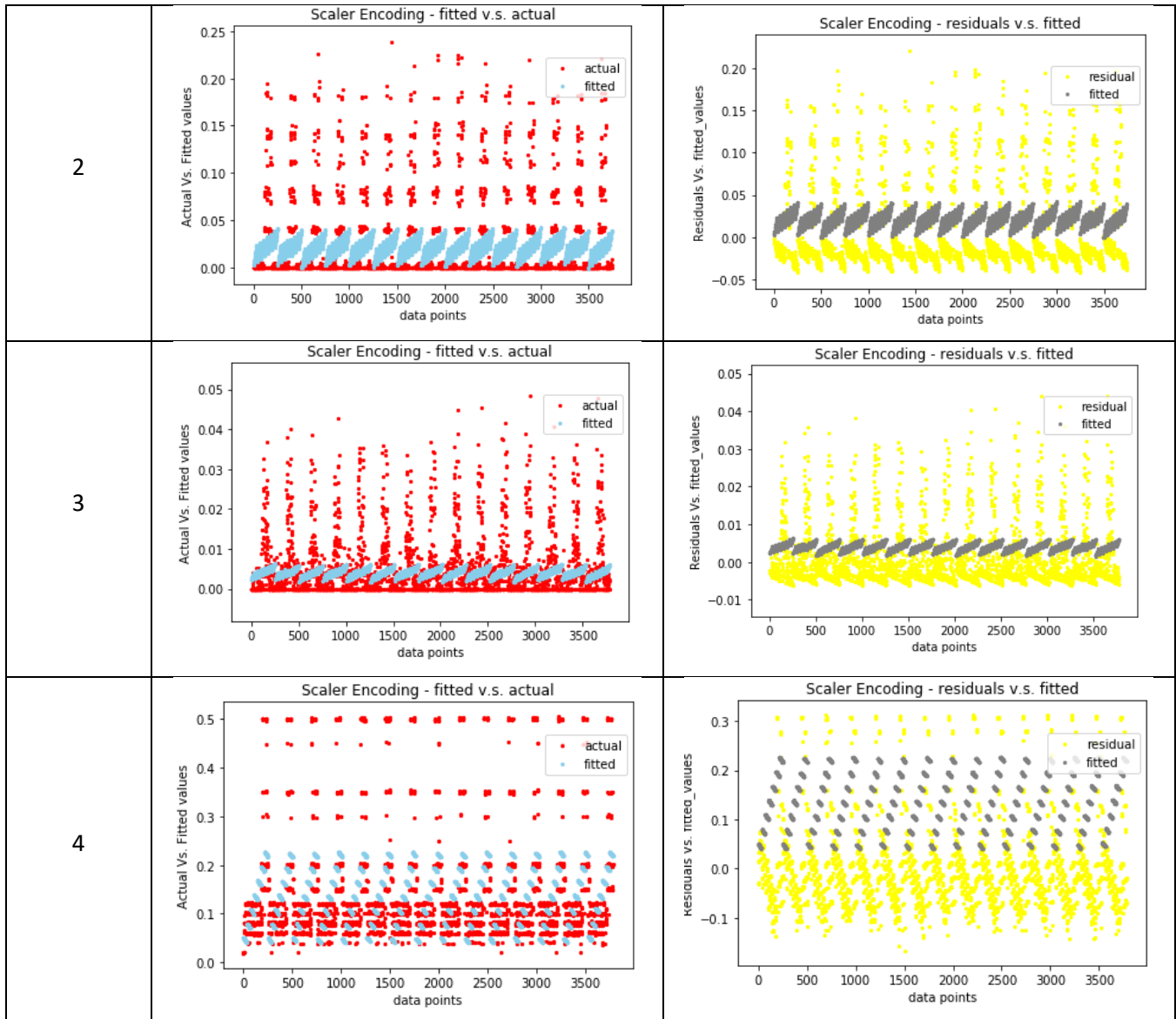
**(d)****i**

**Description:** Using linear regression model. Explain if the fit is improved?

**Results:**

Work_Flow_ID	0	1	2	3	4
Training_RMSE	0.0358	0.1487	0.0429	0.0072	0.0859
Testing_RMSE	0.0359	0.1471	0.0426	0.0072	0.0853





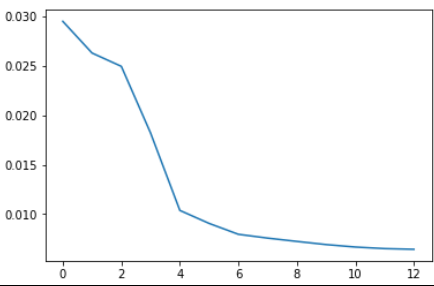
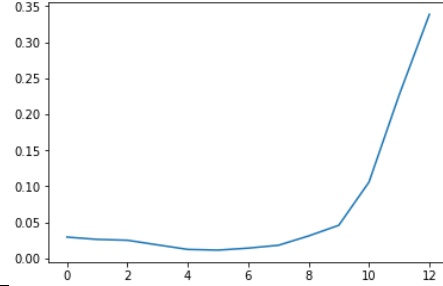
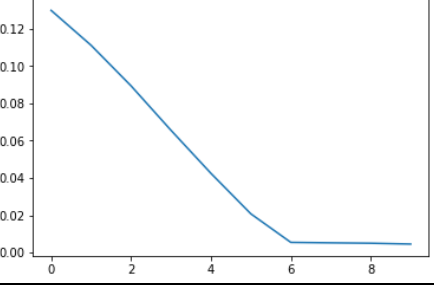
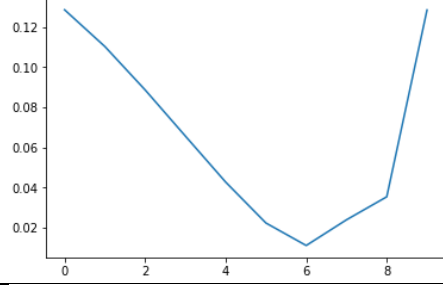
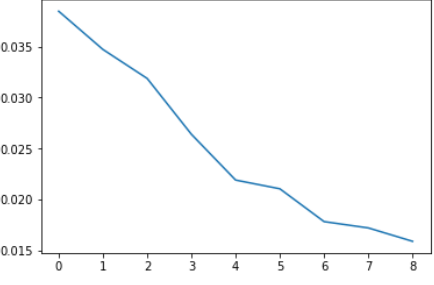
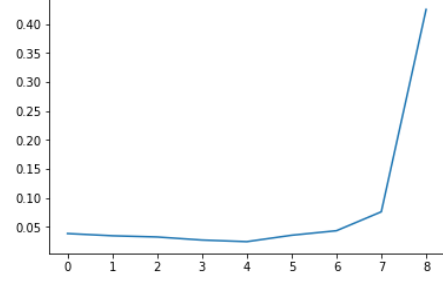
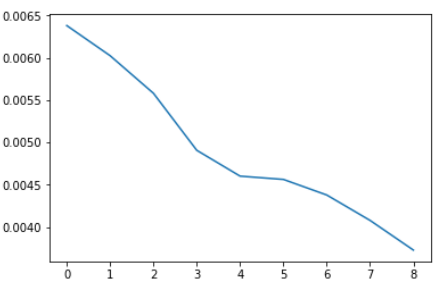
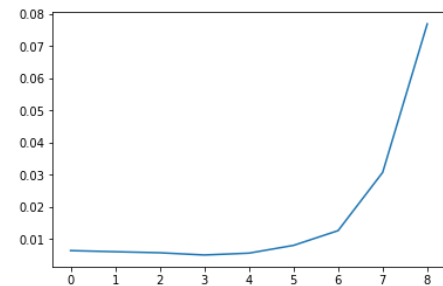
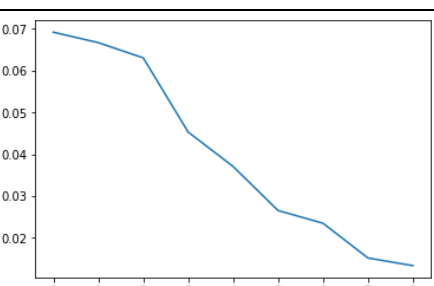
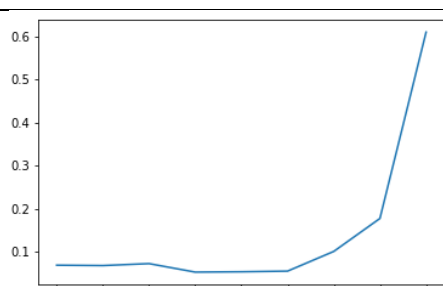
### Discussion:

Comparing the results from a(i) (average testing RMSE = 0.1036), we can directly get the result that the fit is improved except for work\_flow\_1. Given the separation in work\_flow\_1 is very big, the RMSE result turns to be larger than other workflows.

## ii

**Description:** 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. Plot the average train and test RMSE of the trained model against the degree of the polynomial you use.

# Results:

Work Flow ID	Average Train_RMSE	Average Test_RMSE	Best Test parameter
0			Input_set = [2,3,4,5,6,7,8,9,10,11,12,13,14] Best_Poly = 7 Test_RMSE = 0.0114
1			Input_set = [2,3,4,5,6,7,9,10,11,13] Best_Poly = 9 Test_RMSE = 0.0112
2			Input_set = [2,3,4,5,6,7,9,10,12] Best_Poly = 6 Test_RMSE = 0.0247
3			Input_set = [2,3,4,5,6,7,9,10,12] Best_Poly = 5 Test_RMSE = 0.0050
4			Input_set = [2,3,4,5,6,7,9,10,12] Best_Poly = 5 Test_RMSE = 0.0527

### Discussion:

Q: Can you find a threshold on the degree of the fitted polynomial beyond which the generalization error of your model gets worse?

A: In general, the model gets worse when the degree of the fitted polynomial becomes greater than 10. Actually, when the degree of polynomial becomes larger, the training set is already overfitted. Thus, it may cause worse testing RMSE.

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

A: The cross validation decreases the randomness of results, in order to give a more general output.

For instance, when given the degree of polynomial to 18 to fit work\_flow\_4, the 10-fold cross validation results are:

10-fold	1	2	3	4	5	6	7	8	9	10
Train_result	0.0114	0.0110	0.0112	0.0106	0.0113	0.0109	0.0113	0.0107	0.0111	0.0111
Test_result	20.1827	1.7680	0.0734	0.0392	0.0220	0.0276	0.0275	0.1464	2.1170	59.9007

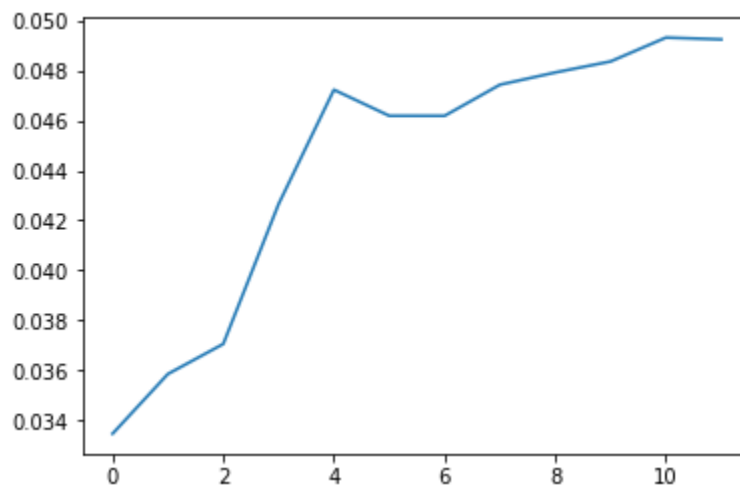
From this example, we can see using cross validation, the effect of extremely large result (i.e. 59.9007) will be decreased, so that we can have better evaluation of certain parameters.

### (e)

**Description:** Use k-nearest neighbor regression and find the best parameter.

**Results:**

Given a set of kNN parameters = [2,3,4,5,6,7,8,9,10,11,12,13], the result of testing\_RMSE vs kNN\_parameters is shown as below:



### Discussion:

From the plot we can tell that:

1. The best kNN parameter is 2.
2. As the number of nearest neighbor grows, the test\_RMSE becomes larger.

### Problem 3

**Description:** 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?

#### Results:

In this project, we totally use 5 models to do regression analysis, which are linear regression model with different regularizers, random forest, neural network, KNN and polynomial regression.

Linear regression model and neural network regression model are mainly used to handle sparse features where one-hot encoding is applied to categorical variables. And random forest model, polynomial regression model and K-nearest neighbor regression can handle categorical variables without having to use on-hot or scalar encodings.

**Best results for each model**

		Test RMSE	Train RMSE	Out of Bag error
Linear regression		0.0883677	0.0883358	/
Random forest		0.0598023	/	0.3345596
Neural network		0.01960	/	/
KNN		0.0335	0.0289	/
Separately fitted	Flow_0	0.0359	0.0358	/
	Flow_1	0.1471	0.1487	
	Flow_2	0.0426	0.0429	
	Flow_3	0.0072	0.0072	
	Flow_4	0.0853	0.0859	
Polynomial	Flow_0	0.0114	0.0091	/
	Flow_1	0.0112	0.0055	
	Flow_2	0.0247	0.0219	
	Flow_3	0.0050	0.0049	
	Flow_4	0.0527	0.0453	

Linear regression model reaches its best result when applying one-hot encoding for variables Day of Week, Backup Start Time – Hour of Day and File name and ridge regularization with  $\alpha = 5$ . Random Forest model of maximum number of features equal to 5 and maximum depth equal to 4 achieves the smallest testing RMSE and Out of Bag error.

For neural network, we one-hot encoded every feature and apply different activities with one hidden layer. The best result happens with ReLU activity and 500 hidden units.

When fit and predict workflows separately, the average testing RMSE improved significantly. After applying polynomial feature fitting, the average testing RMSE also improved. Though the experiment we can tell that with higher degree polynomial parameter, the train set is already overfitted, thus causes worse results in test sets. The best polynomial parameter for different workflow is given in the following table:

Workflow ID	0	1	2	3	4
Degree	7	9	6	5	5

**Conclusion:**

From the results of each regression model, we can conclude that neural network regression model is the best model at handling categorical features and polynomial regression model is the best model at handling sparse features, especially when we predict backup size separately for each workflow ID. Overall, neural network regression model generates the best results.