EE 219 Project 4

Regression Analysis

WenShan Li
UID: 105026914
helenali19@g.ucla.edu

Yunchu Zhang
UID: 80503502
yunchu@g.ucla.edu

Wei Du
UID:005024944
ericdw@g.ucla.edu

Zeyu Zhang
UID: 50530513
zeyu.zhang@cs.ucla.edu

Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, we explore basic regression models on a given dataset, along with basic techniques to handle overfitting; namely cross-validation, and regularization. With cross-validation, we test for overfitting, while with regularization we penalize overly complex models.

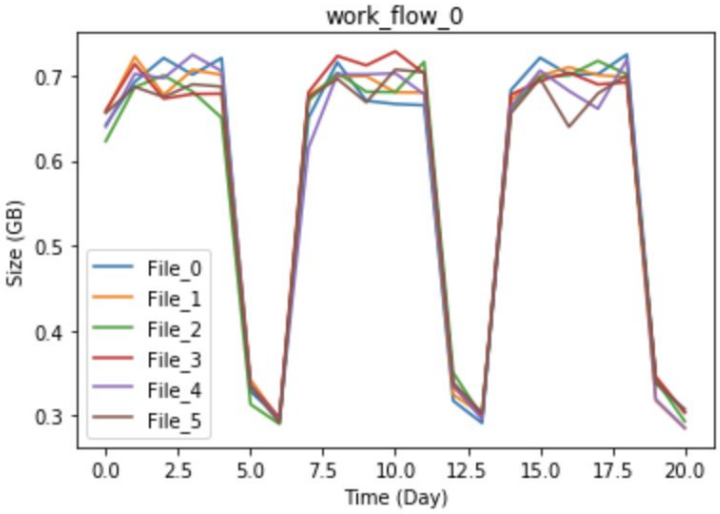
We use a simulated traffic data on a backup system over a network --- Network backup Dataset. The dataset has around 18000 data points with the following columns/variables: 'week' , 'day_of_week' , 'start_time' , 'work_flow' , 'file_name' , 'size' , 'duration'

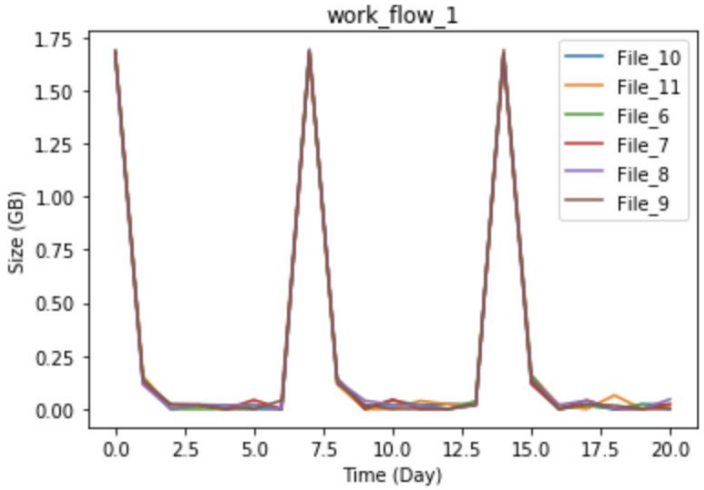
Problem 1

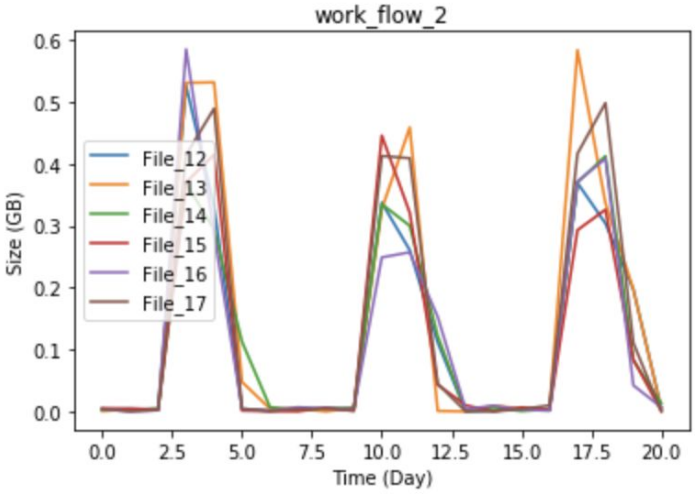
1.1 Load the dataset.

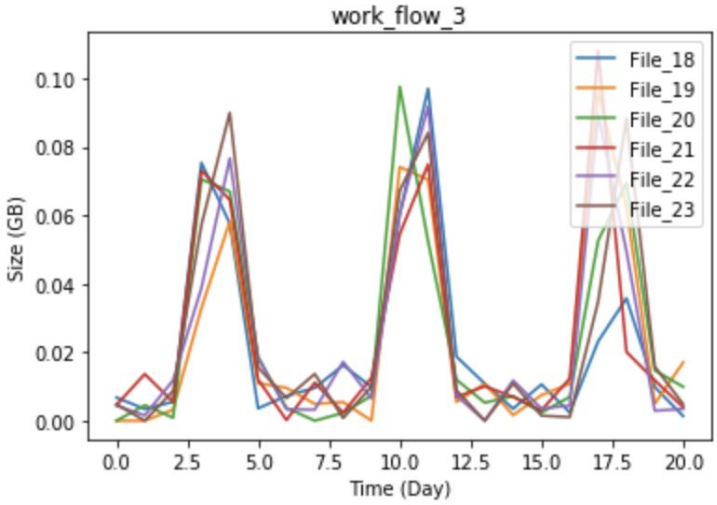
(a) For a twenty-day period (X-axis unit is day number) plot the backup sizes for all workflows (color coded on the Y-axis)

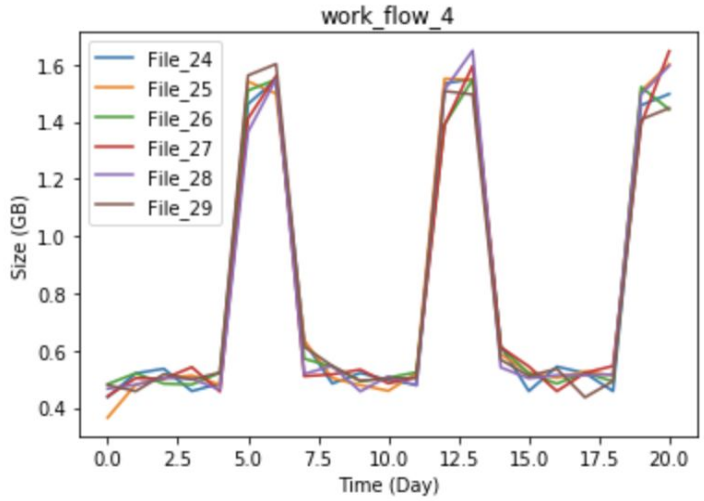
The pics respective to work_flow from 0 to 4 are shown below:



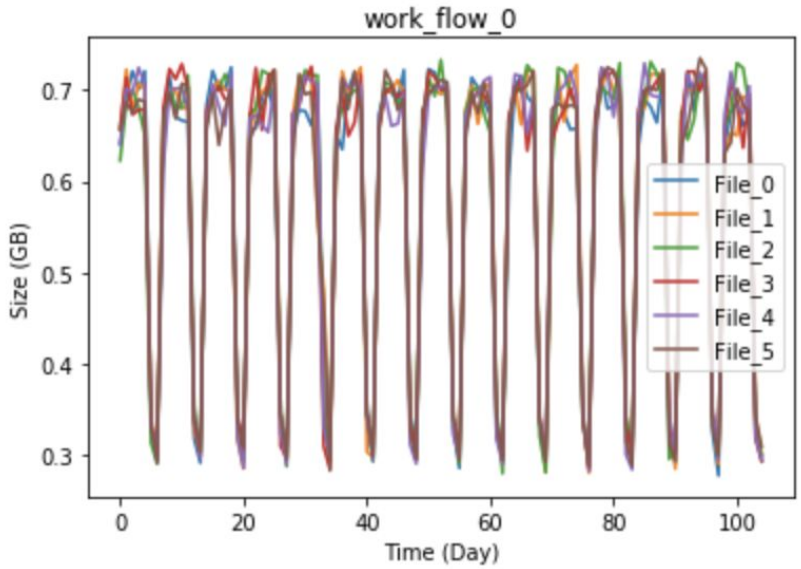

work_flow_1
Size (GB)
Time (Day)
File_10
File_11
File_6
File_7
File_8
File_9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

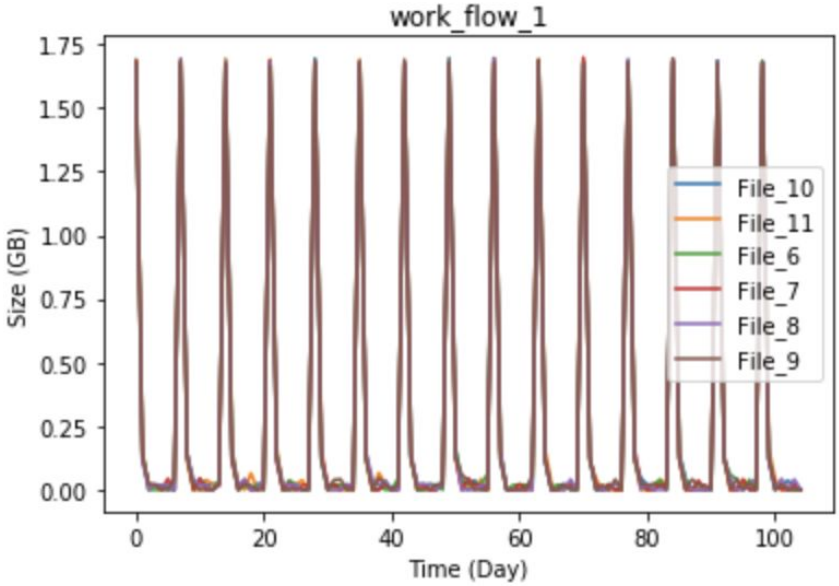

work_flow_2
Size (GB)
Time (Day)
File_12
File_13
File_14
File_15
File_16
File_17
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

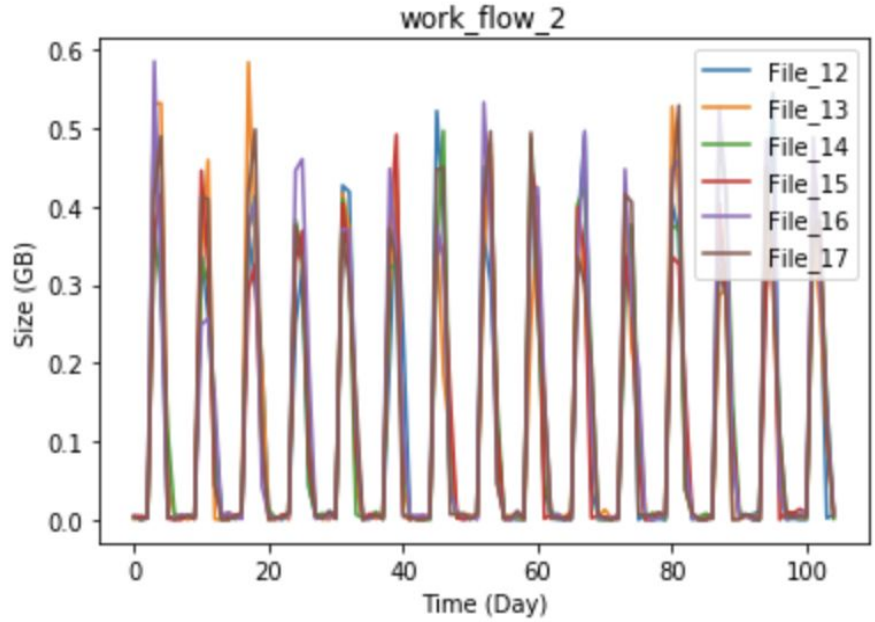


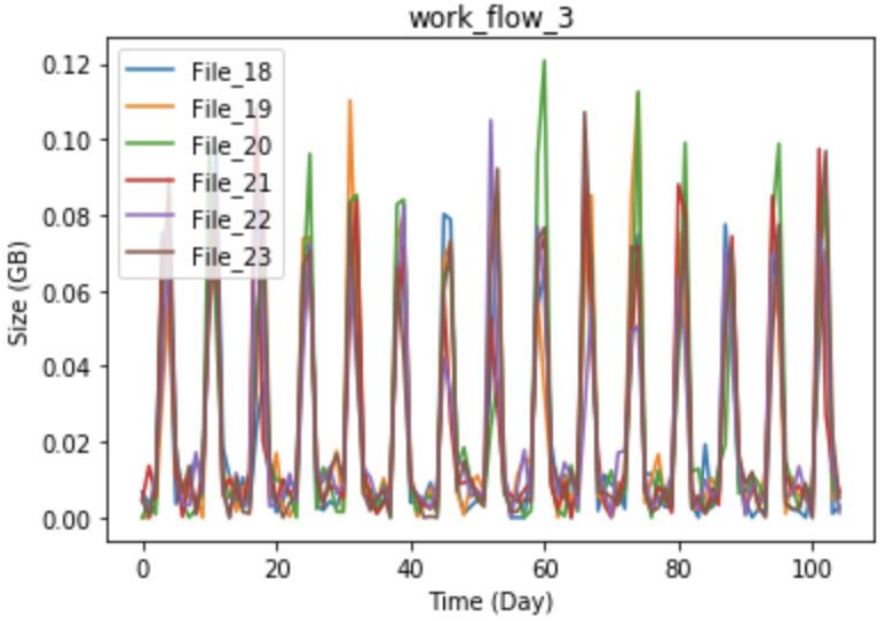


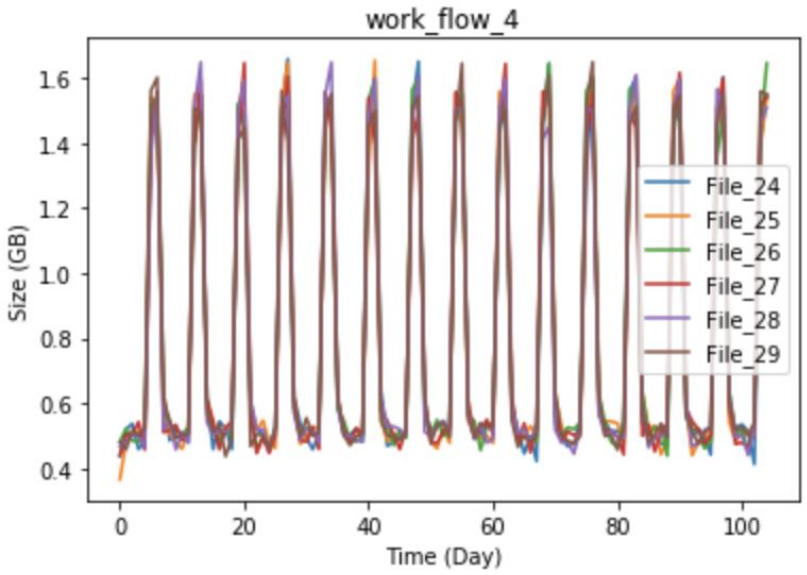
(b) Do the same plot for the first 105-day period.
The pics respective to work_flow from 0 to 4 are shown below:


work_flow_0
Size (GB)
Time (Day)
File_0
File_1
File_2
File_3
File_4
File_5


work_flow_1
Size (GB)
Time (Day)
File_10
File_11
File_6
File_7
File_8
File_9


work_flow_2
File_12
File_13
File_14
File_15
File_16
File_17
Size (GB)
Time (Day)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100


work_flow_3
File_18
File_19
File_20
File_21
File_22
File_23
Size (GB)
Time (Day)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
20
40
60
80
100



1.2 Conclusion

When we draw the graphs: times versus size in different time period, we can find that when we expand the time period from 20 to 105, every 20 time period, the pattern will repeat for one time. Hence, for the 105 days period, the repeating pattern for every work_flow occurs approximately 5 time.

Problem 2

We use all attributes, except Backup time, as candidate features for the prediction of backup size. We will try different feature sets, as well as different encoding schemes for each feature and in combination. Scalar encoding and one hot encoding to encode the attributes. And we have 32 possible combinations of encoding these 5 features.

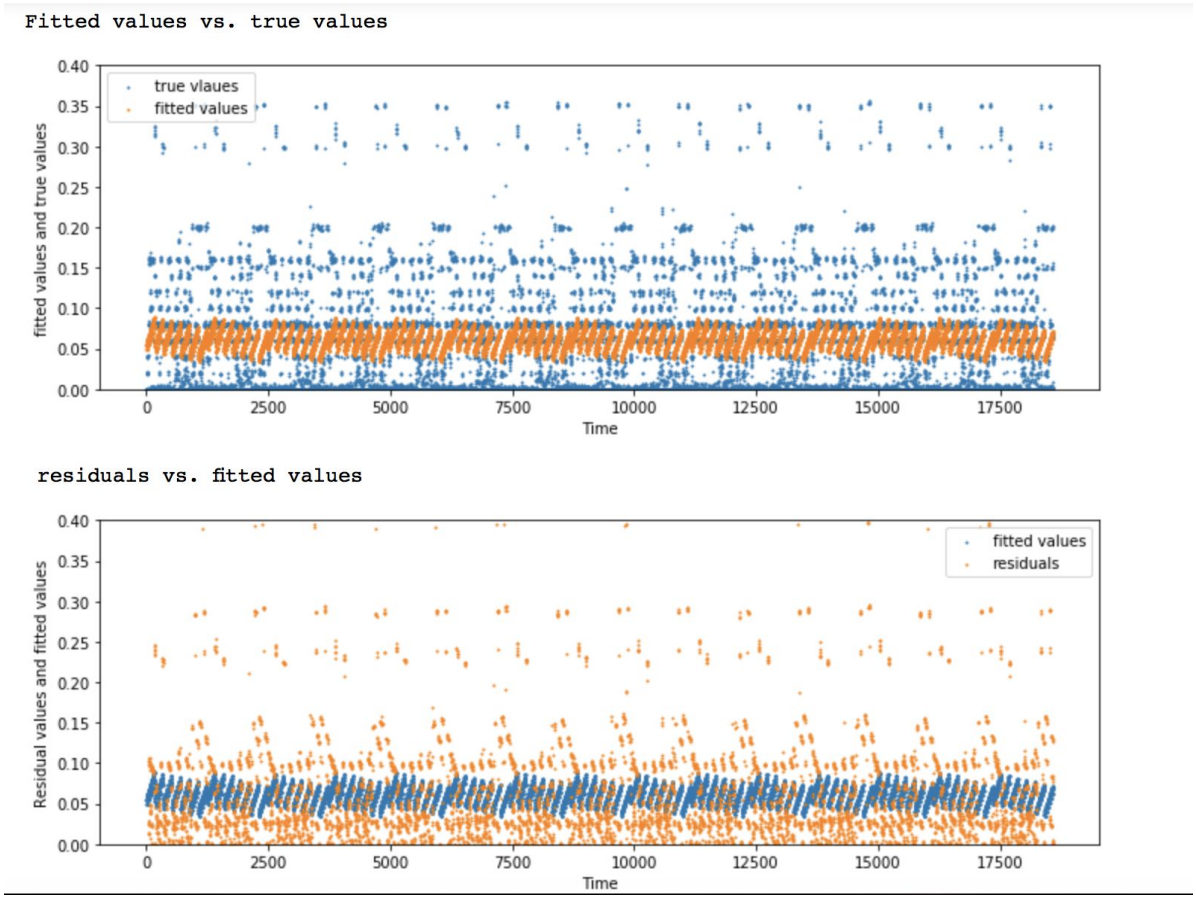
For the next part , we will report training and test RMSE from 10-fold cross validation as basic evaluation of the performance. We will use training RMSE and test RMSE to evaluate . In addition, we will plot fitted values against true values scattered over the number of data points and plot residuals versus fitted values scattered over the number of data points using the whole dataset for each model with the best parameters we have found.

2.1 Linear Regression Model

Fit a linear regression model. We use ordinary least square as the penalty function.

2.1.1 Directly method

First convert each categorical feature into one dimensional numerical values using scalar encoding , and then directly use them to fit a basic linear regression model.

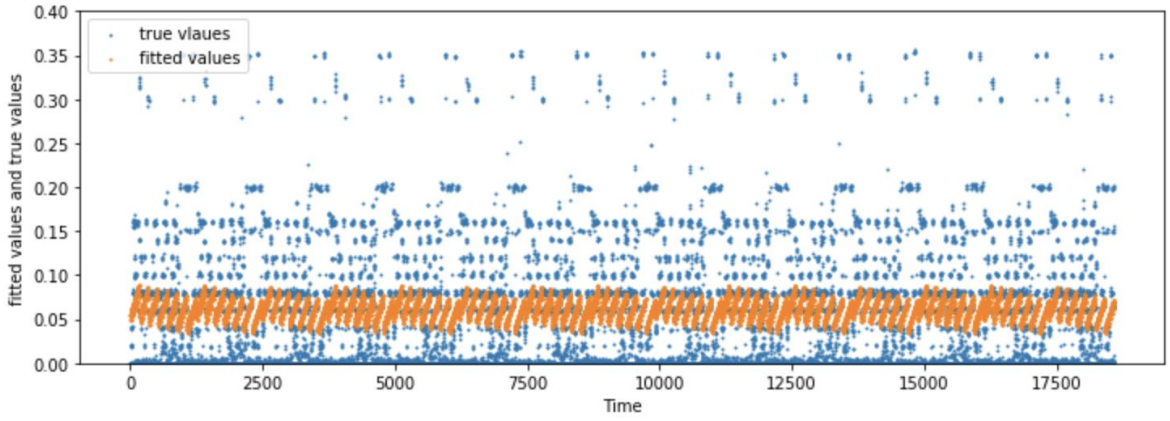


Test RMSE with cross validation: 0.1036758
Train RMSE with cross validation: 0.1035854

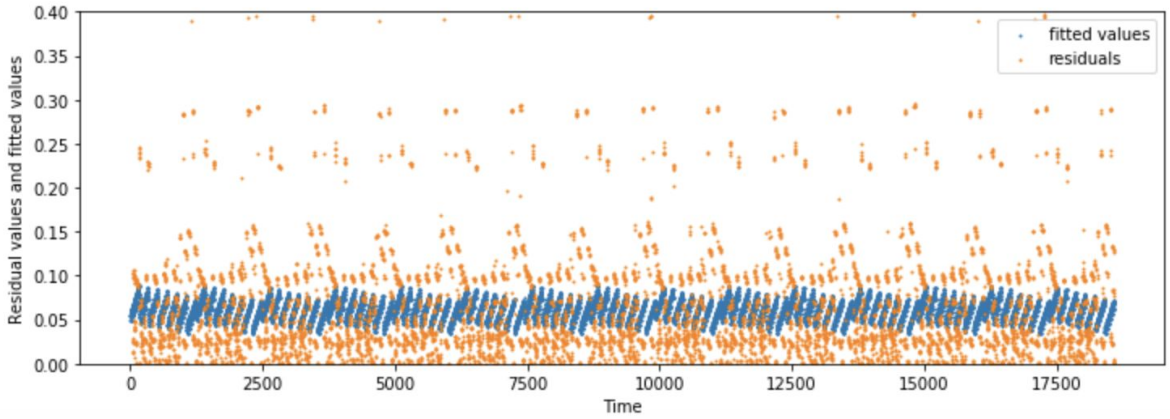
2.1.2 Data Procession

Standardize all these numerical features, then fit and test the model.

Fitted values vs. true values



residuals vs. fitted values



Test RMSE with cross validation: 0.1036758
Train RMSE with cross validation: 0.1035854

From the RMSE and the plotted images, we find that that the standardization doesn't make the performance better in this case.

2.1.3 Feature Selection

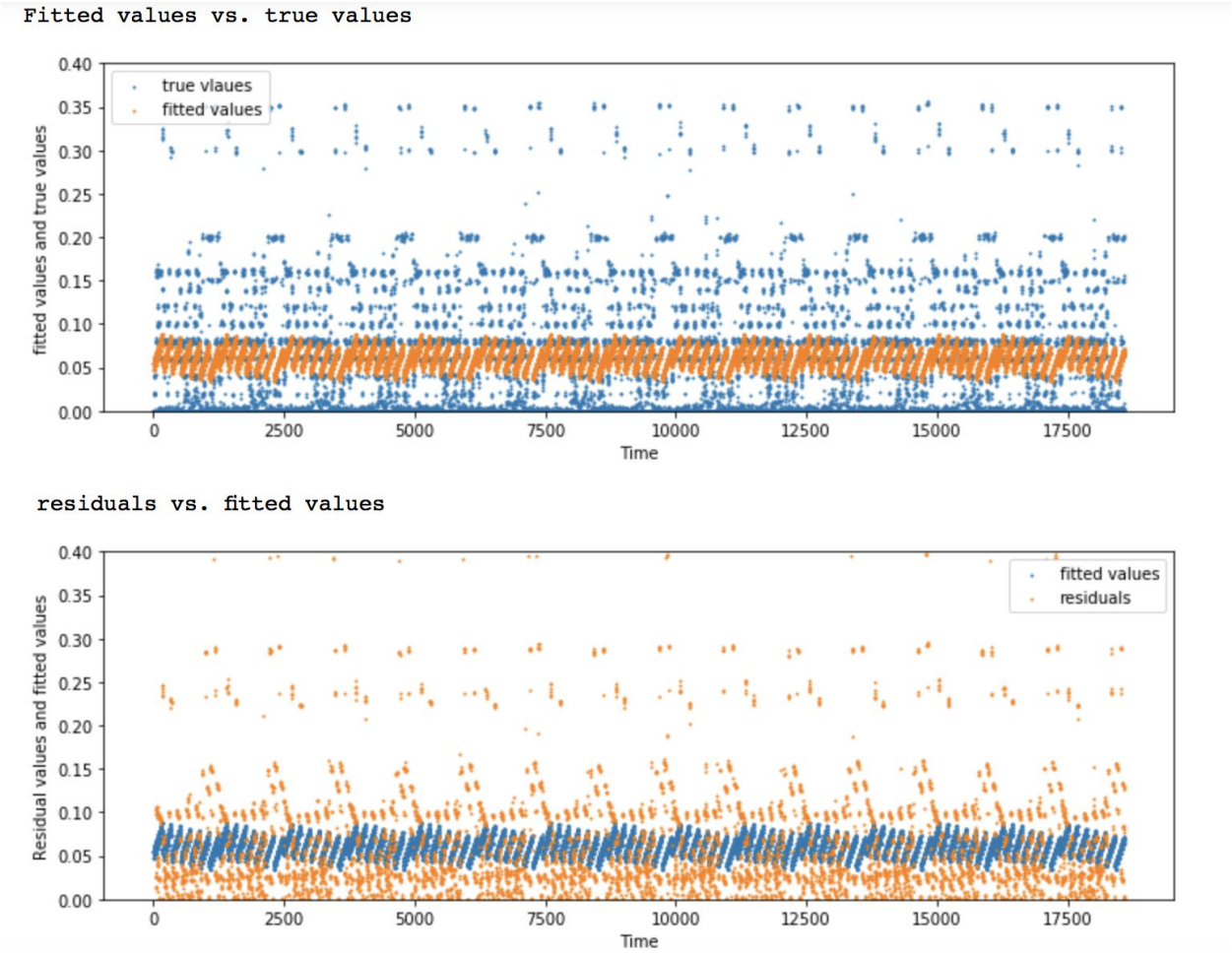
Use f-regression and mutual information regression measure to select three most important variables respectively.

2.1.3.1 f-regression selection

First we will use f-regression to select the three most important features. The importance we get is shown below:

[8.45006285e-03 3.88163795e+01 1.50740935e+02 2.61386654e+01
2.53200943e+01]

From f_regression, three most important variables are 'Start Time', 'Day of Week', and 'Work-Flow'.Then we use these three most important variables to train a new linear model.



Test RMSE with cross validation: 0.1036707
Train RMSE with cross validation: 0.1035857

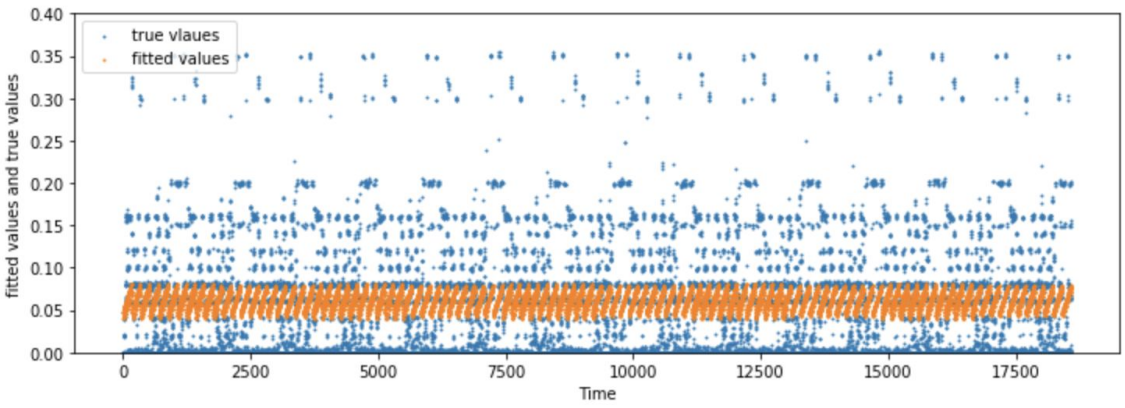
2.1.3.2 mutual information regression

Then we will use mutual information regression to select the three most important features. The importance we get is shown below:

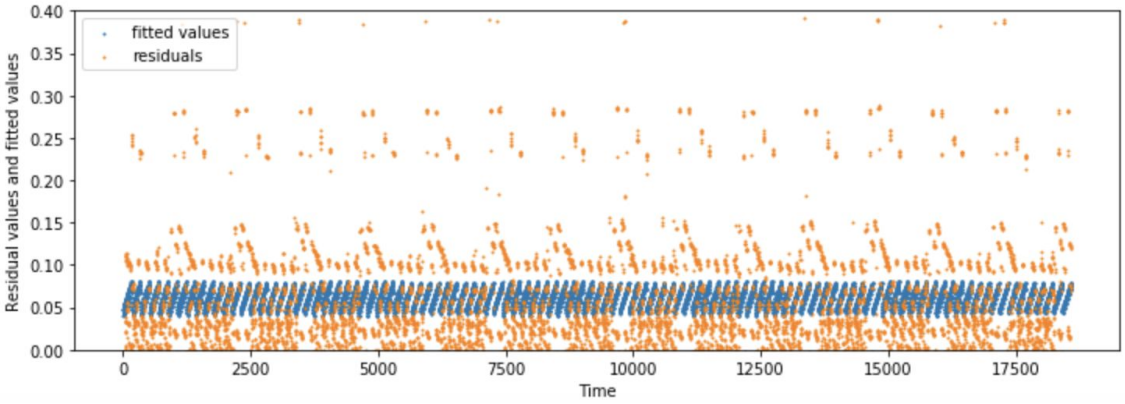
[0.00270929 0.22328277 0.24224216 0.71009406 0.43137242]

From mutual information regression, three most important variables are 'Start Time', 'File Name', and 'Work-Flow'.Then we use these three most important variables to train a new linear model.

Fitted values vs. true values



residuals vs. fitted values



Test RMSE with cross validation: 0.1037723
Train RMSE with cross validation: 0.1036945

2.1.3.3 Conclusion

Even though we use different methods to find the three important features respectively, the performance of this model is still not improve. The RMSE is still around 0.103.

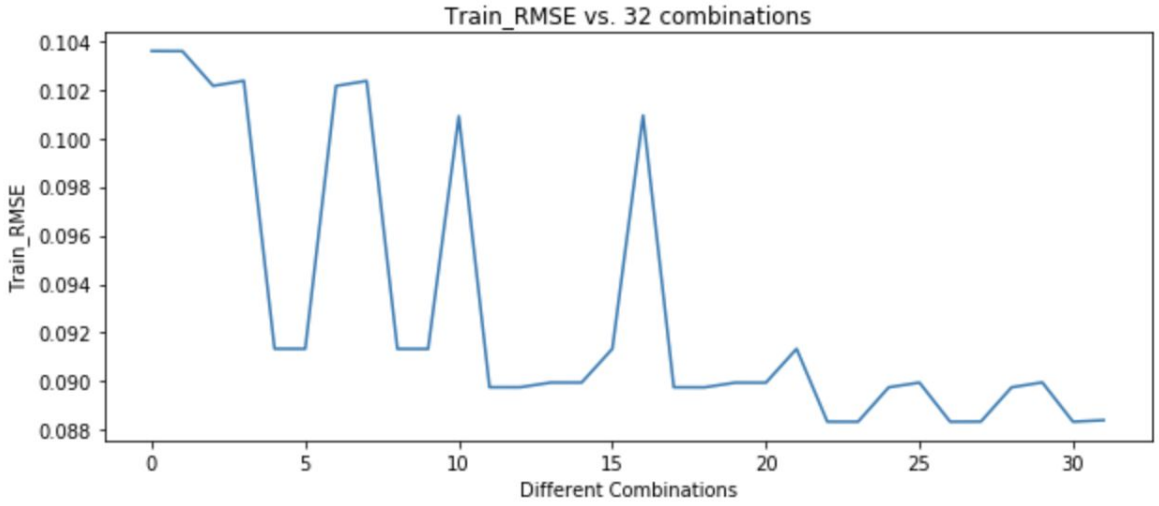
This shows that the encoding method in this way is not optimal, and there will be a better encoding way.

2.1.4 Feature Encoding

There are 32 possible combinations of encoding the five categorical variables. And we will try the 32 combinations next and find the best performance.

We can get the train_RMSE vs 32 combinations in Jupyter.

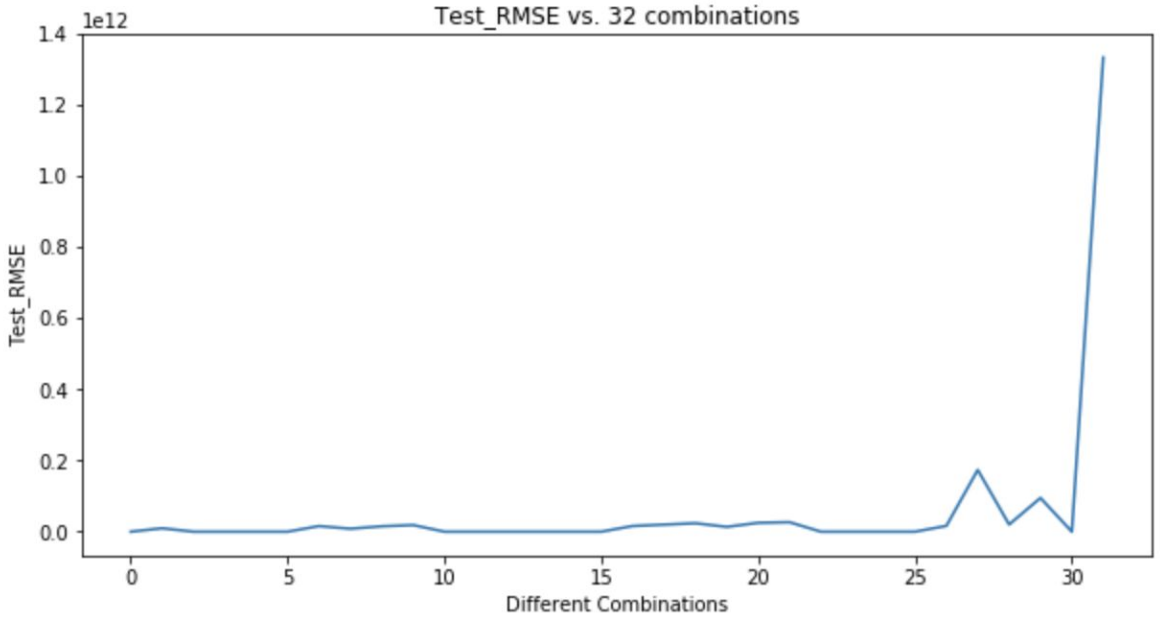
We plot these Train RMSE together and get the curve shown below:



We get the min Train_RMSE is 0.088338 and the best combos is (1,2,4), which means that the best combination is using One Hot encoder for 'Day of Week', 'Start Time ','File Name' and use scalar encoding for 'Week ', 'Work-Flow'.

Also, we can get the Test_RMSE vs 32 combinations(in Jupyter)

We plot these Test RMSE together and get the curve shown below:



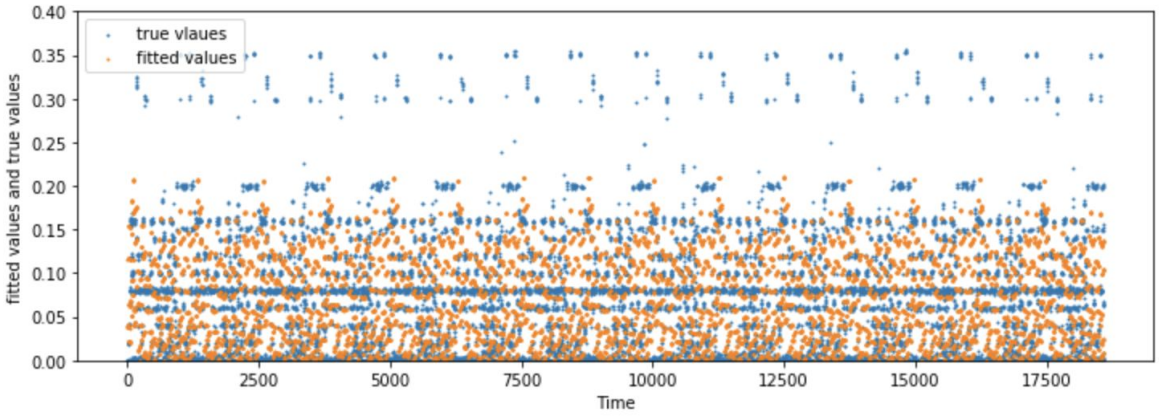
We get the min Train_RMSE is 0.088338 and the best combos is (1,2,4), the result of which is the same as Train_RMSE. And we can find that the curve increases sharply when the different combinations number over No.30. The reason is that using one hot encoder to all these features will cause overfitting.

Then we will use the best combination: that is using One Hot encoder for 'Day of Week', 'Start Time ','File Name' and use scalar encoding for 'Week ', 'Work-Flow' to fit and test the model.

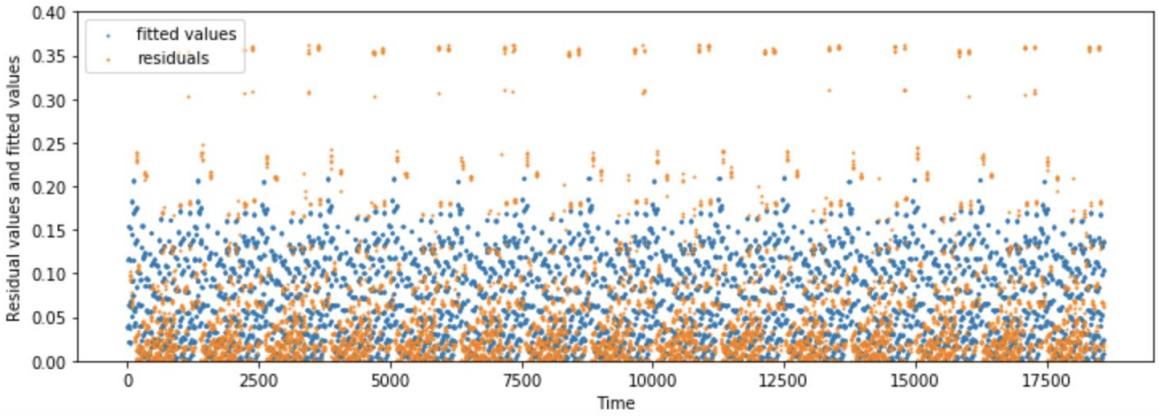
The intuitive explanation of this is that the scalar of week and work_flow contains the important information. When we use one hot encoder, these information will be lost.

The result of the combination is shown below:

Fitted values vs. true values



residuals vs. fitted values



Test RMSE with cross validation: 0.0885063
Train RMSE with cross validation: 0.0883382

We can observe from these pics and datas that find a better encoding way can improve our performance in this model better, even though we don't use standardize in this case.

2.1.5 Controlling ill-conditioning and over-fitting

We found obvious increases in test RMSE compared to training RMSE in some combinations. Because different features have different attributes. Some features contains numerical information so we use scalar encoding. Others are just categories and one hot encoding is better. And using one hot encoder to all these features will cause overfitting.

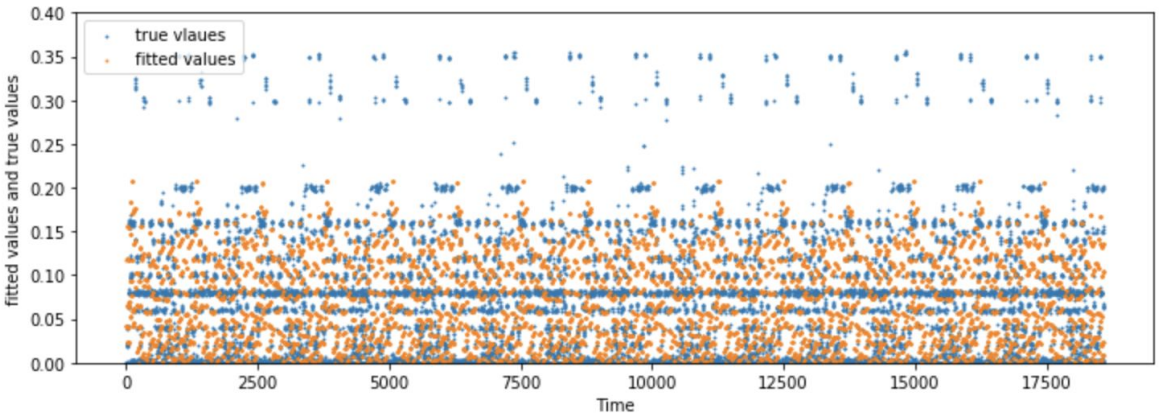
In this section, we will implement different regularizations, including Ridge Regression, Lasso Regression and Elastic Net Regression. For any choice of the hyper-parameter(s) we will find one of the 32 models with the lowest Test-RMSE.

2.1.5.1 Ridge Regularizer

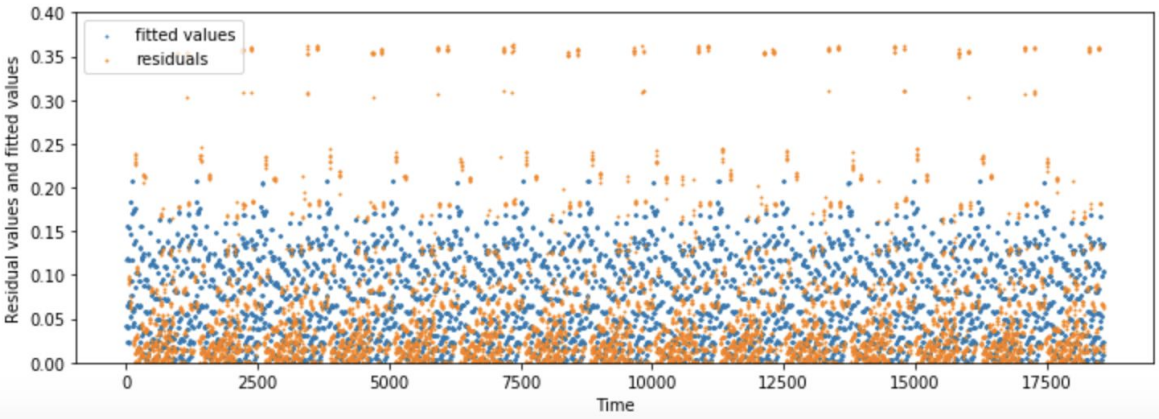
For Ridge Regularizer, after using different alpha values and make comparison of different results find when alpha=10.5, apply one hot encoding for ' Week of day', 'Start Time ','Work-Flow' and scalar encoding for 'Week #', 'File Name', we can get the lowest Test-RMSE value, which equals to 0.0885039932839.

The Test_RMSE values vs. 32 combinations results shown in Jupyter and plot the images as below:

Fitted values vs. true values



residuals vs. fitted values

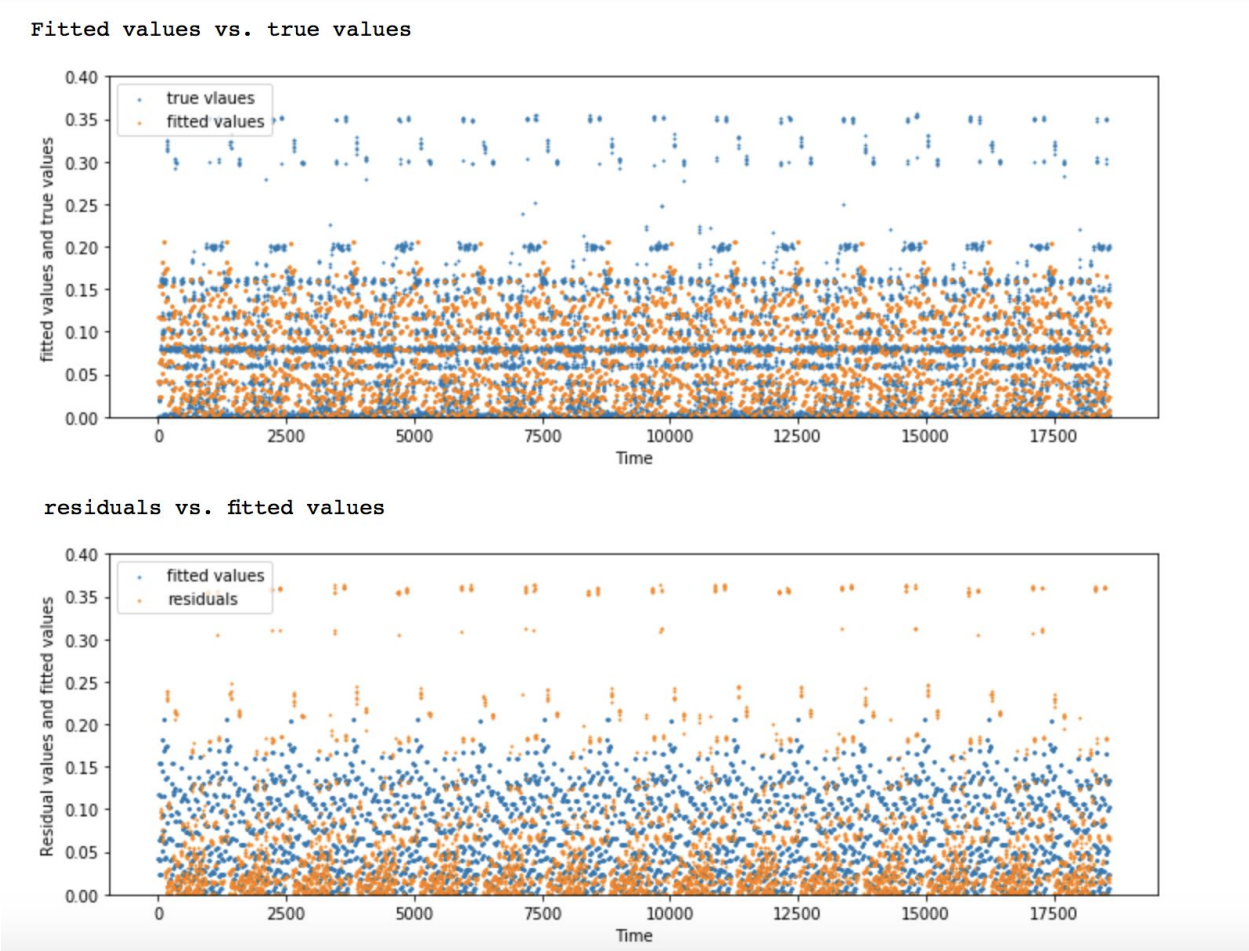


Test RMSE with cross validation: 0.0885040
Train RMSE with cross validation: 0.0883377

2.1.5.2 Lasso Regularizer

For Lasso Regularizer, after using different alpha values and make comparison of different results find when alpha=0.0001, apply one hot encoding for 'Week #','Day of Week', 'Start Time', 'Work-Flow' and 'File Name', we can get the lowest Test-RMSE value, which equals to 0.0885082492502

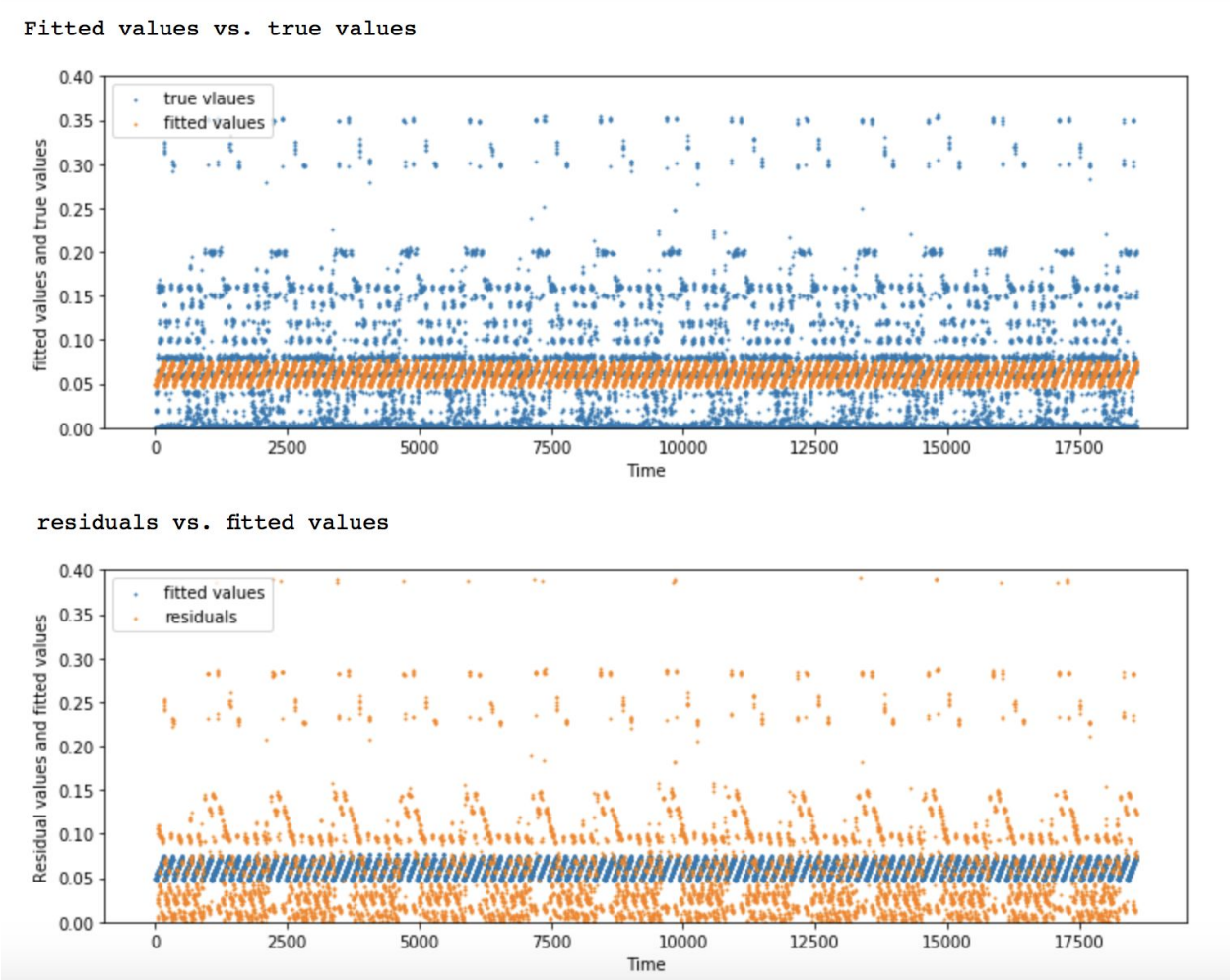
The Test_RMSE values vs. 32 combinations results shown in Jupyter and plot the images as below:



Test RMSE with cross validation: 0.0885082
Train RMSE with cross validation: 0.0883430

2.1.5.3 Elastic Net Regularizer

In order to find the best parameters, we set alphas=[1,2,5,10,15], and l1_ratios=[0.01,0.02,0.05,0.1, 0.5,0.9] to train and test the model. From the results in Jupyter, when choose alpha and L1 ratio equals to [1, 0.01] and One hot encoding combination is (3, 4) ,we can get the minimum Test_RMSE :0.103080.



Test RMSE with cross validation: 0.1030802
Train RMSE with cross validation: 0.1030681

2.1.5.4 Conclusion

Compare the values of the estimated coefficients for these regularized good models, with the un-regularized best model, we find that the Ridge Regularizer and Lasso Regularizer improved the performance a little(or can be omitted). And the Elastic Net Regularizer in this parameters not improve the performance. From the RMSE results above, we get that we can use Ridge and Lasso Regularizer to prevent overfitting, however, the performance will not improve.

2.2 Random Forest Regression

During the training process, we use one feature to minimize a chosen measure of impurity in a branching decision. For classification, we use Gini or entropy. In the random forest regression, since we use bootstrapping, it's easier and faster to evaluate the generalization ability. We will use RMSE and out of bag error to estimate our model.

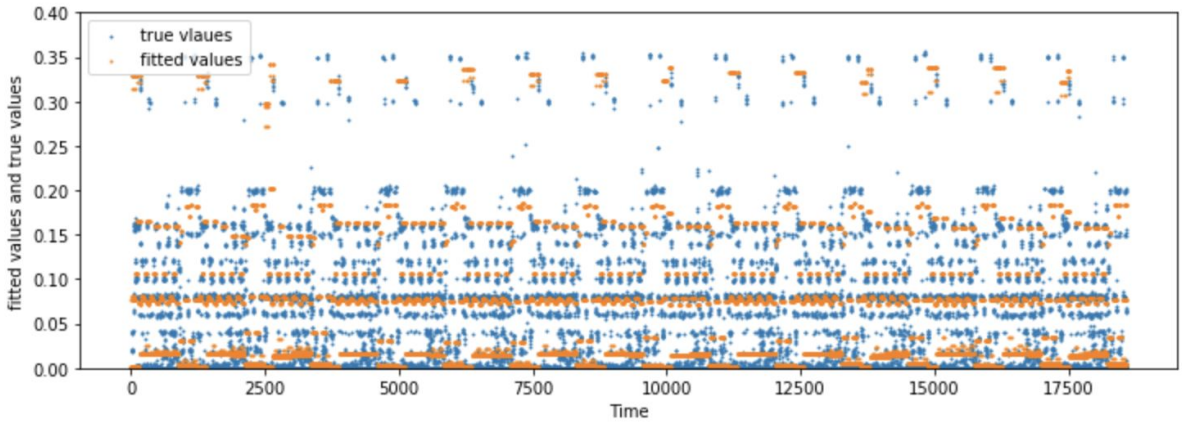
2.2.1 Initial Model

We first set the parameters of our model with the following initial values:

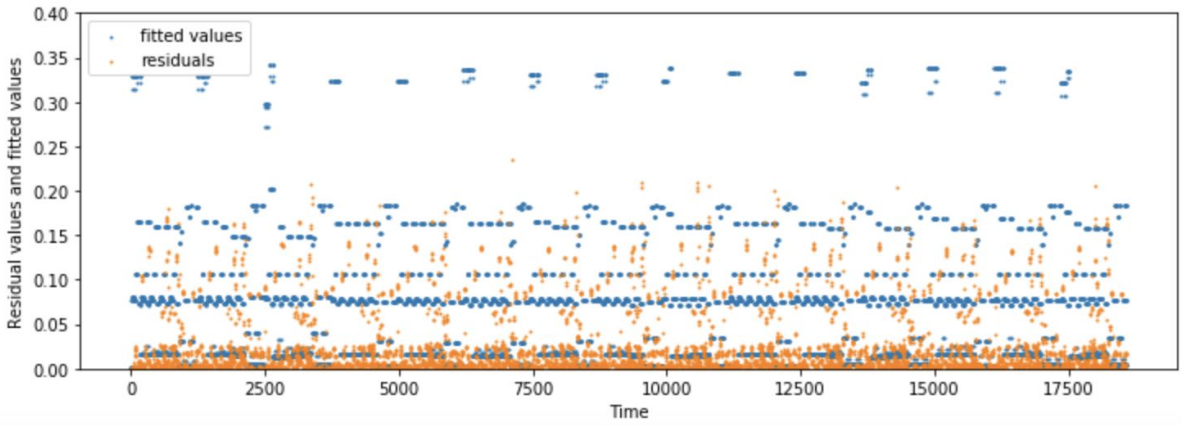
- Number of trees: 20
- Depth of each tree: 4
- Bootstrap: True
- Maximum number of features: 5

And then train and test the model and get the results shown below:

```
Fitted values vs. true values
```




```
residuals vs. fitted values
```

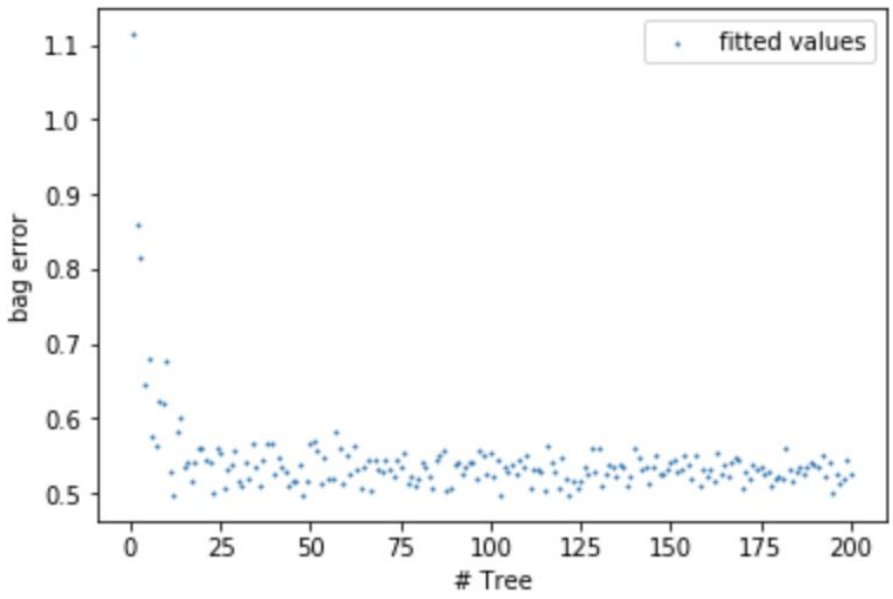


Test RMSE with cross validation: 0.0605150
Train RMSE with cross validation: 0.0603017
Out Of Bag error: 0.335498030454

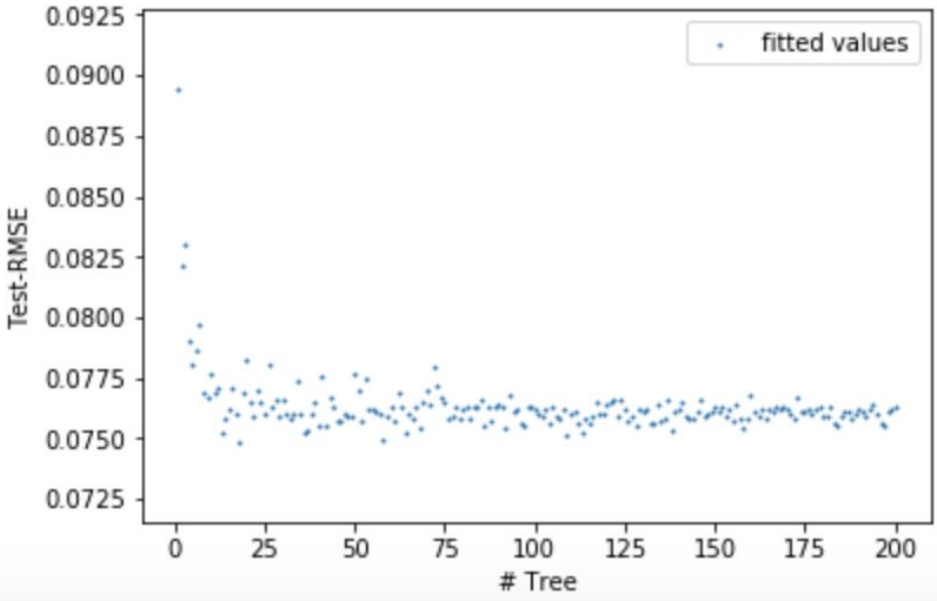
2.2.2 Change Parameter

Sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5, plot out of bag error(y axis) against number of trees(x axis), and average Test-RMSE(y axis) against number of trees(x axis).

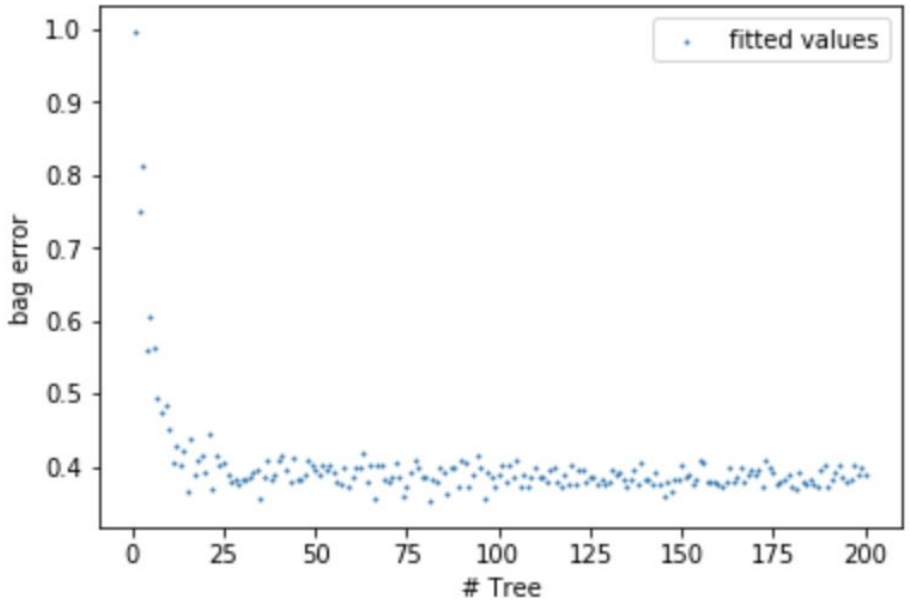
out of bag error against number of trees with 1 features



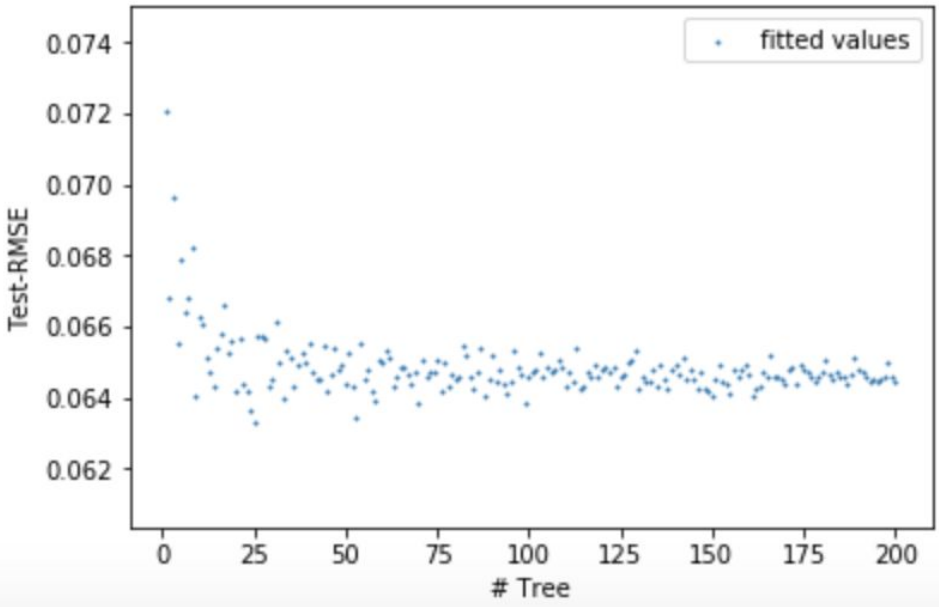
average Test-RMSE against number of trees with 1 features



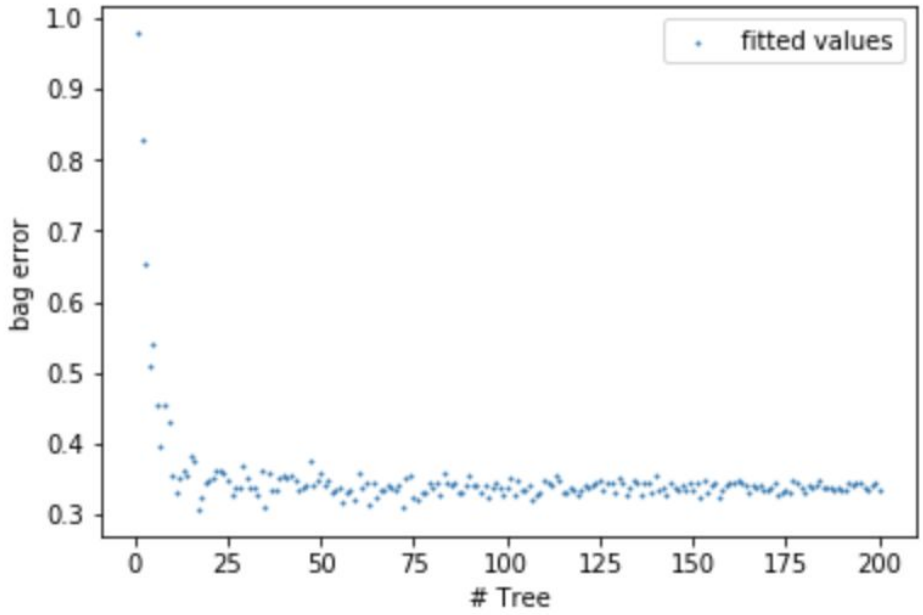
out of bag error against number of trees with 2 features



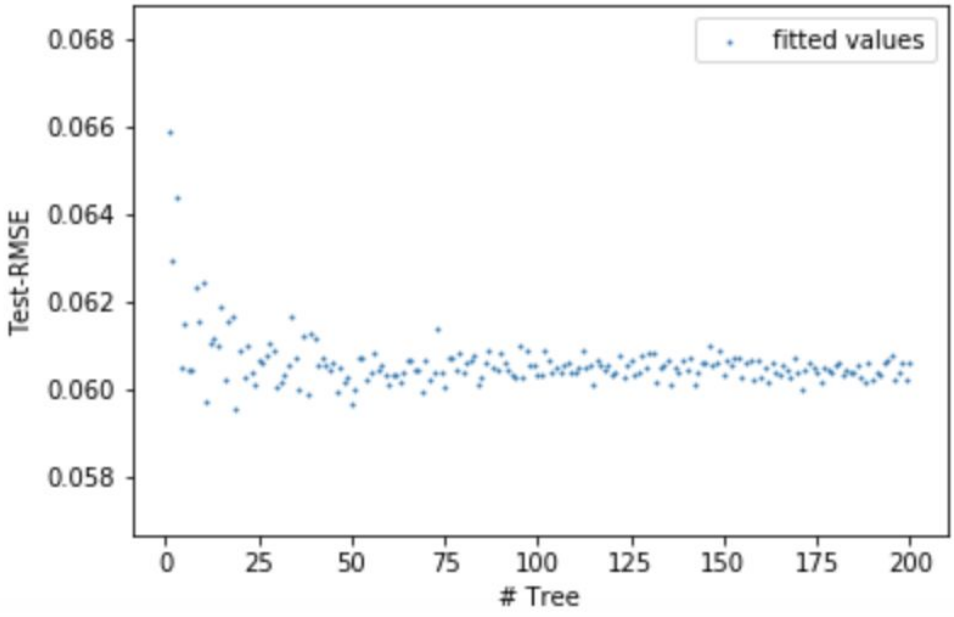
average Test-RMSE against number of trees with 2 features



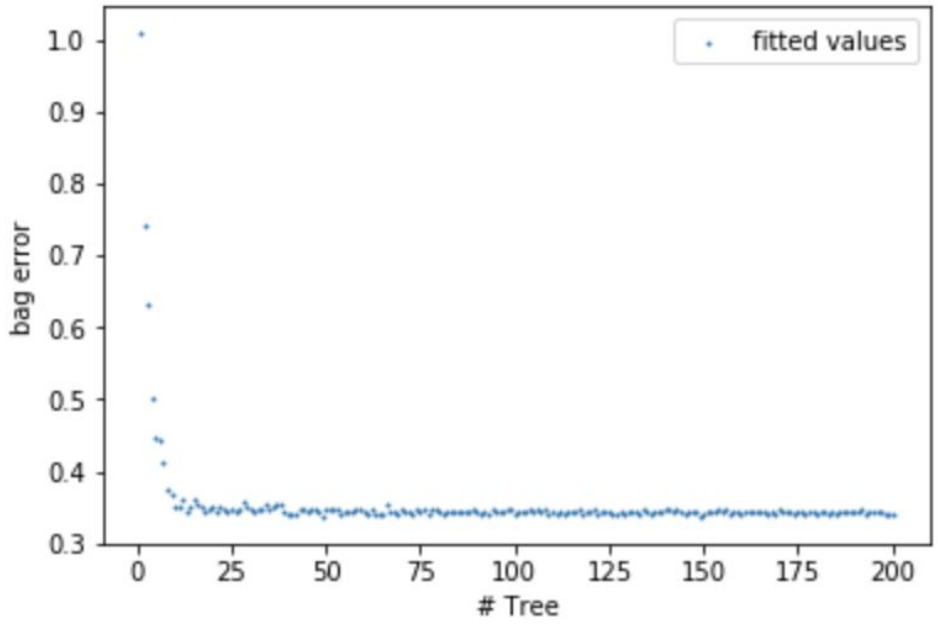
out of bag error against number of trees with 3 features



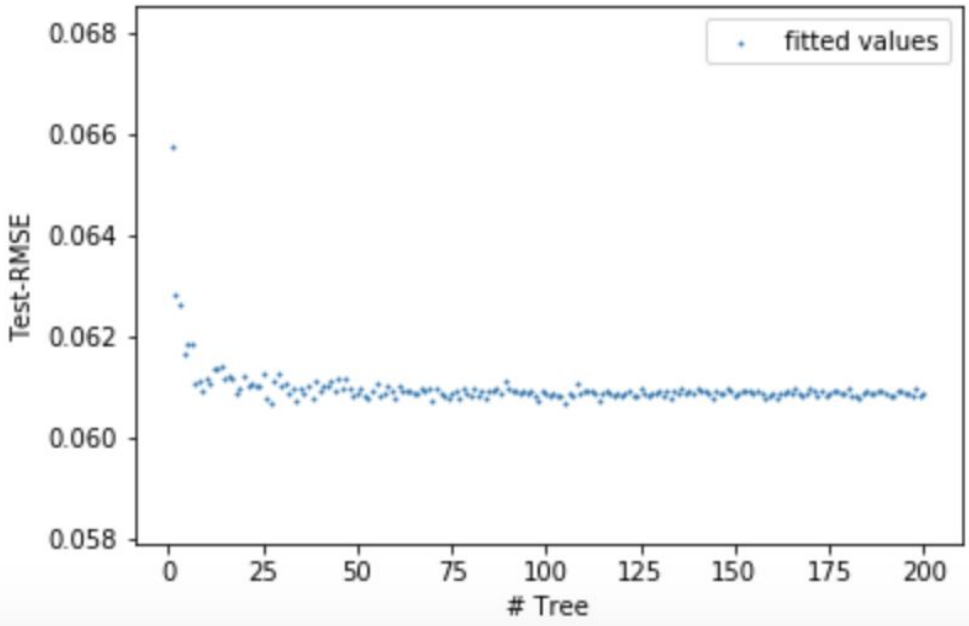
average Test-RMSE against number of trees with 3 features



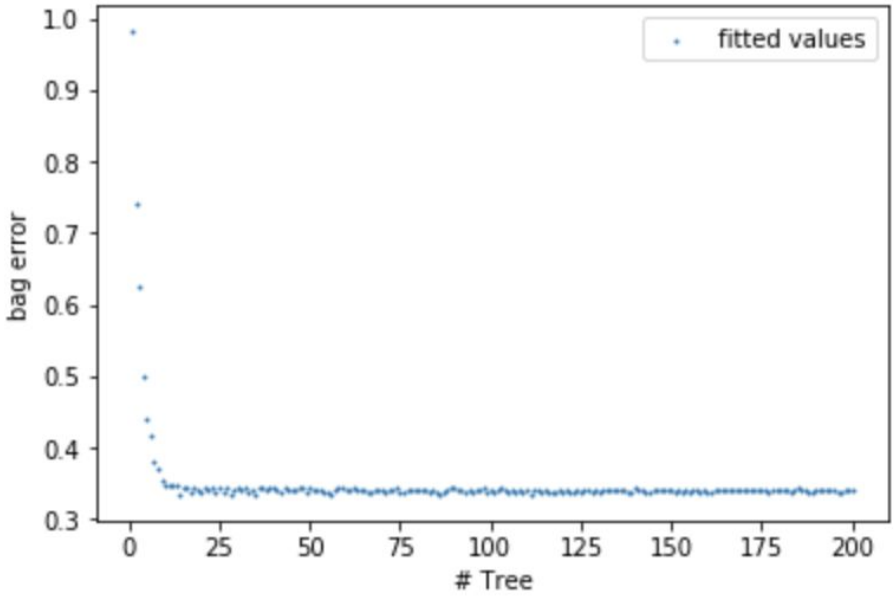
out of bag error against number of trees with 4 features



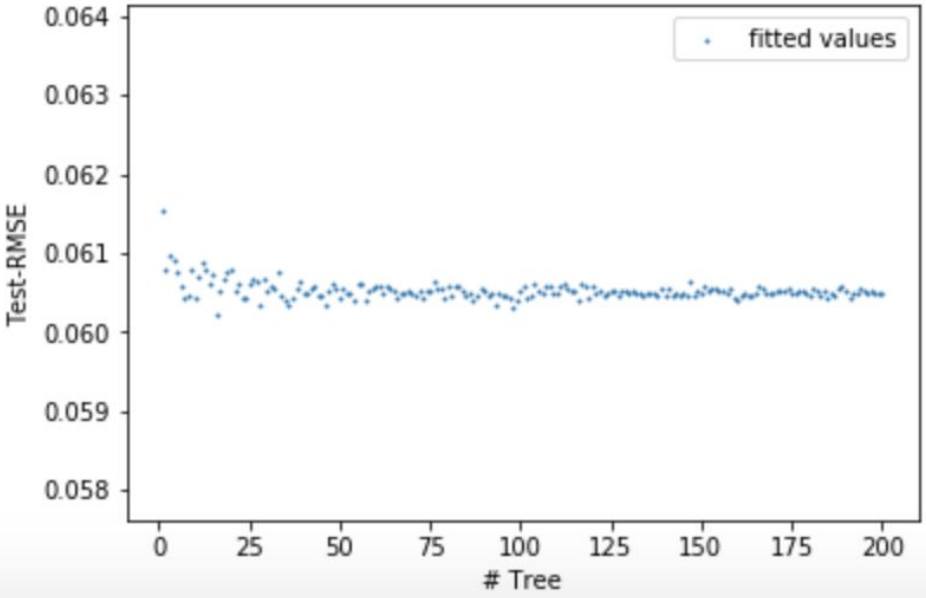
average Test-RMSE against number of trees with 4 features



out of bag error against number of trees with 5 features



average Test-RMSE against number of trees with 5 features



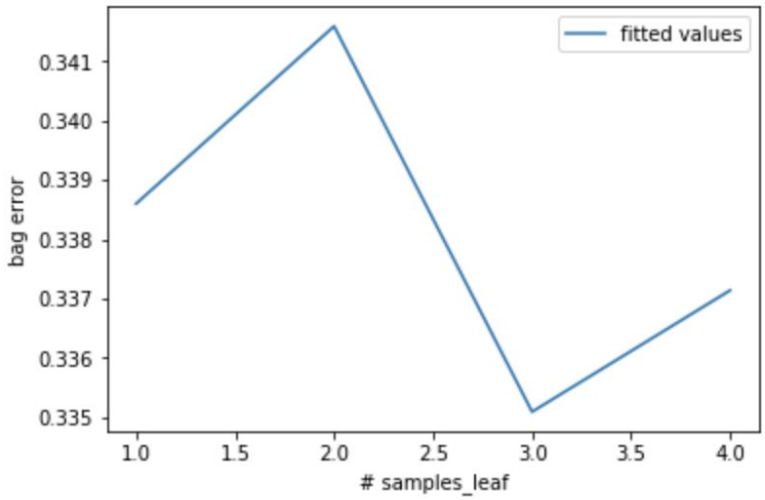
We find that , with the increase of the # of features, the Test-RMSE and the bag error lower and lower. And with the increase of the # of trees, the Test-RMSE and the bag error trend to stable. Nearly at 150 number of trees, we can have a stable error and RMSE.

2.2.3 Pick min sample leaf

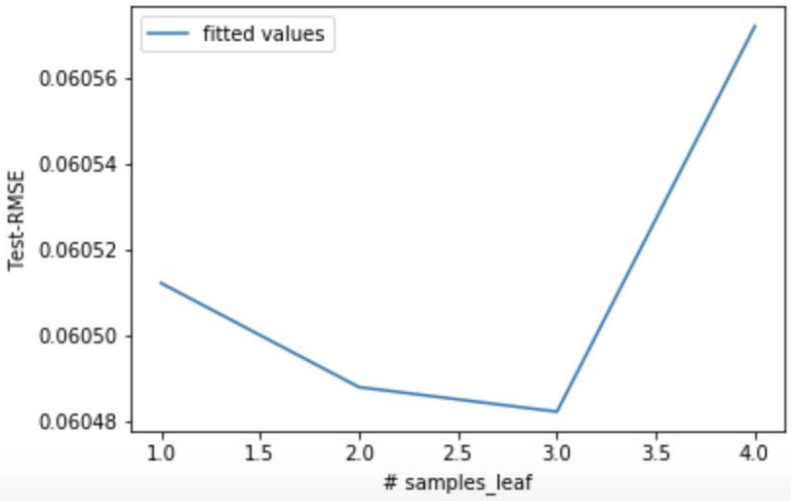
From the materials on the website and other information we find that the min sample leaf has great impact on decision tree, so we loop the min sample leaf from 1 to 4 and get the out of bag error and RMSE curve shown below.

And we find that when the min sample leaf set to 3, we can get better performance.

```
out of bag error against number of trees with min_samples_leaf
```




```
average Test-RMSE against number of trees with min_samples_leaf
```

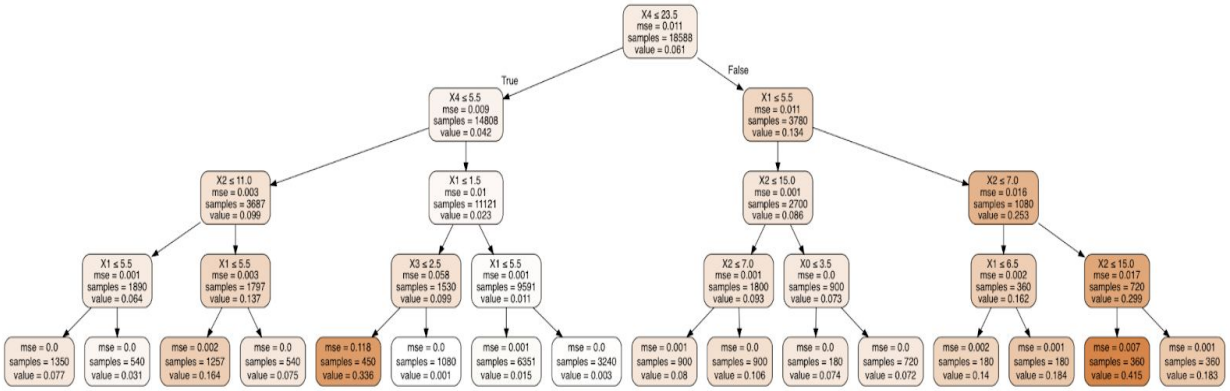


2.2.4 Best Feature

According to the result shown in Jupyter, we get that the importances of each feature is: # of week: 6.32004878e-06, 'Day of Week':2.72400153e-01, 'Backup Start Time - Hour of Day': 3.79095883e-01, 'Work-Flow-ID':2.17991111e-01 , 'File Name':3.60506532e-01

So the top 3 important features are: Start Time, File Name, Day of Week

2.2.5 Visualize Decision Tree



According to the graph above,the root node is x4<=23.5 and the top 3 important features for max_depth=4 is x3,x1,x4. The result got by attribute feature_importances_ is x4,x1,x3. They are almost the same.

2.3 Neural Network Regression

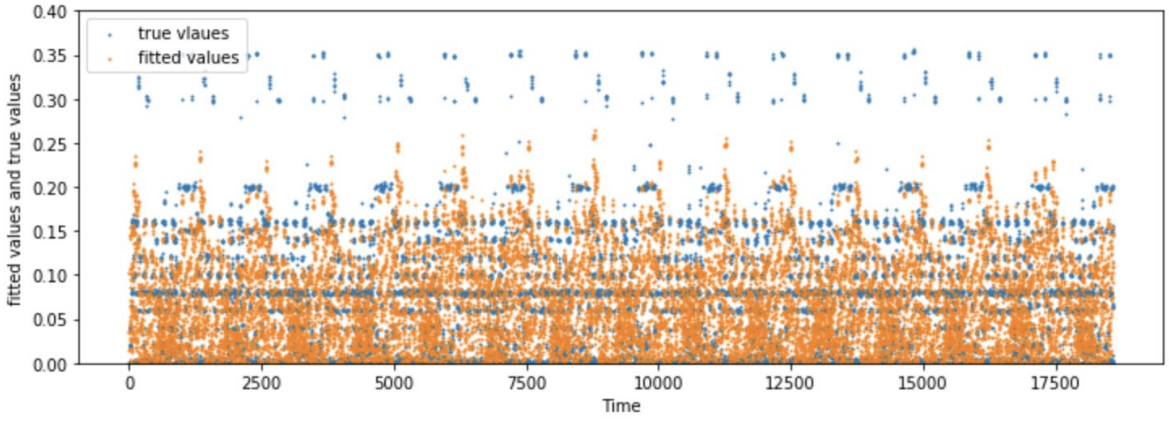
Now we will use a neural network regression model (one hidden layer) with all features one-hot encoded. In this section, we will implement three activate function: relu, logistic and tanh and different hidden units to test the model. We will then plot Test-RMSE as a function of the number of hidden units for different activity functions.

2.3.1 Relu

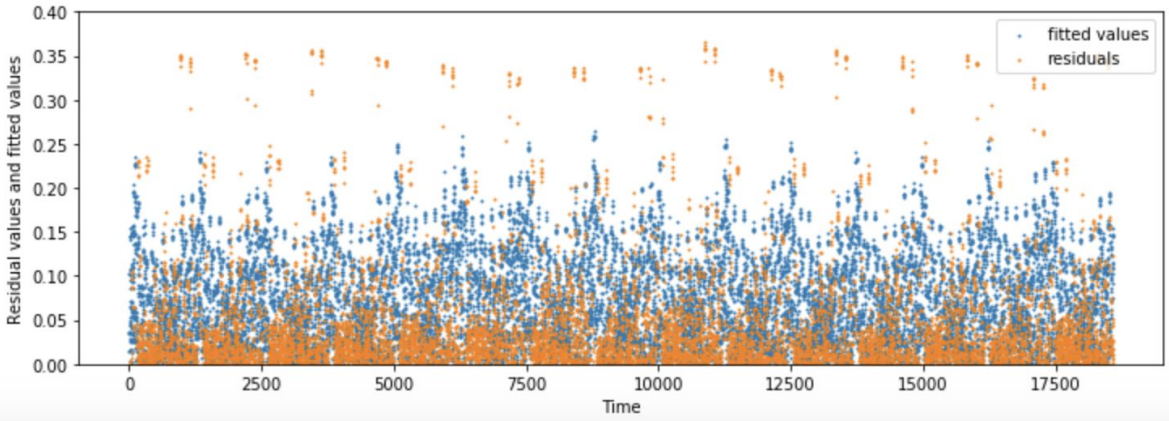
We will set the activate function as Relu, and the hidden units from 5 to 105. Step is 10. And we get the result shown below:

hidden_units:5
Test RMSE with cross validation: 0.0866618
Train RMSE with cross validation: 0.0857770

Fitted values vs. true values

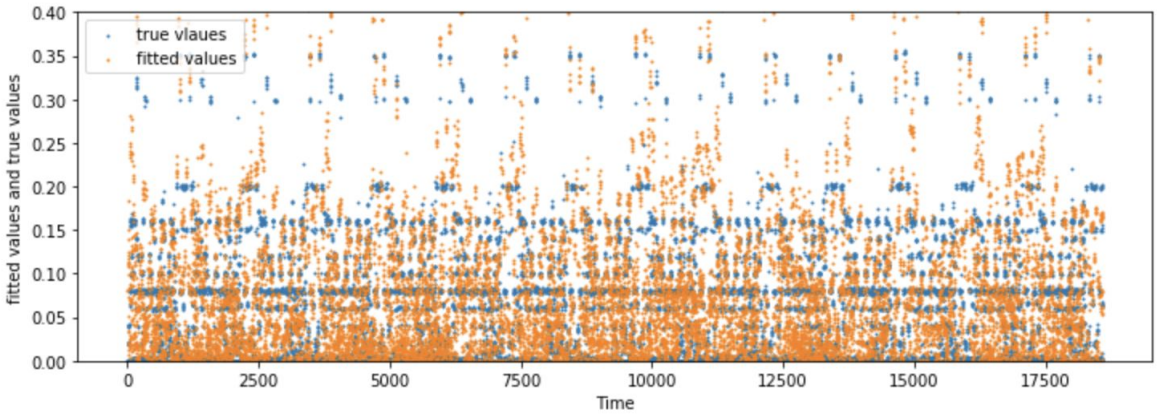


residuals vs. fitted values

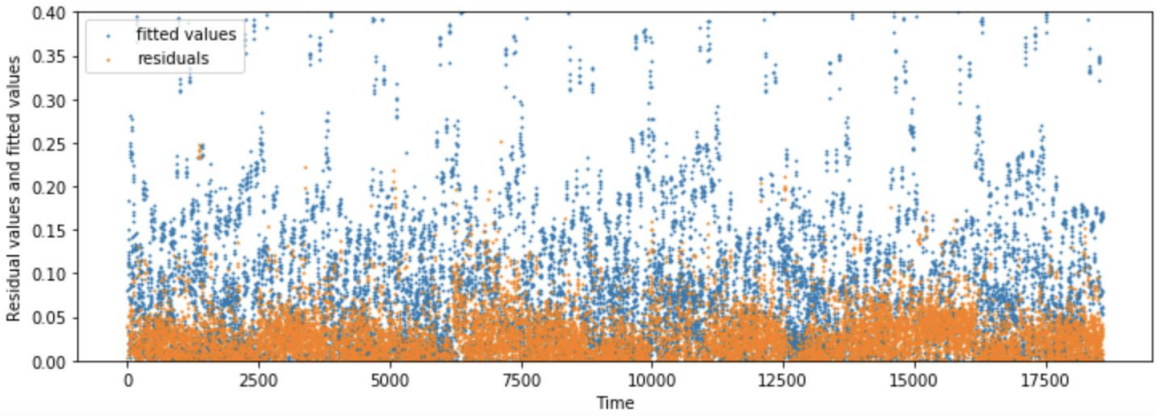


hidden_units:15
Test RMSE with cross validation: 0.0433685
Train RMSE with cross validation: 0.0294423

Fitted values vs. true values

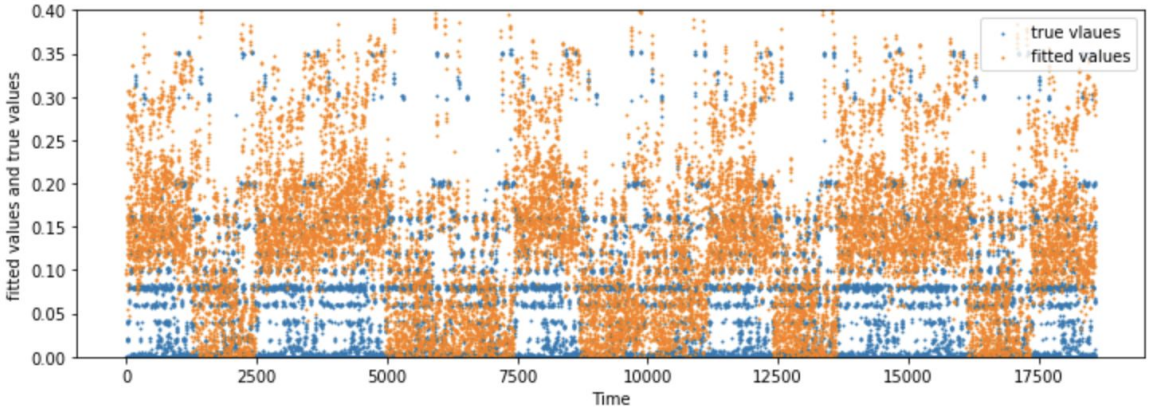


residuals vs. fitted values

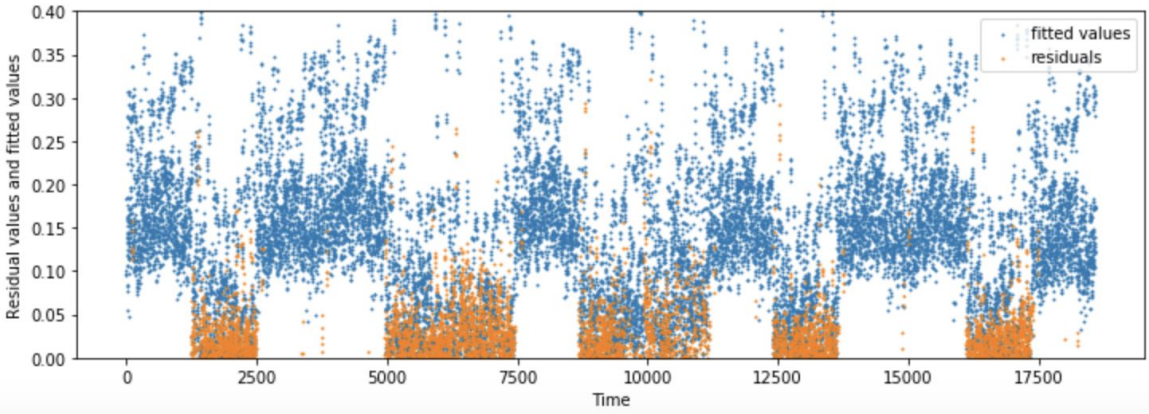


hidden_units:25
Test RMSE with cross validation: 0.1028732
Train RMSE with cross validation: 0.0332647

Fitted values vs. true values

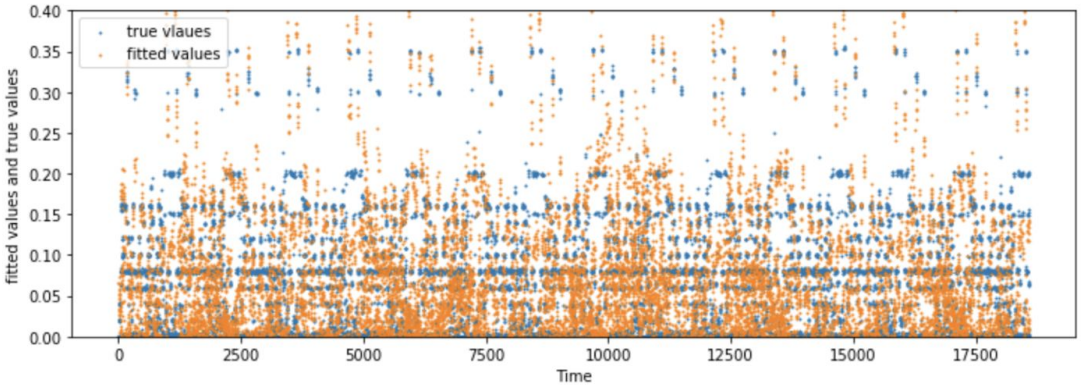


residuals vs. fitted values

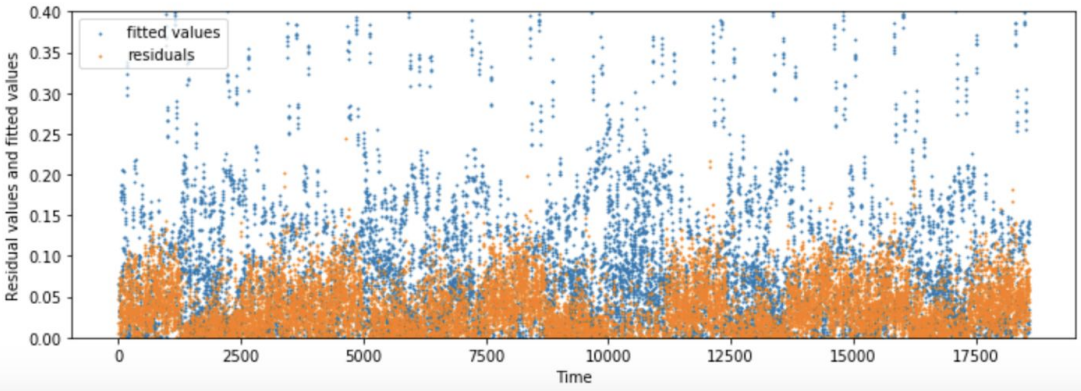


hidden_units:35
Test RMSE with cross validation: 0.0470812
Train RMSE with cross validation: 0.0256187

Fitted values vs. true values

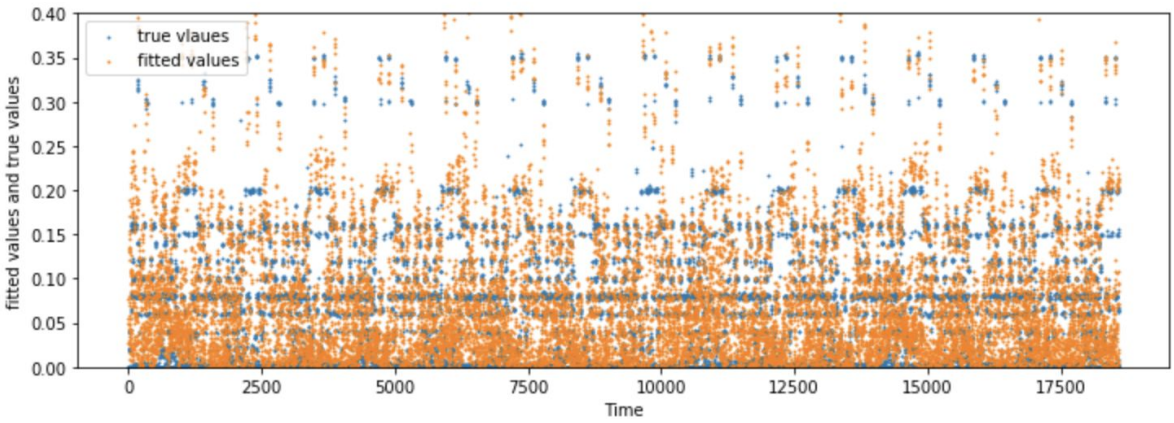


residuals vs. fitted values

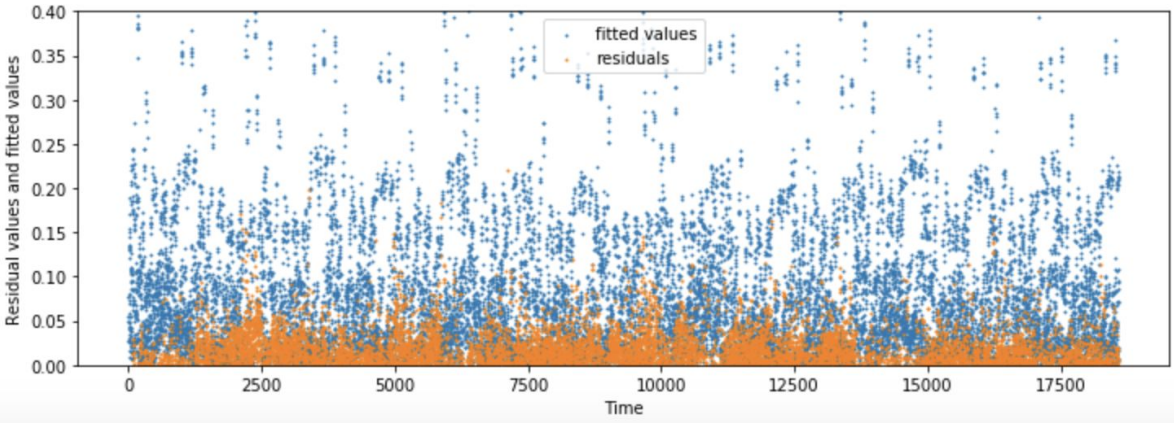


hidden_units:45
Test RMSE with cross validation: 0.0335226
Train RMSE with cross validation: 0.0243424

```
Fitted values vs. true values
```

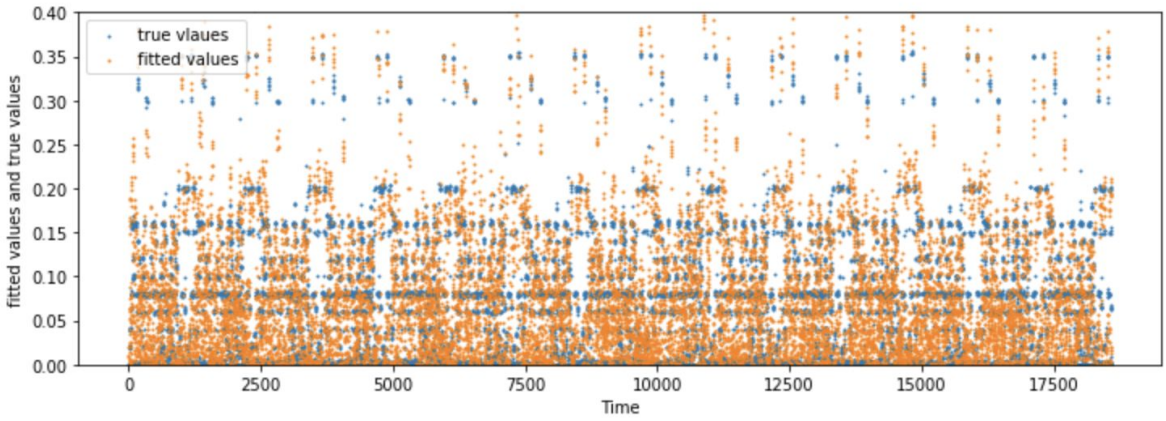



```
residuals vs. fitted values
```

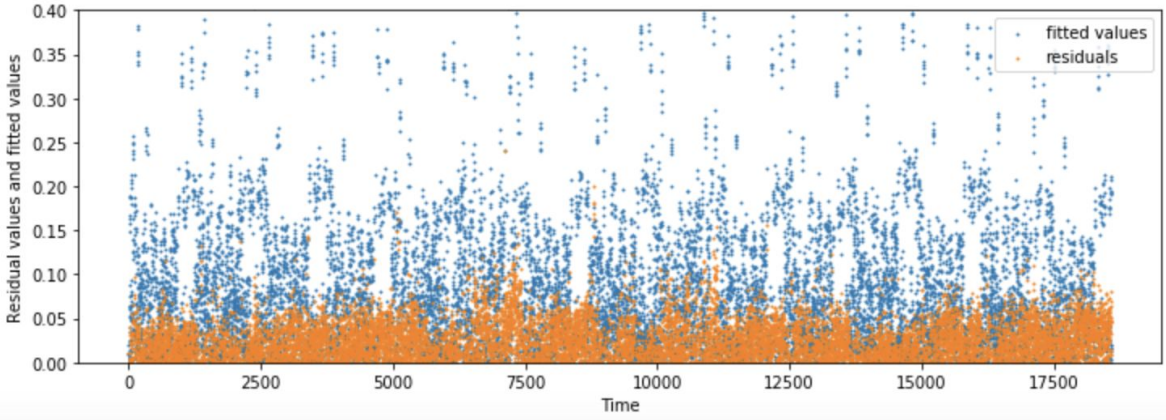


hidden_units:55
Test RMSE with cross validation: 0.0343067
Train RMSE with cross validation: 0.0238233

Fitted values vs. true values

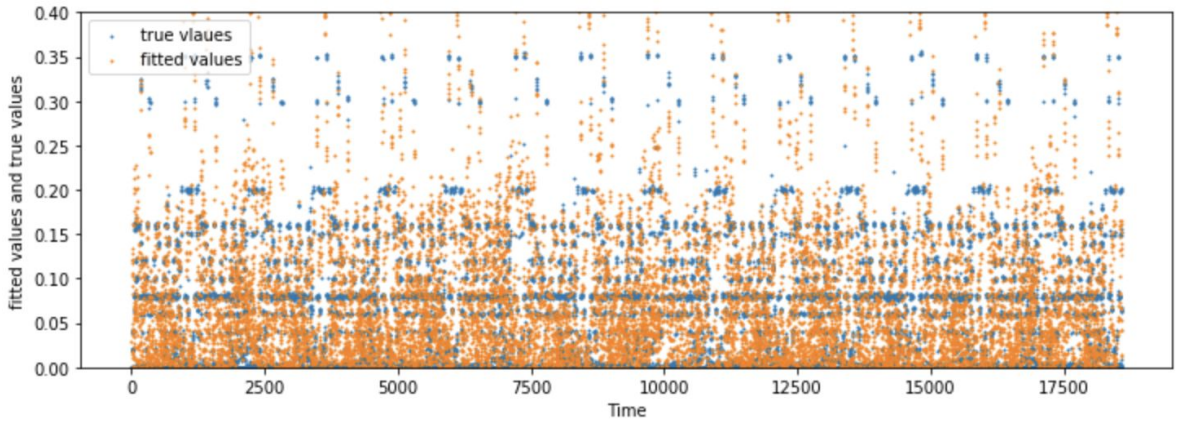


residuals vs. fitted values

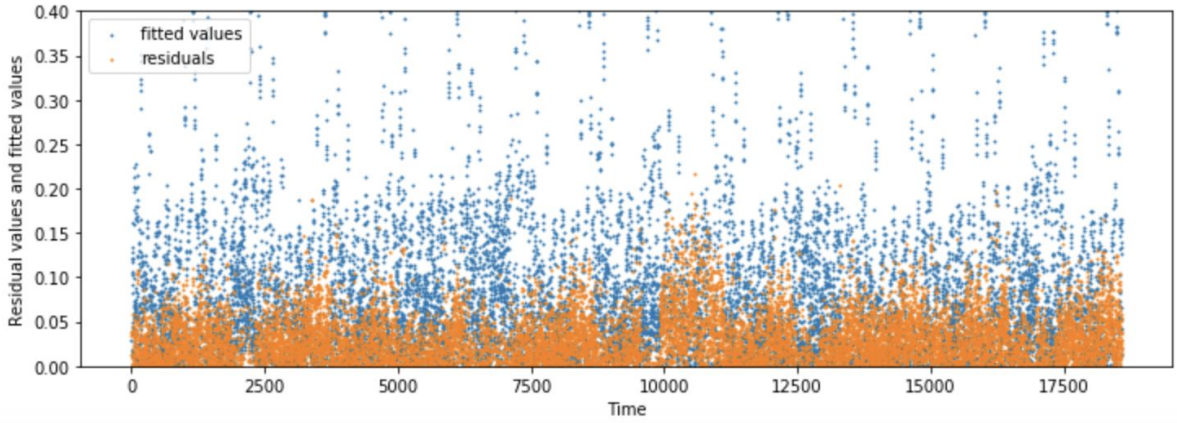


hidden_units:65
Test RMSE with cross validation: 0.0421932
Train RMSE with cross validation: 0.0281629

Fitted values vs. true values

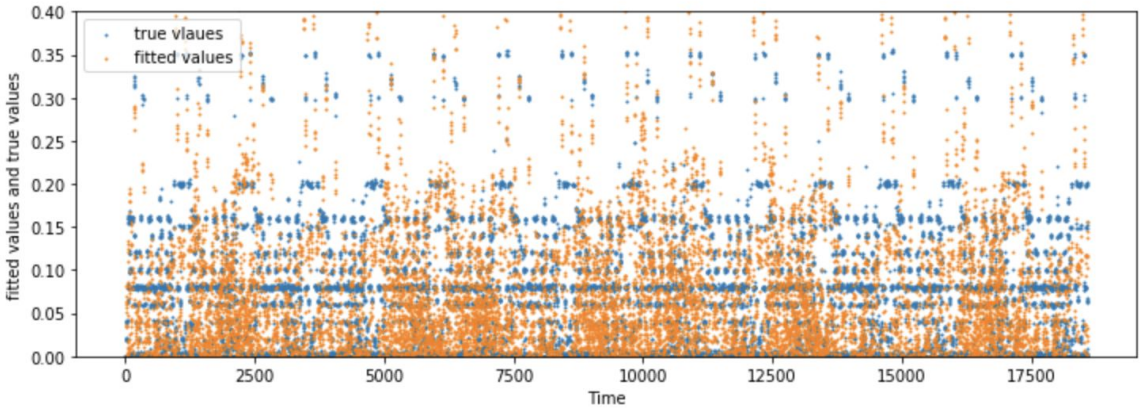


residuals vs. fitted values

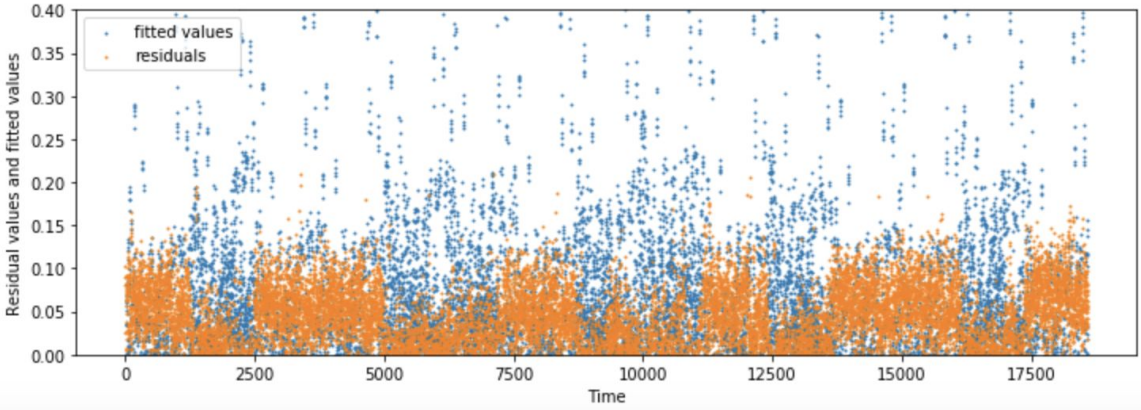


hidden_units:75
Test RMSE with cross validation: 0.0549438
Train RMSE with cross validation: 0.0270968

```
Fitted values vs. true values
```

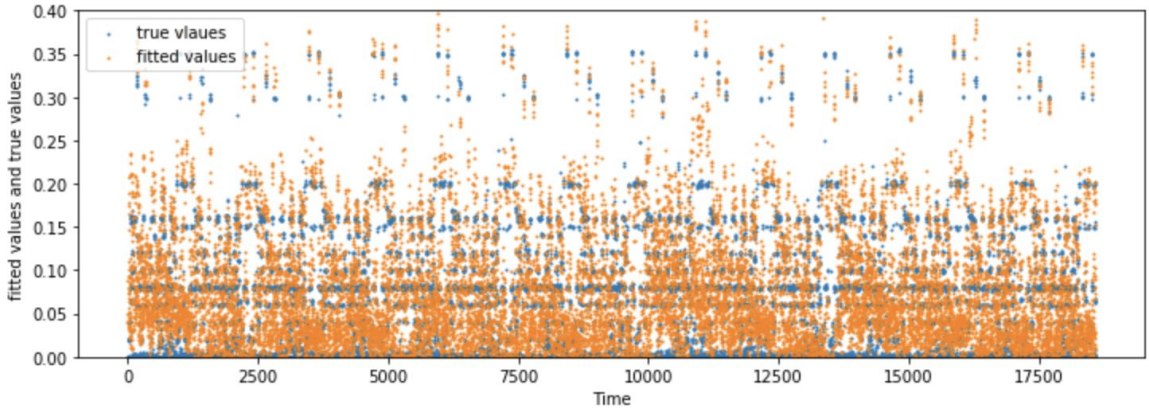



```
residuals vs. fitted values
```

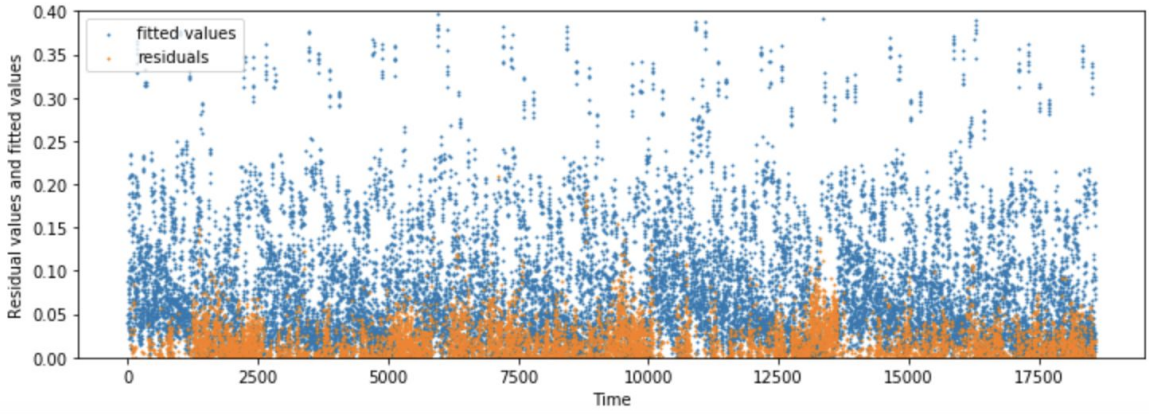


hidden_units:85
Test RMSE with cross validation: 0.0367662
Train RMSE with cross validation: 0.0203984

Fitted values vs. true values

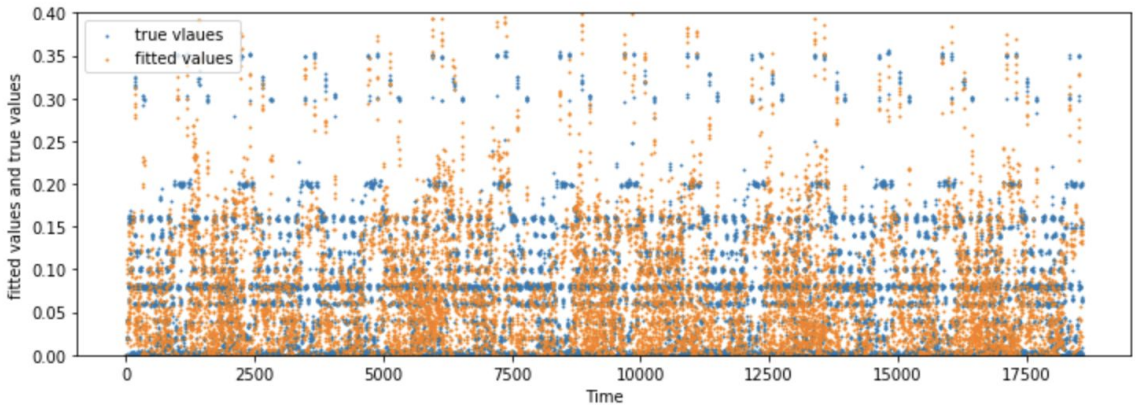


residuals vs. fitted values

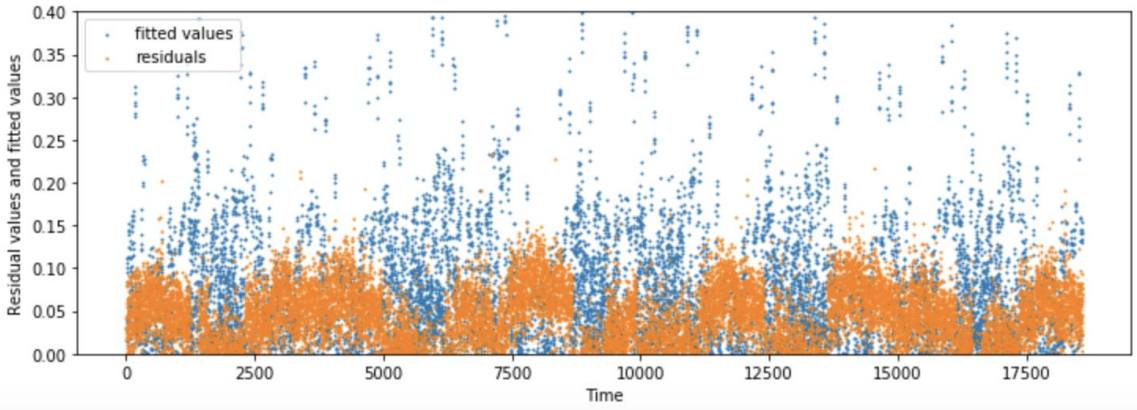


hidden_units:95
Test RMSE with cross validation: 0.0547244
Train RMSE with cross validation: 0.0224440

Fitted values vs. true values



residuals vs. fitted values

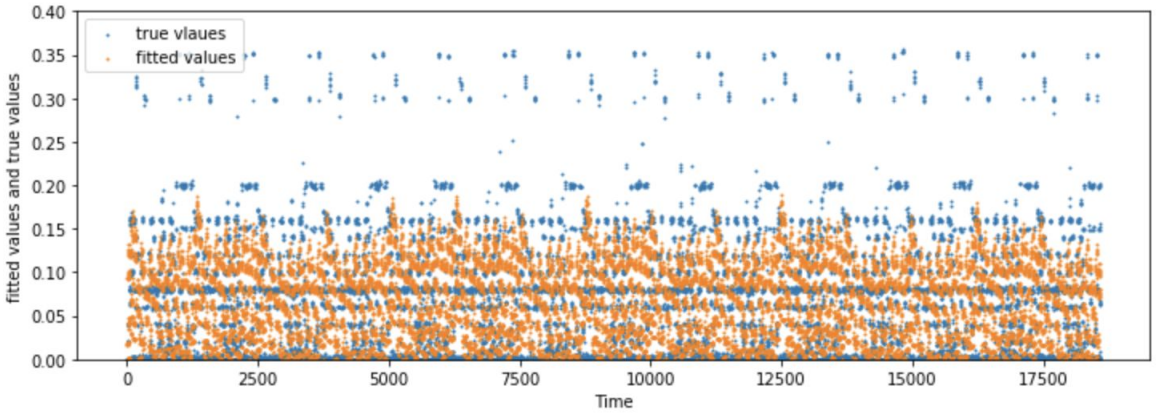


2.3.2 Logistic

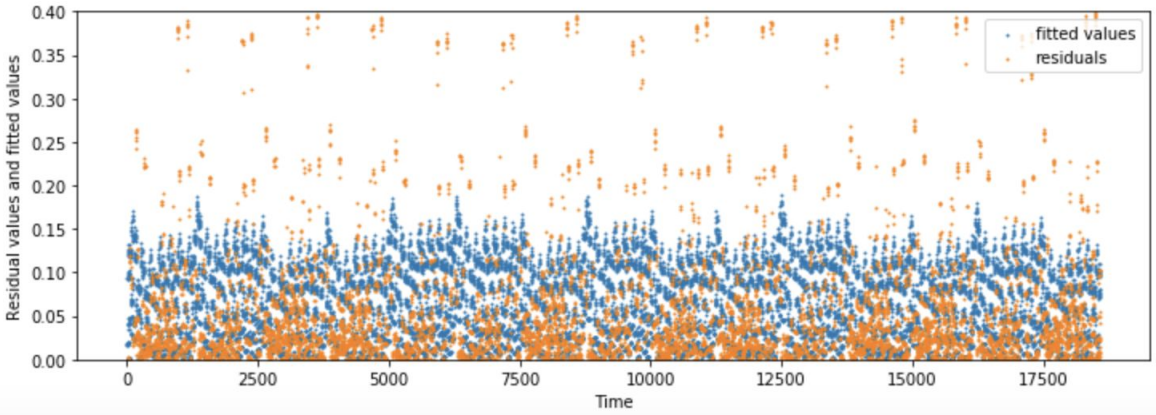
We will set the activate function as Logistic, and the hidden units from 5 to 105. Step is 10. And we get the result shown below:

hidden_units:5
Test RMSE with cross validation: 0.0903741
Train RMSE with cross validation: 0.0886132

Fitted values vs. true values

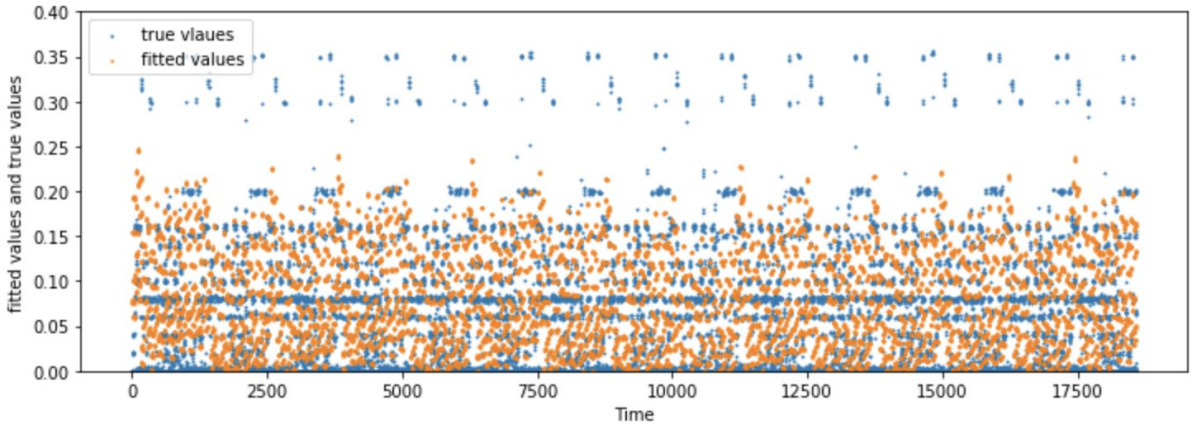


residuals vs. fitted values

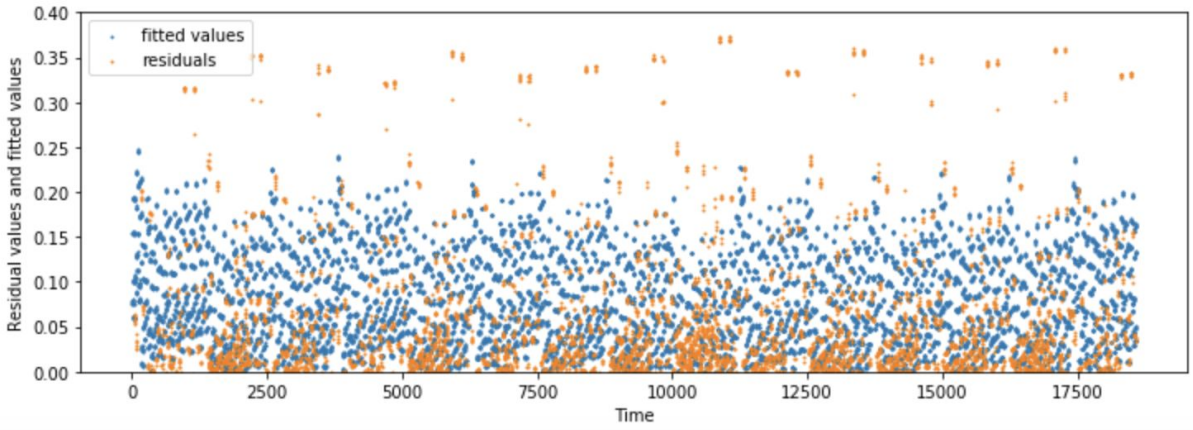


hidden_units:15
Test RMSE with cross validation: 0.0904965
Train RMSE with cross validation: 0.0883978

```
Fitted values vs. true values
```

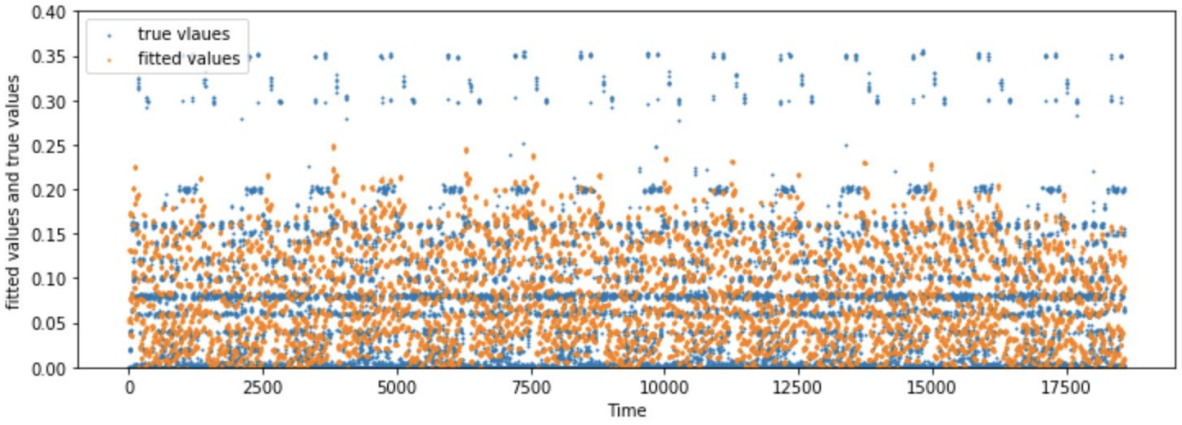



```
residuals vs. fitted values
```

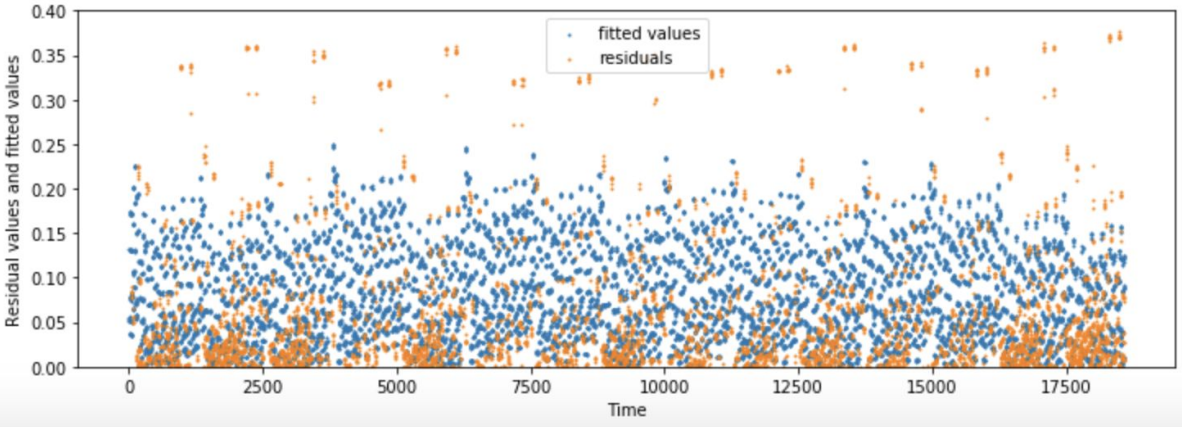


hidden_units:25
Test RMSE with cross validation: 0.0907964
Train RMSE with cross validation: 0.0884249

Fitted values vs. true values

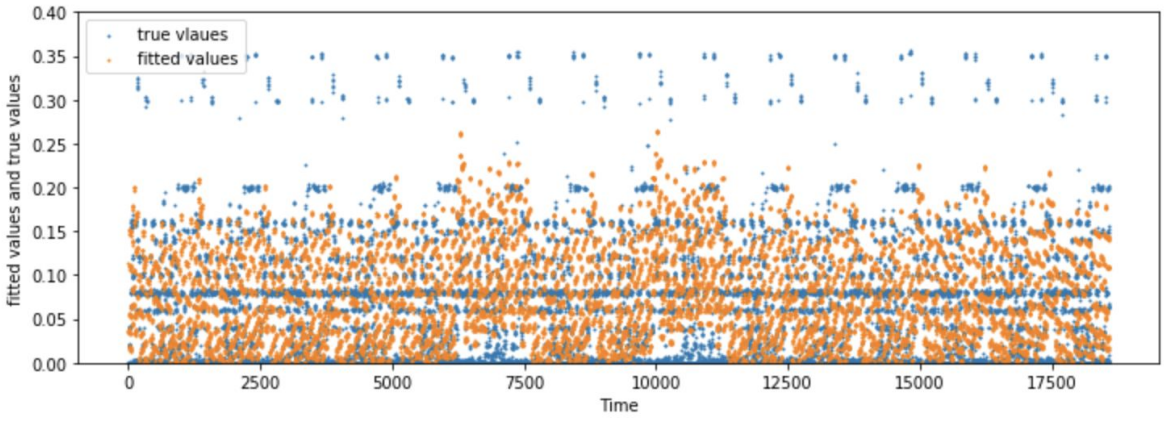


residuals vs. fitted values

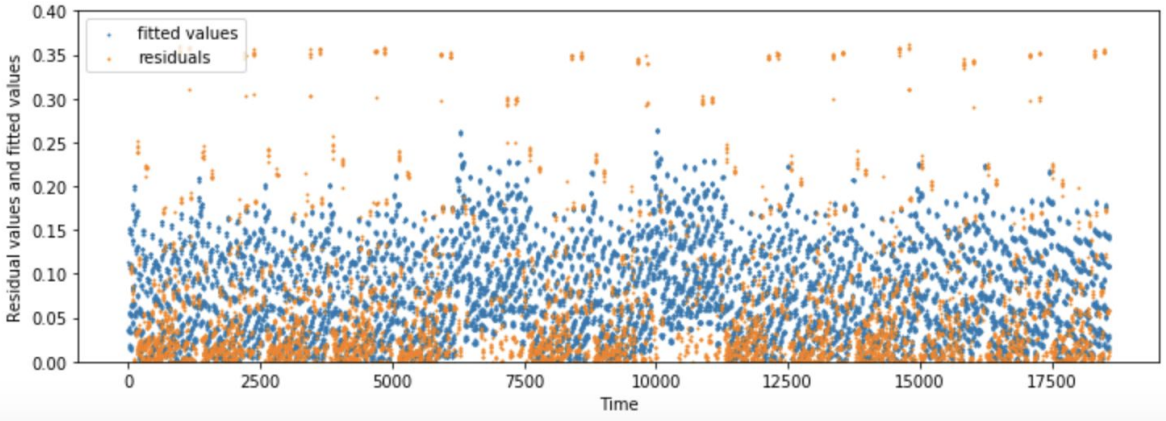


hidden_units:35
Test RMSE with cross validation: 0.0912045
Train RMSE with cross validation: 0.0887311

Fitted values vs. true values

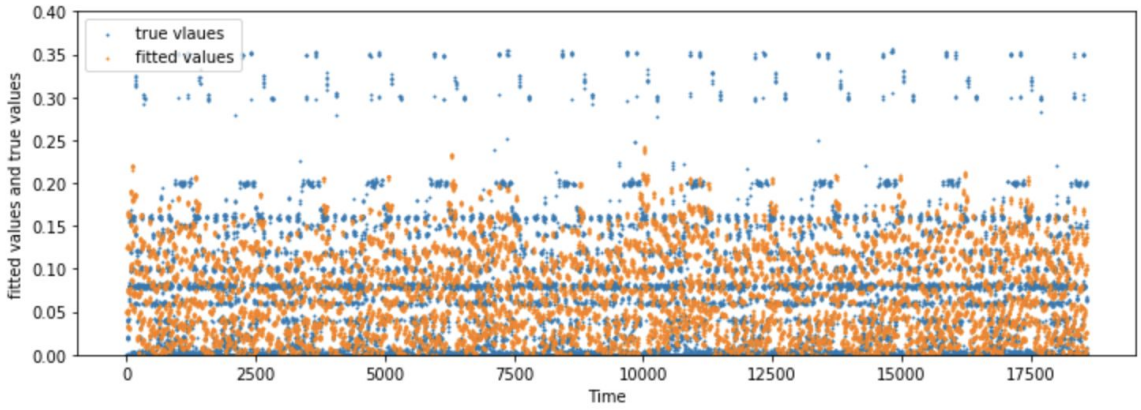


residuals vs. fitted values

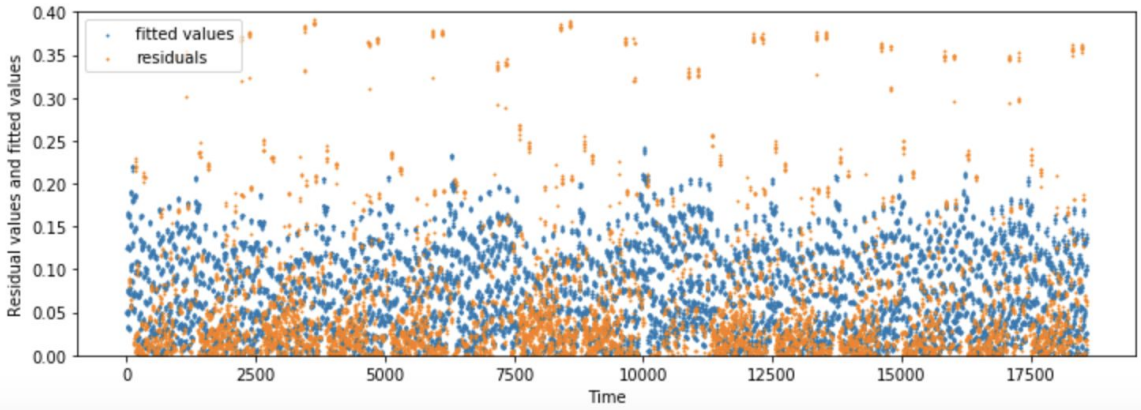


hidden_units:45
Test RMSE with cross validation: 0.0898509
Train RMSE with cross validation: 0.0888340

Fitted values vs. true values

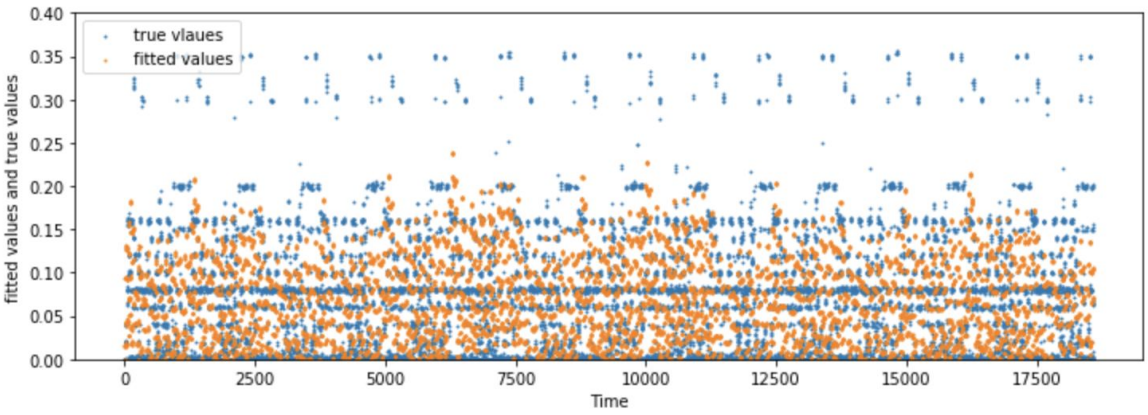


residuals vs. fitted values

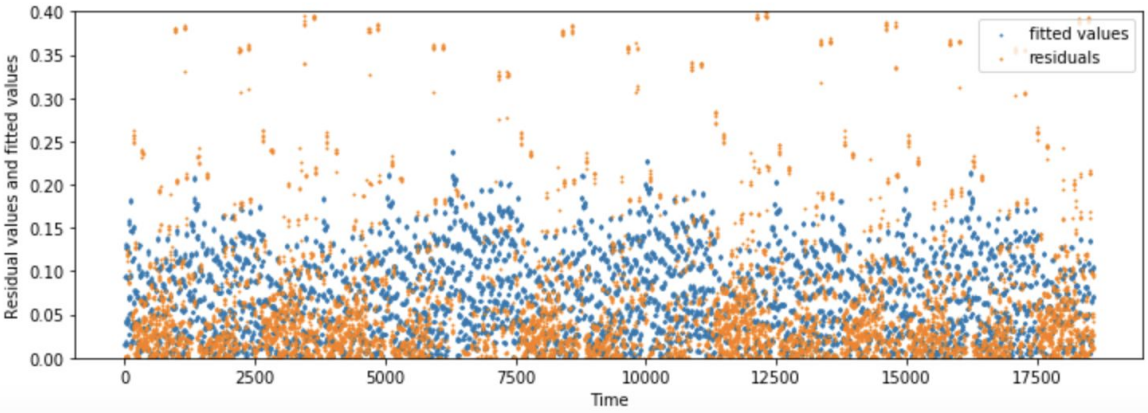


hidden_units:55
Test RMSE with cross validation: 0.0913186
Train RMSE with cross validation: 0.0883993

Fitted values vs. true values

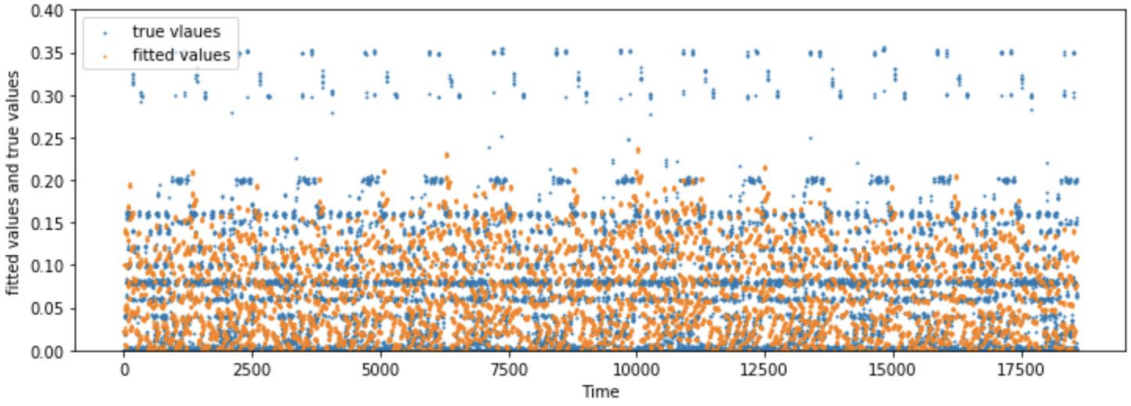


residuals vs. fitted values

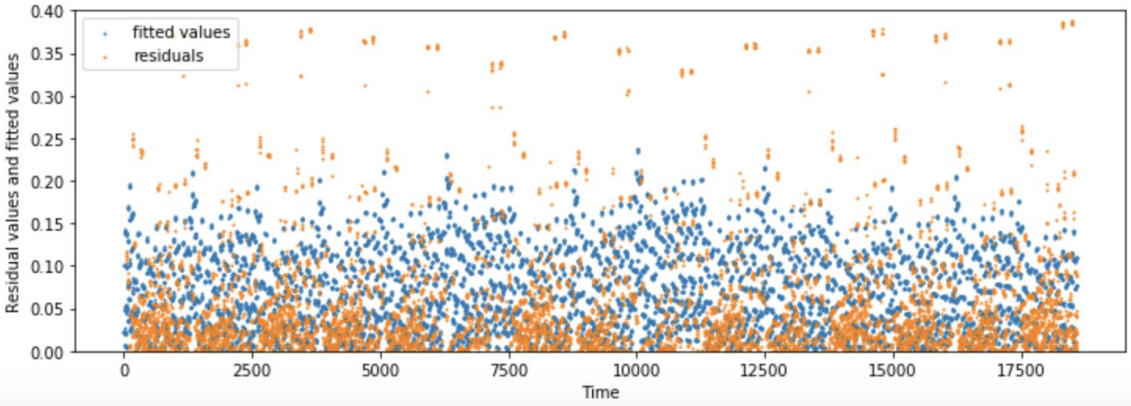


hidden_units:65
Test RMSE with cross validation: 0.0899202
Train RMSE with cross validation: 0.0884044

Fitted values vs. true values

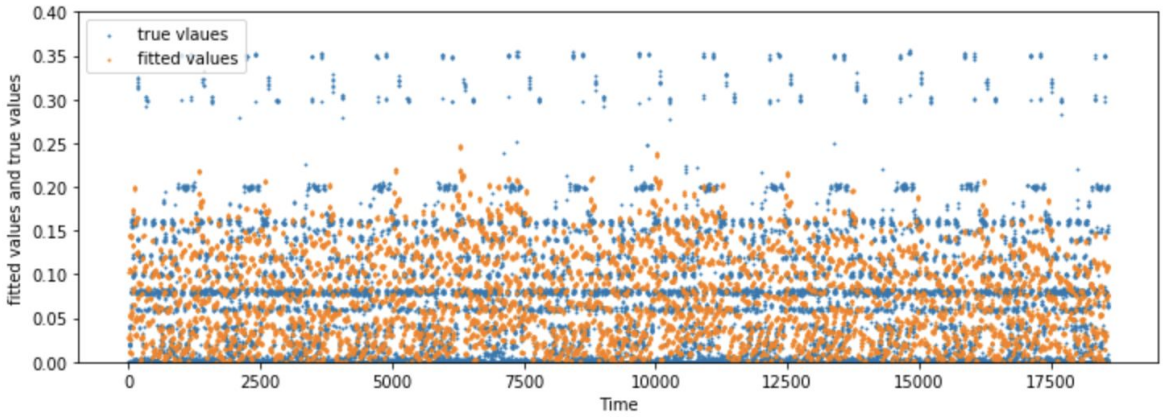


residuals vs. fitted values

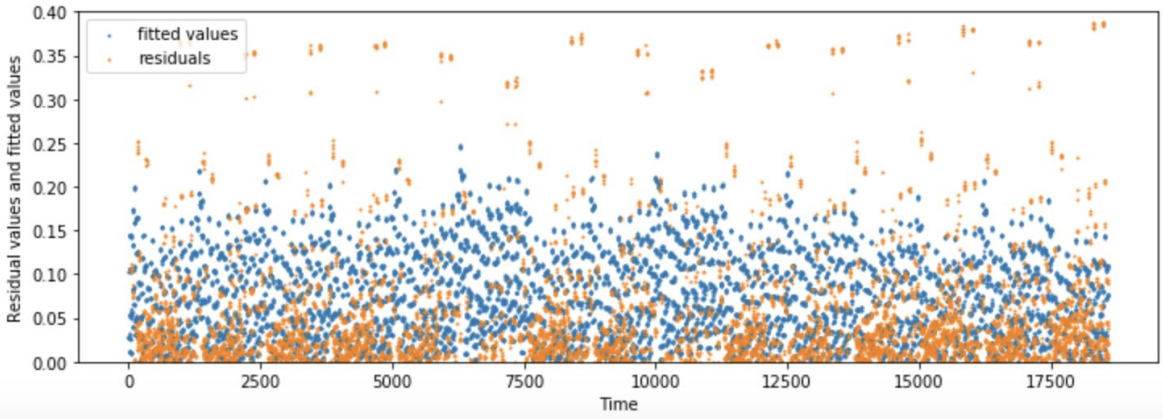


hidden_units:75
Test RMSE with cross validation: 0.0901195
Train RMSE with cross validation: 0.0885306

Fitted values vs. true values

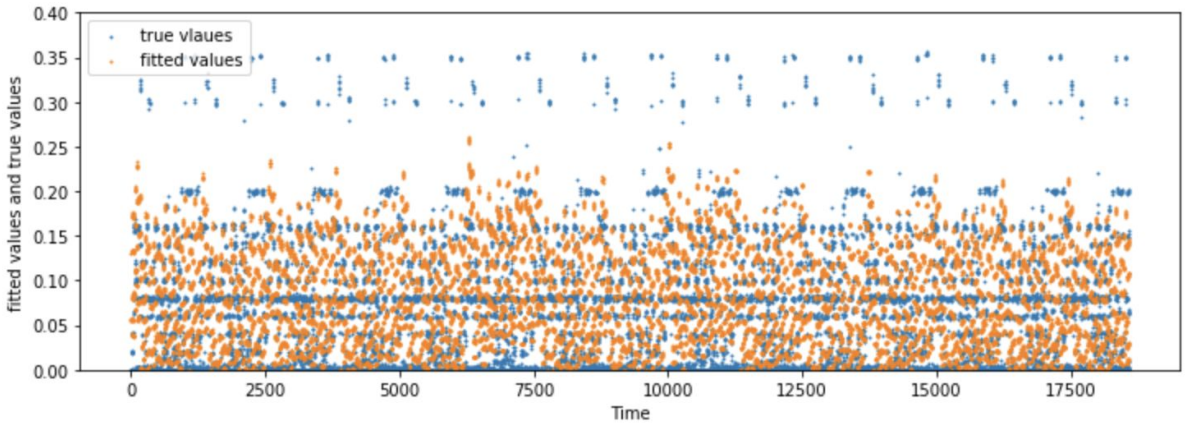


residuals vs. fitted values

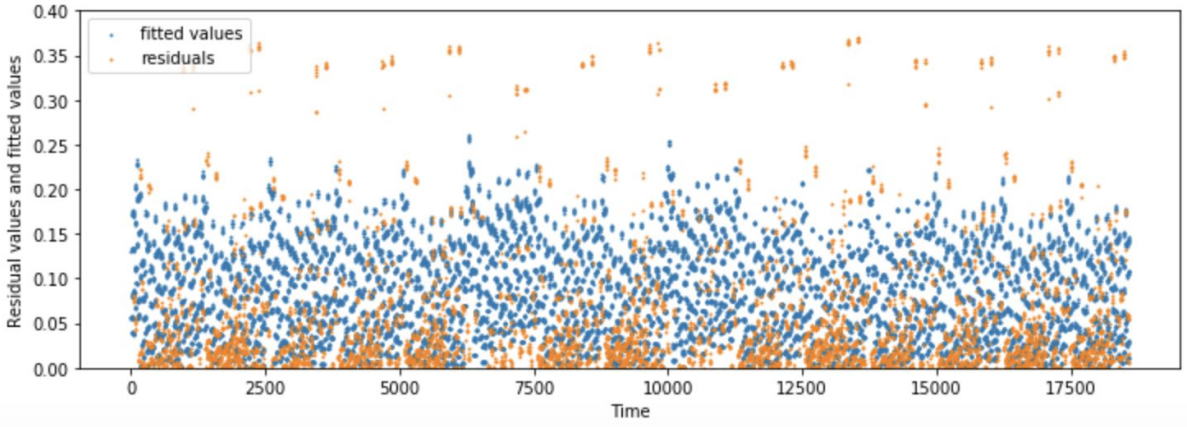


hidden_units:85
Test RMSE with cross validation: 0.0904659
Train RMSE with cross validation: 0.0884202

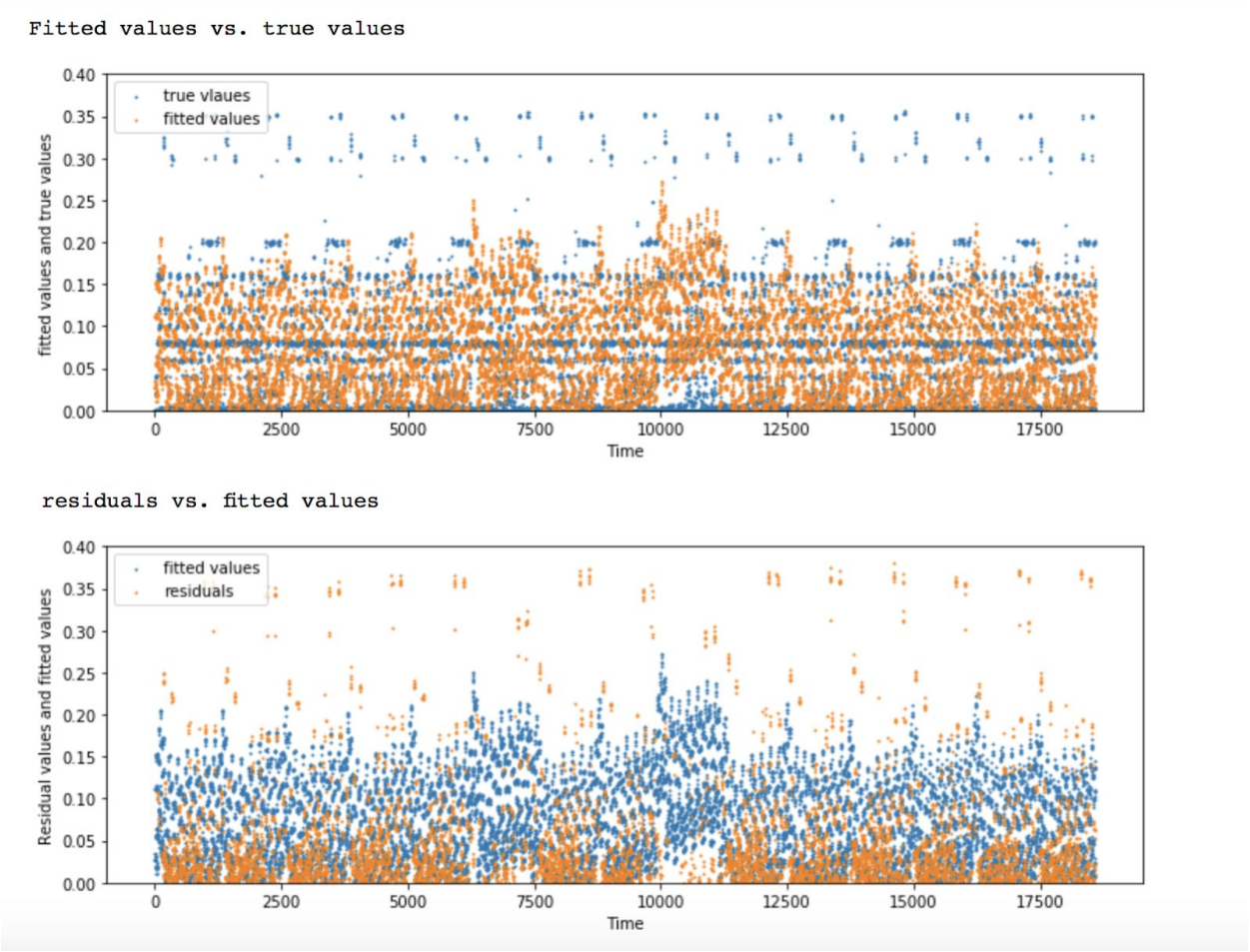
```
Fitted values vs. true values
```




```
residuals vs. fitted values
```



hidden_units:95
Test RMSE with cross validation: 0.0910180
Train RMSE with cross validation: 0.0886494

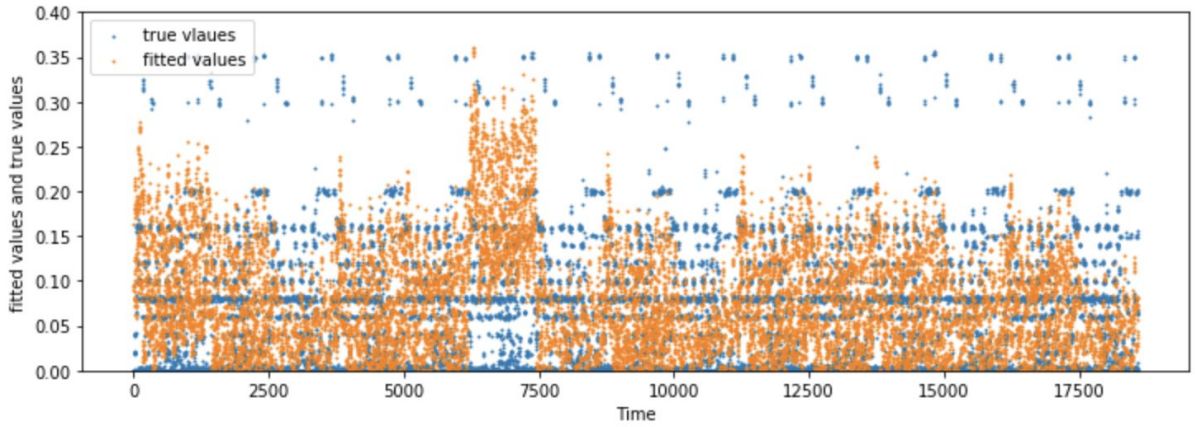


2.3.3 Tanh

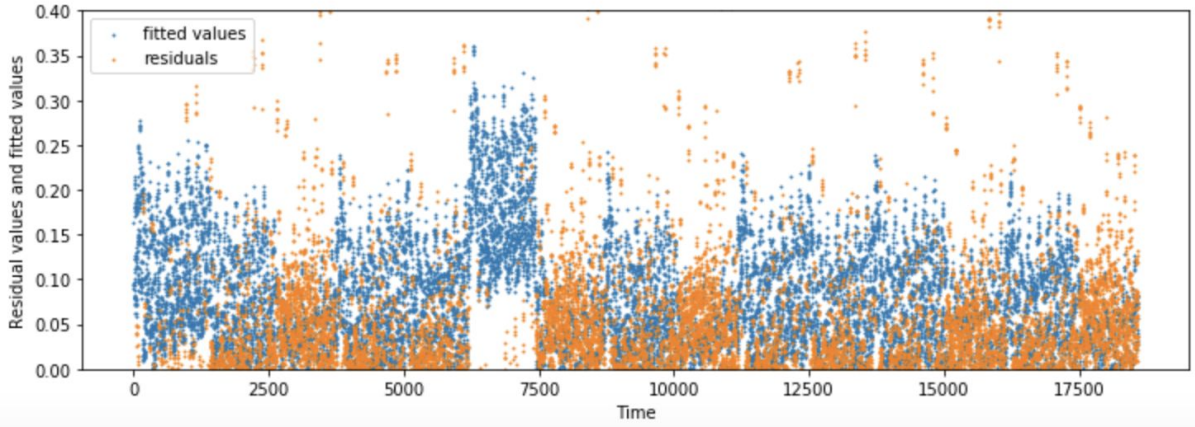
We will set the activate function as Tanh, and the hidden units from 5 to 105. Step is 10. And we get the result shown below:

hidden_units:5
Test RMSE with cross validation: 0.1004719
Train RMSE with cross validation: 0.0877765

Fitted values vs. true values

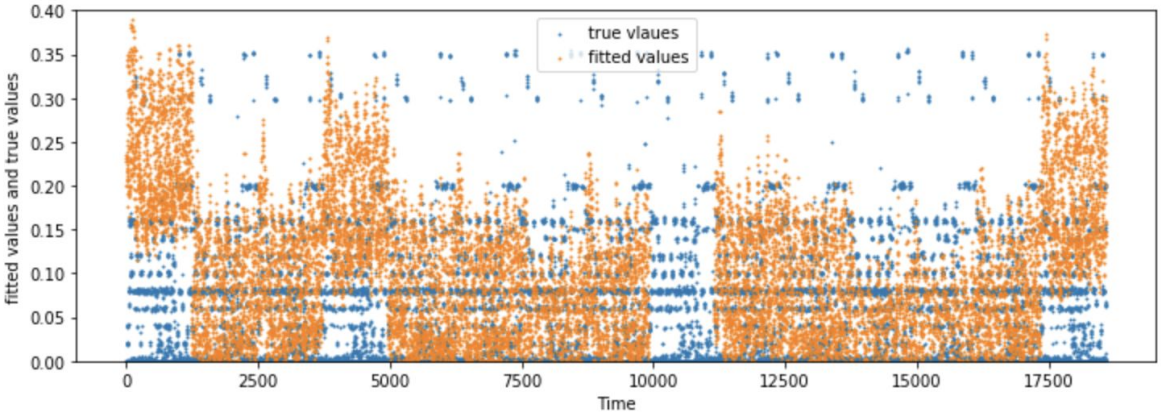


residuals vs. fitted values

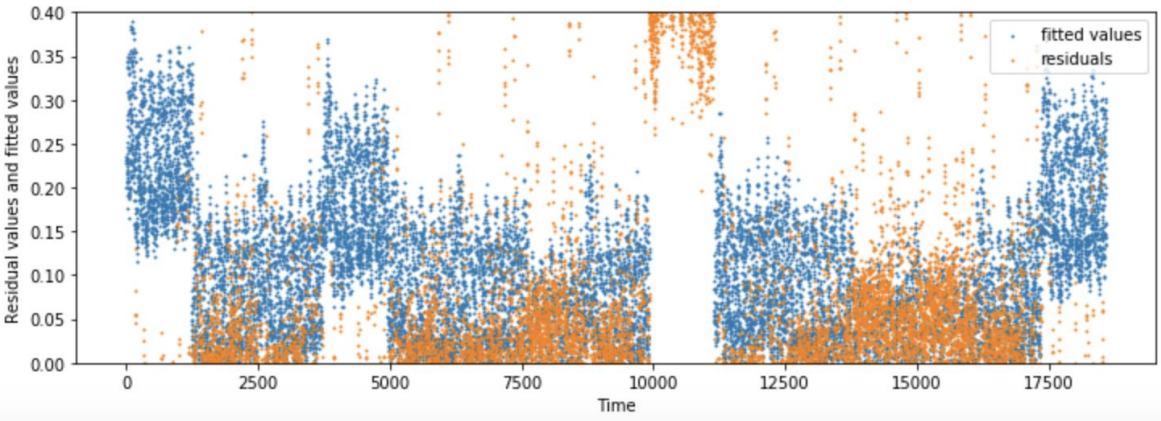


hidden_units:15
Test RMSE with cross validation: 0.1593951
Train RMSE with cross validation: 0.0885327

Fitted values vs. true values

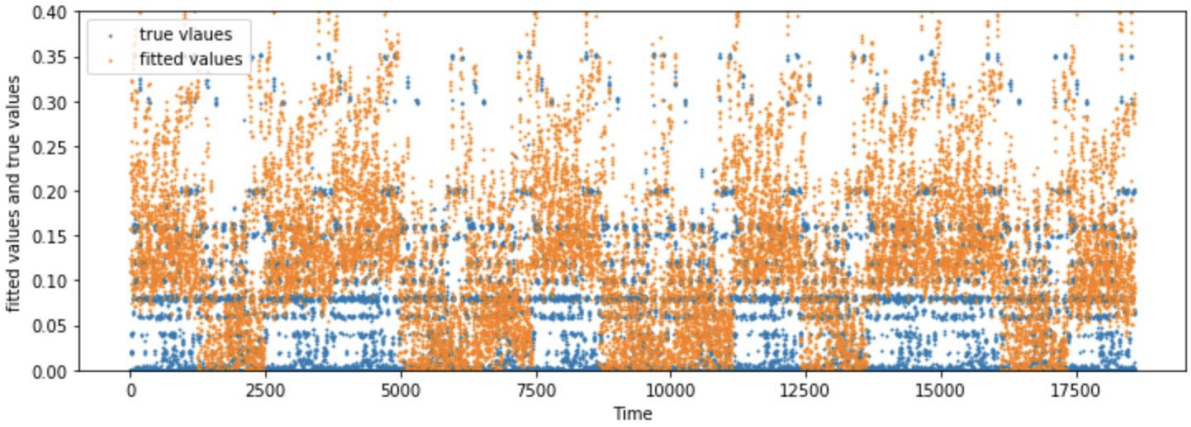


residuals vs. fitted values

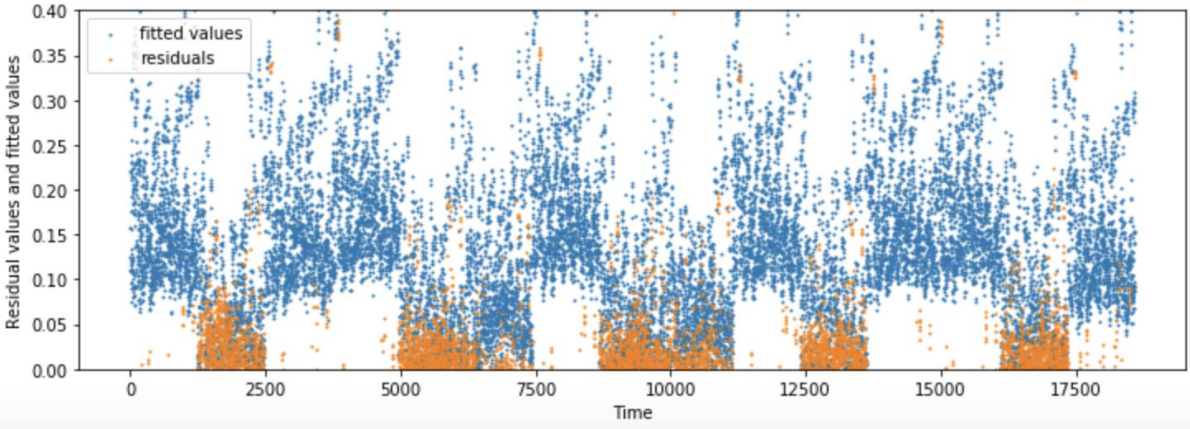


hidden_units:25
Test RMSE with cross validation: 0.1009656
Train RMSE with cross validation: 0.0530941

Fitted values vs. true values

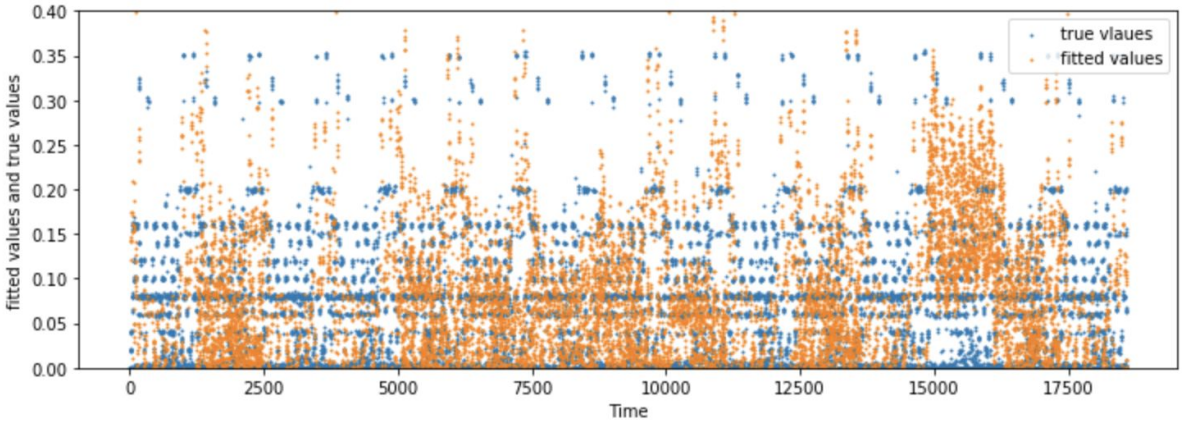


residuals vs. fitted values

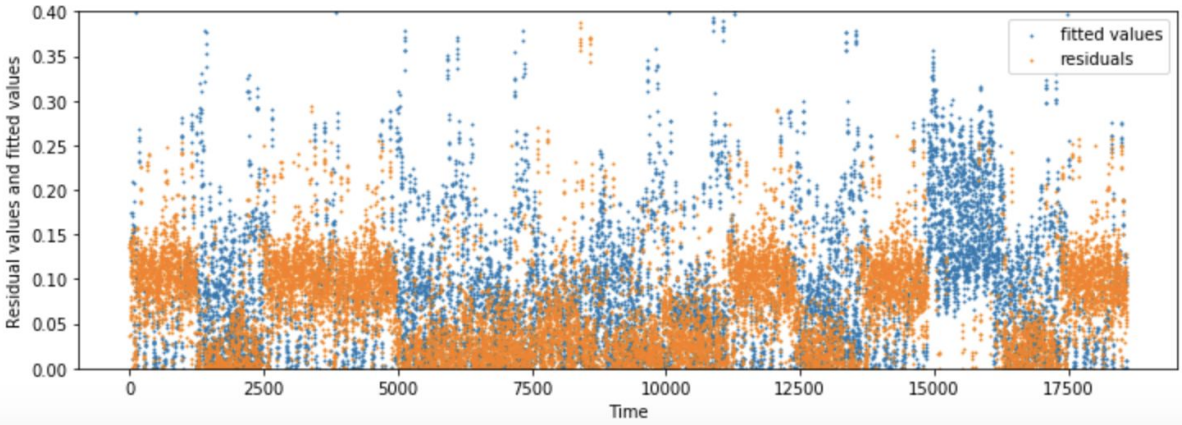


hidden_units:35
Test RMSE with cross validation: 0.0957010
Train RMSE with cross validation: 0.0637125

Fitted values vs. true values

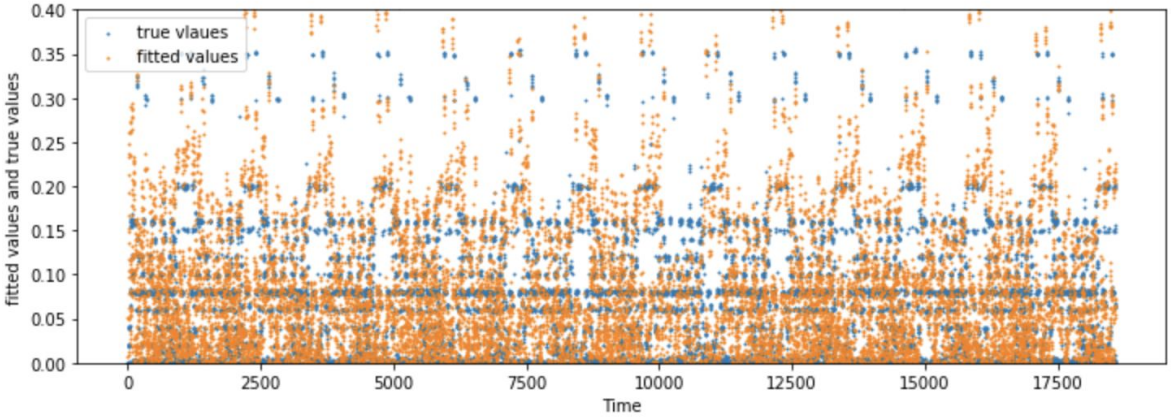


residuals vs. fitted values

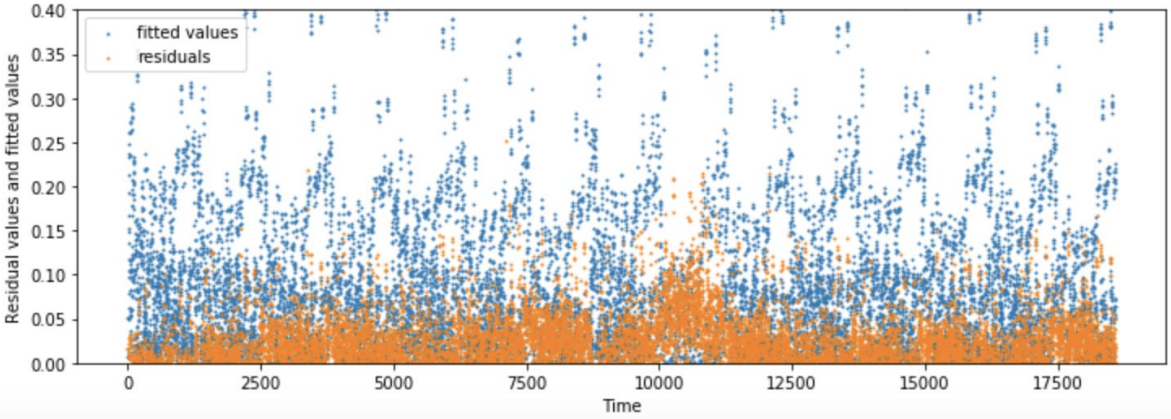


hidden_units:45
Test RMSE with cross validation: 0.0562576
Train RMSE with cross validation: 0.0526471

Fitted values vs. true values

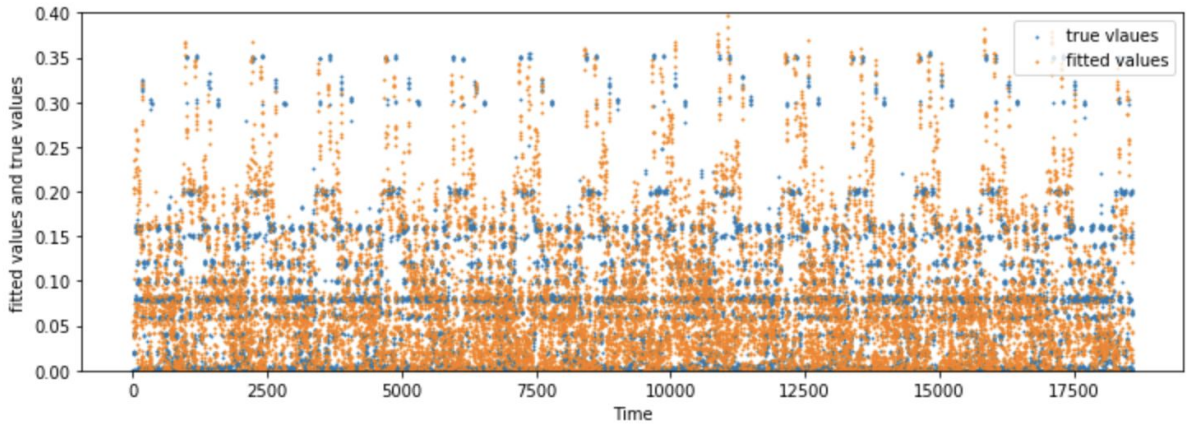


residuals vs. fitted values

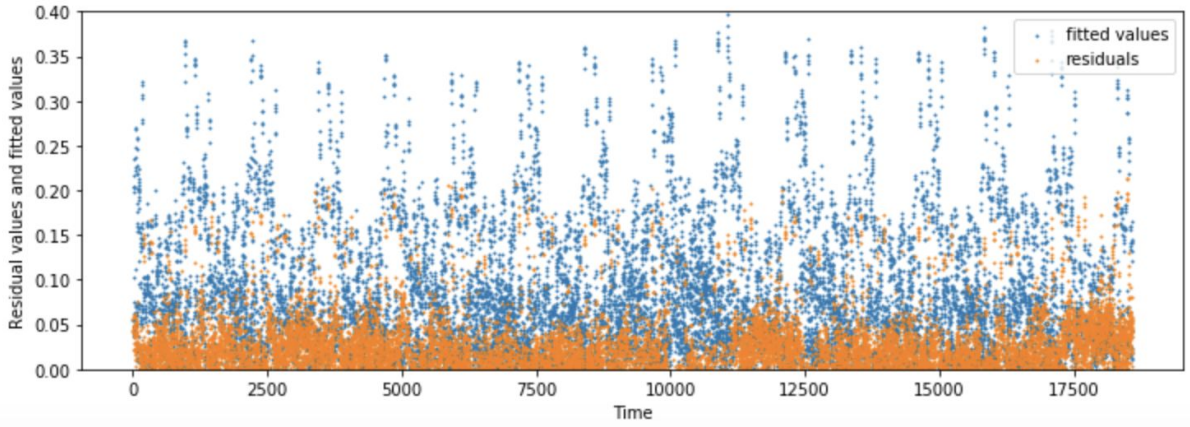


hidden_units:55
Test RMSE with cross validation: 0.0571316
Train RMSE with cross validation: 0.0548299

```
Fitted values vs. true values
```

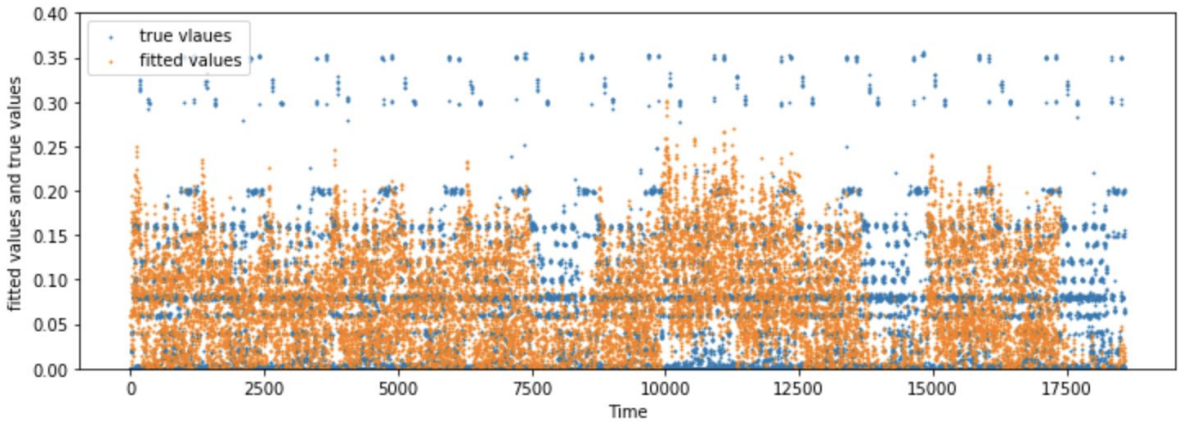



```
residuals vs. fitted values
```

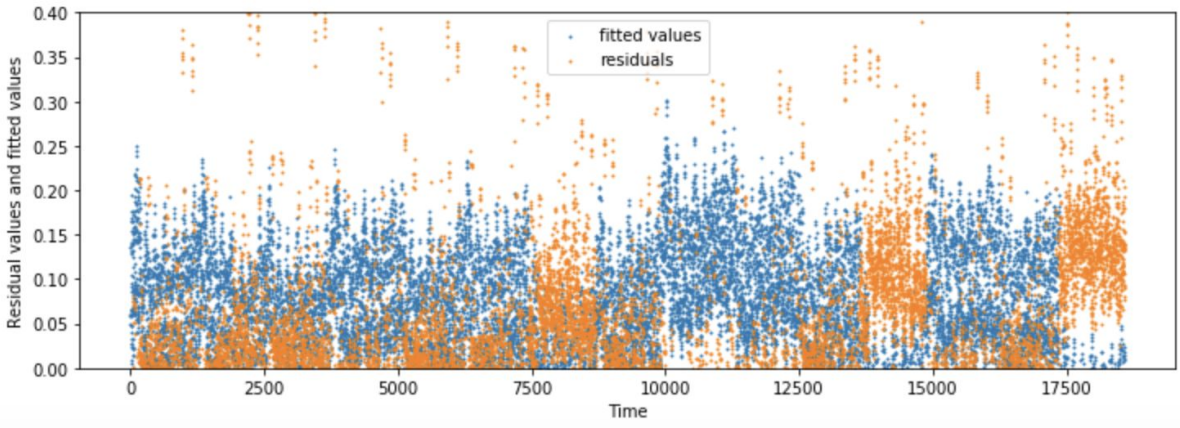


hidden_units:65
Test RMSE with cross validation: 0.1059473
Train RMSE with cross validation: 0.0887478

Fitted values vs. true values

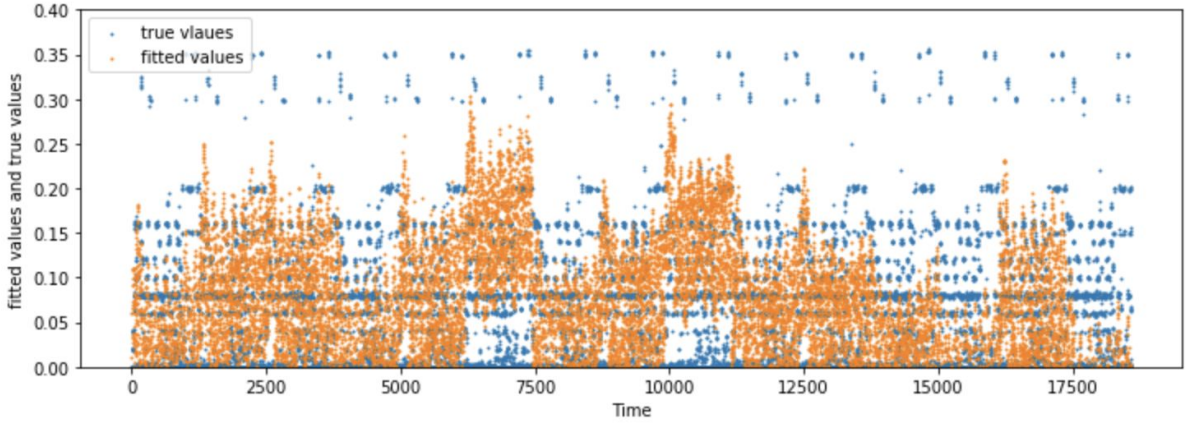


residuals vs. fitted values

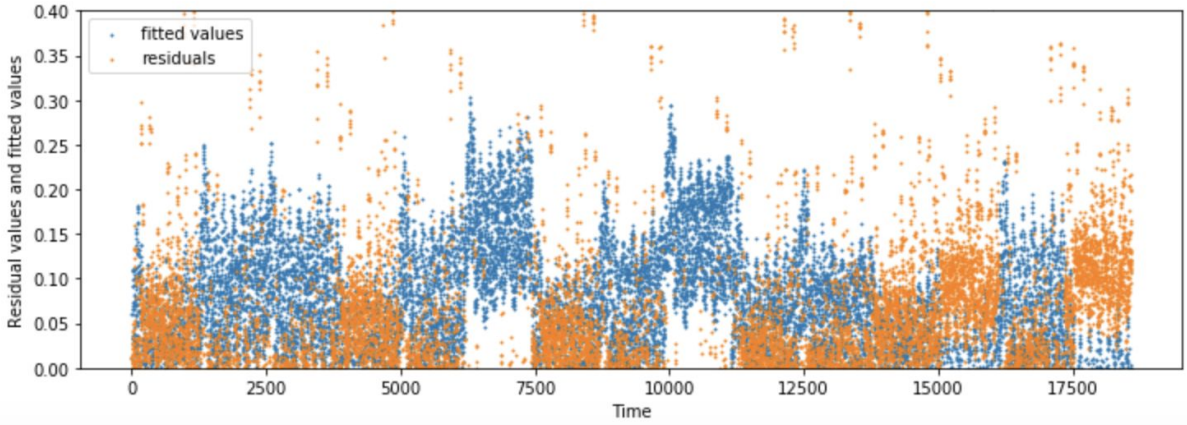


hidden_units:75
Test RMSE with cross validation: 0.1072888
Train RMSE with cross validation: 0.0889648

```
Fitted values vs. true values
```

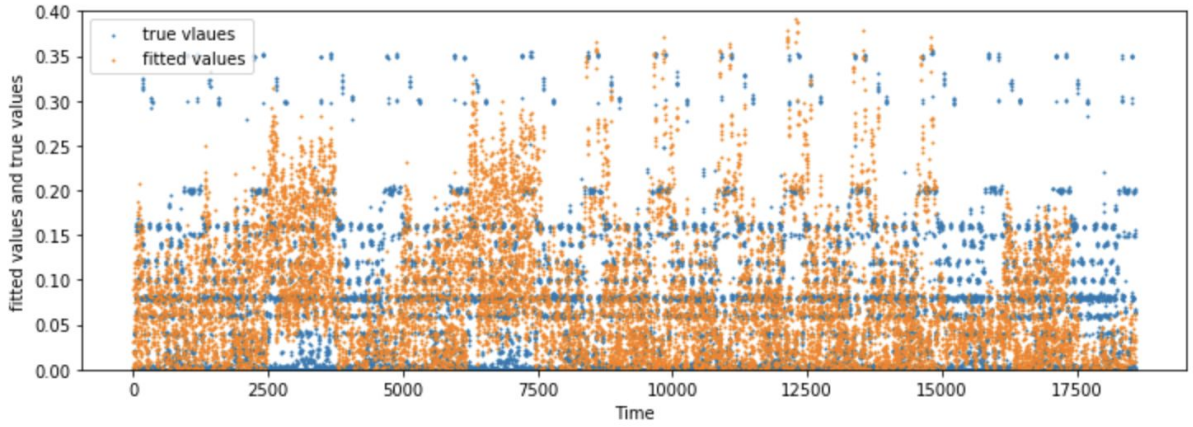



```
residuals vs. fitted values
```

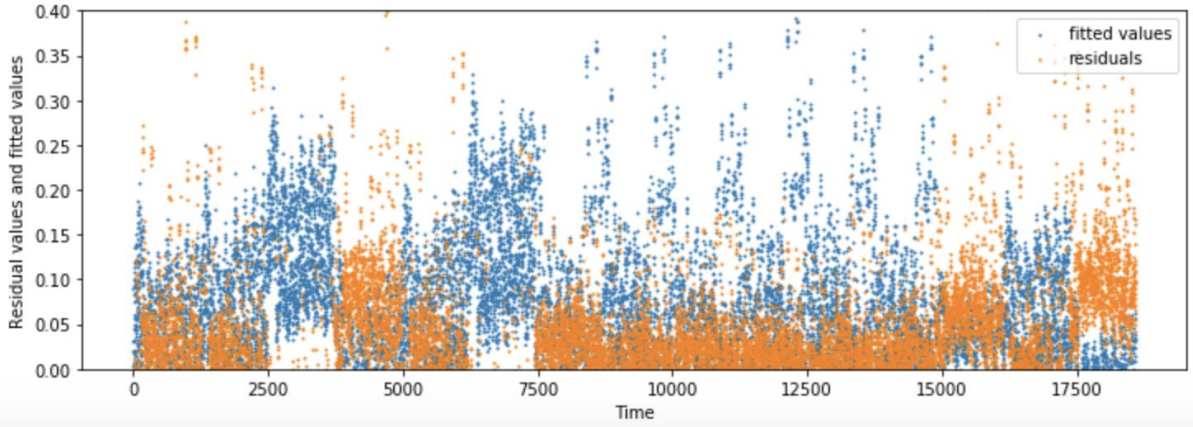


hidden_units:85
Test RMSE with cross validation: 0.0920817
Train RMSE with cross validation: 0.0761157

Fitted values vs. true values

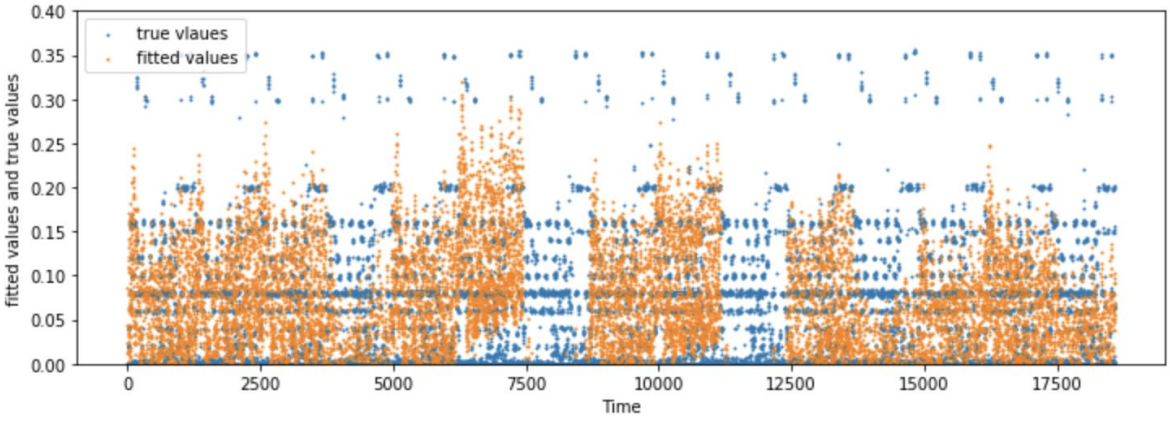


residuals vs. fitted values

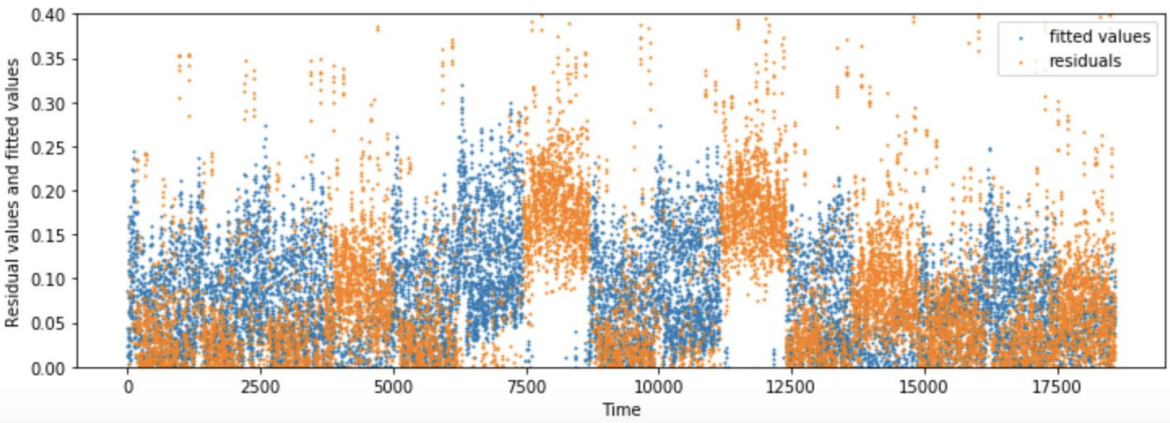


hidden_units:95
Test RMSE with cross validation: 0.1225303
Train RMSE with cross validation: 0.0880251

Fitted values vs. true values



residuals vs. fitted values



2.3.4 Conclusion

If we compare the RMSE in each activate function, we find that:
In Relu, when the hidden units are 85, we can get the best performance.
In Logistic, when the hidden units are 65, we can get the best performance.
In Tanh, when the hidden units are 45, we can get the best performance.

If we compare the RMSE in each function's best performance, we find that the Test RMSE is 0.0367662, 0.0899202, 0.0562576 respectively. We can conclude that for

one hidden layer NN, implementing Relu as activate function can give us best performance.

2.4 Predict Backup Size for each workflow

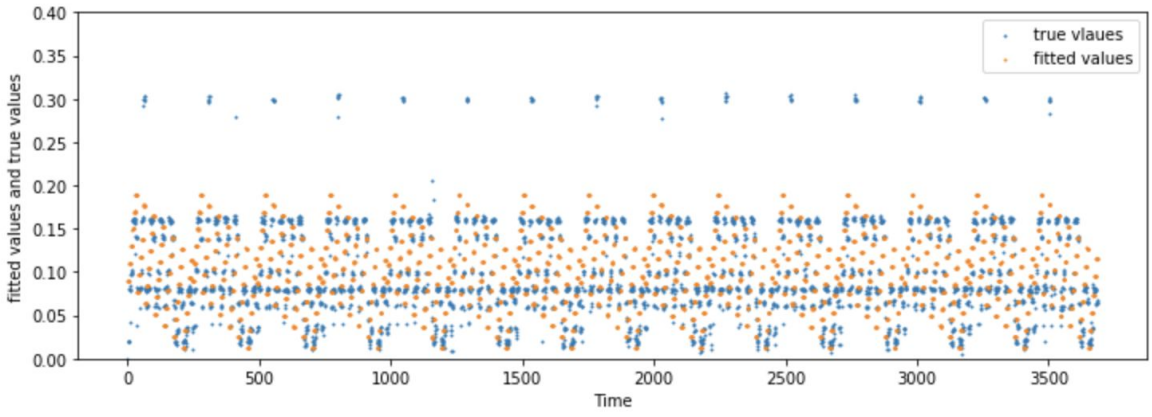
In this part, we will separate the workflow from the whole dataset and test/train them one by one and show the result in RMSE and visualized way.

2.4.1.1 Linear Regression Model

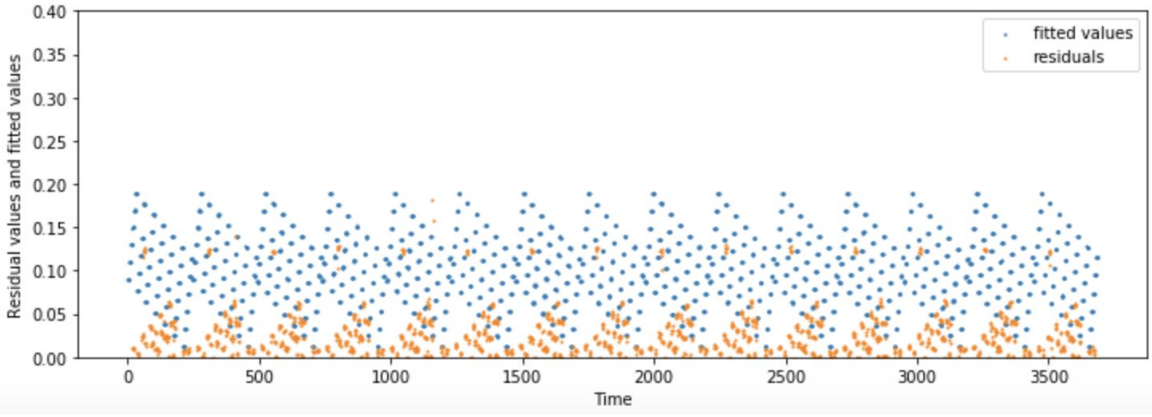
First,we will use Linear Regression Model to train and test the data in different workflow and the results are shown below:

workflow_ID 1
Test RMSE with cross validation: 0.0358870
Train RMSE with cross validation: 0.0358355

Fitted values vs. true values

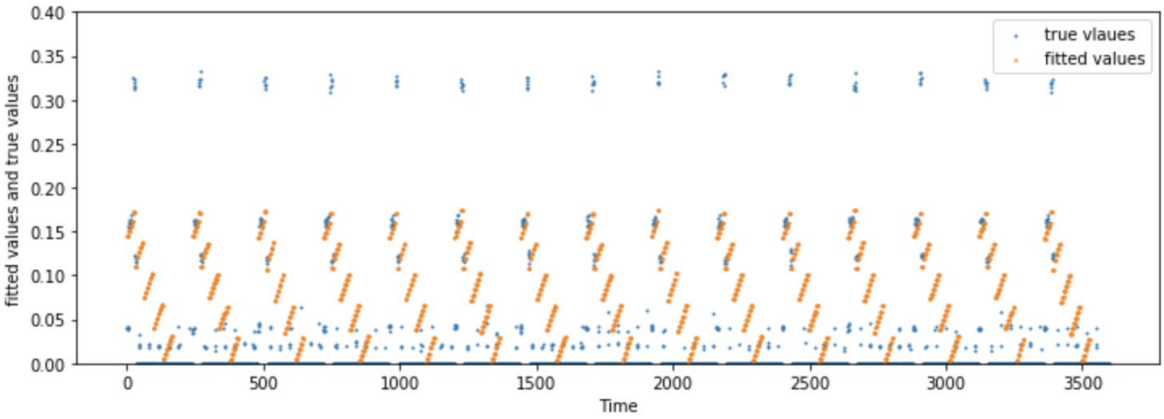


residuals vs. fitted values

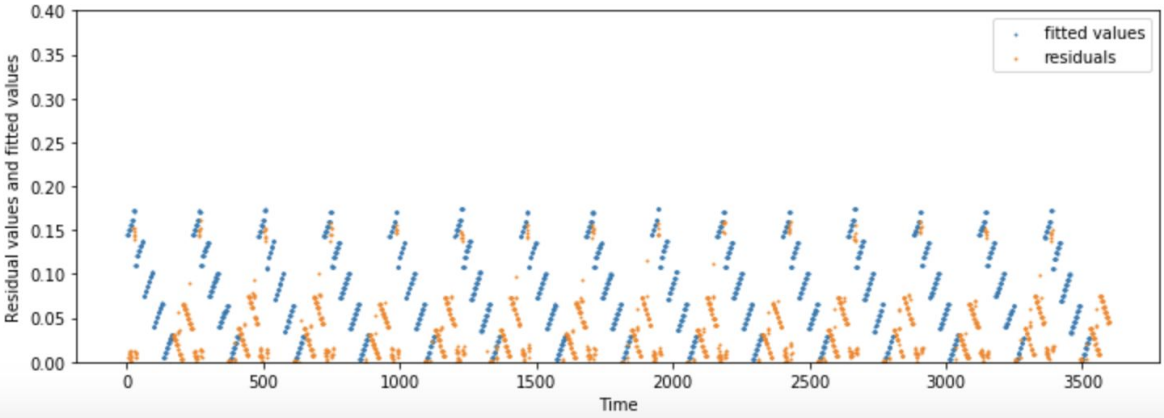


workflow_ID 2
Test RMSE with cross validation: 0.1489186
Train RMSE with cross validation: 0.1487660

Fitted values vs. true values

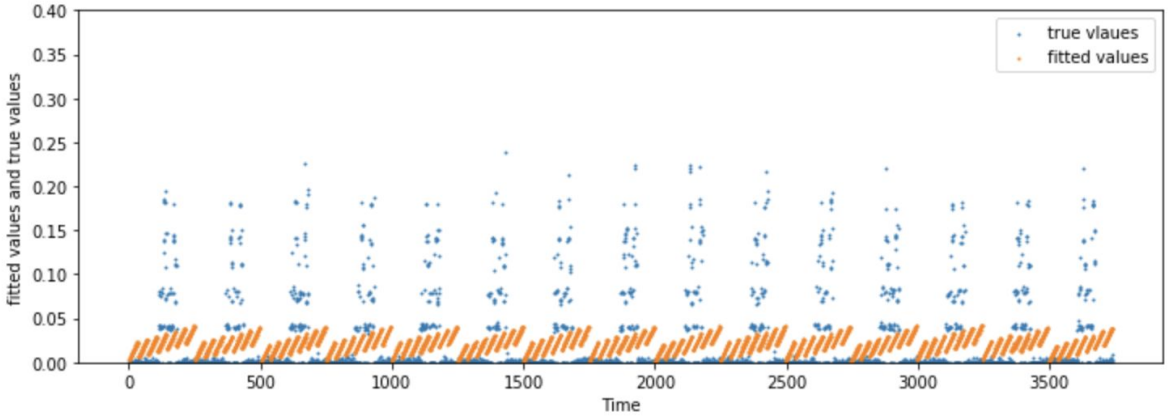


residuals vs. fitted values

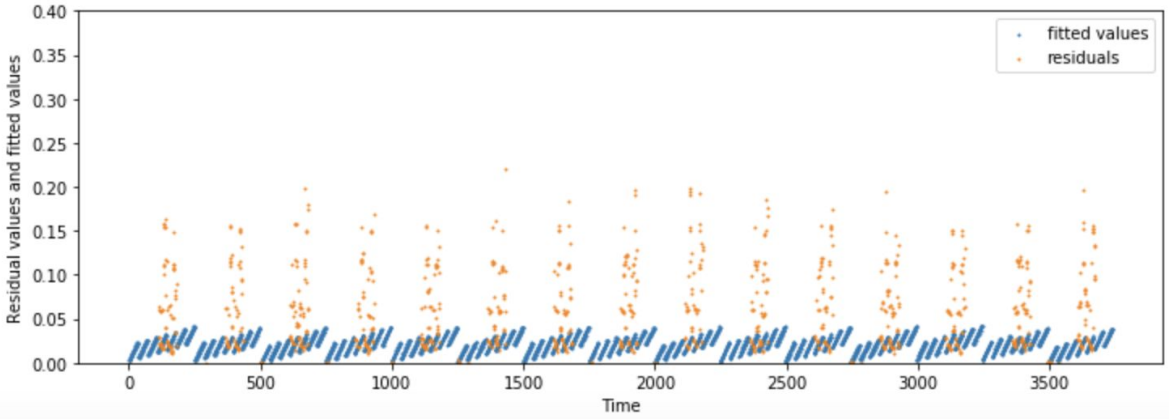


workflow_ID 3
Test RMSE with cross validation: 0.0430669
Train RMSE with cross validation: 0.0429093

Fitted values vs. true values

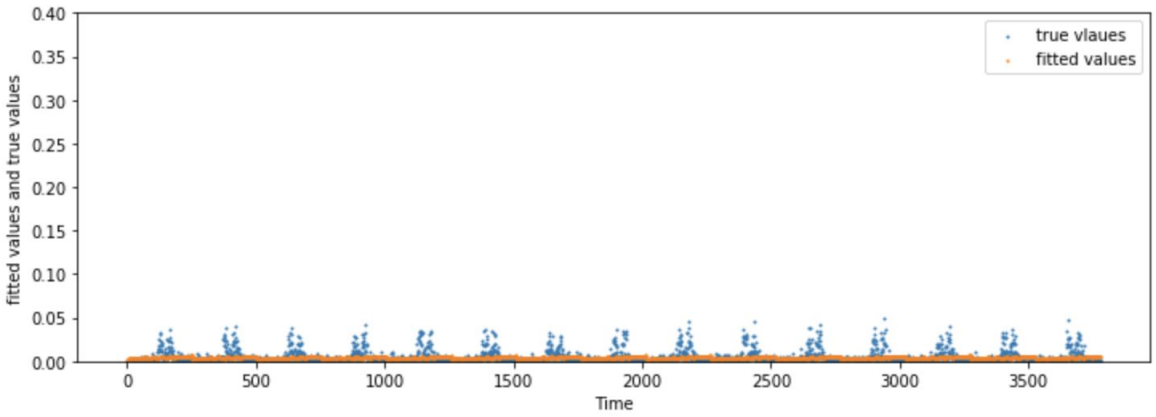


residuals vs. fitted values

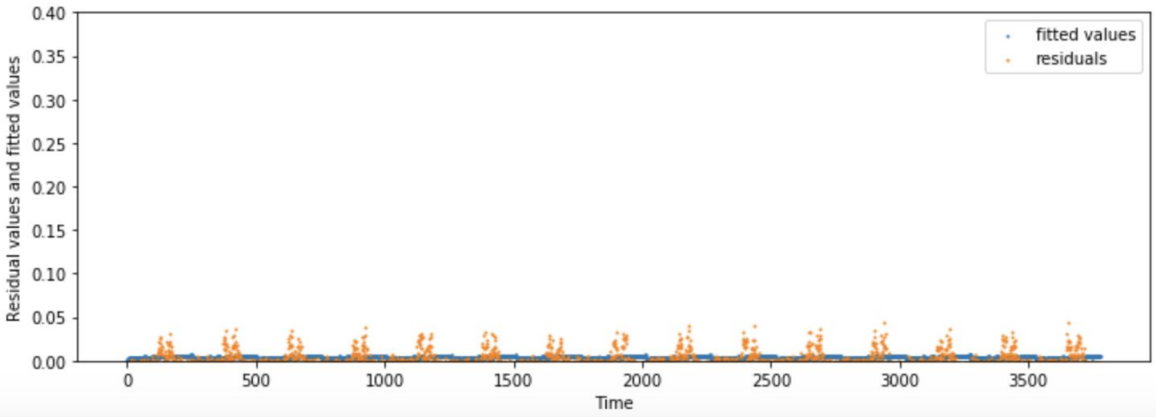


workflow_ID 4
Test RMSE with cross validation: 0.0072609
Train RMSE with cross validation: 0.0072439

Fitted values vs. true values

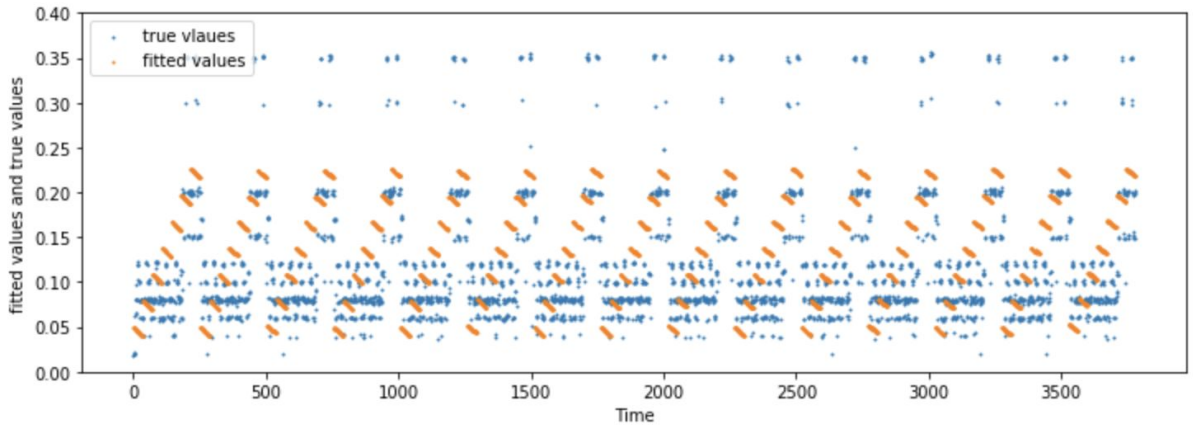


residuals vs. fitted values

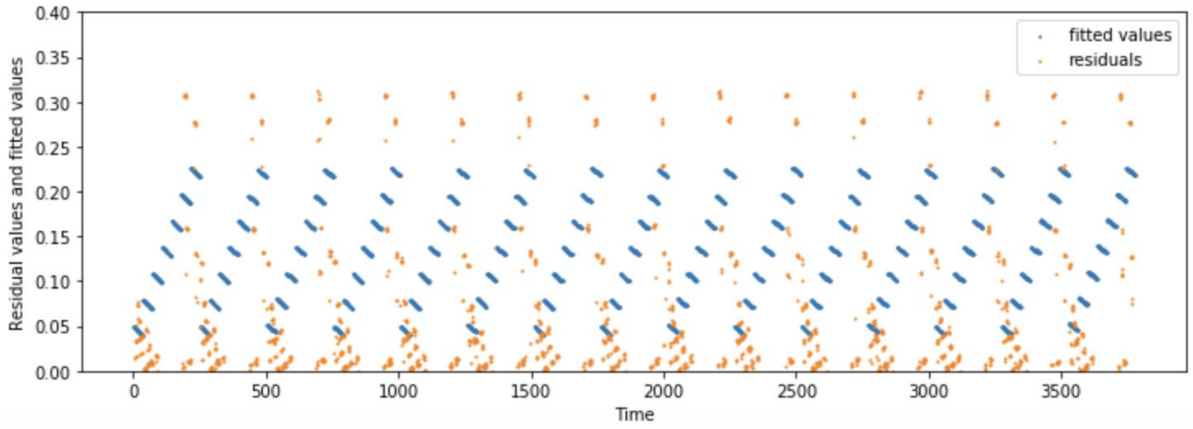


workflow_ID 5
Test RMSE with cross validation: 0.0859906
Train RMSE with cross validation: 0.0859219

Fitted values vs. true values



residuals vs. fitted values

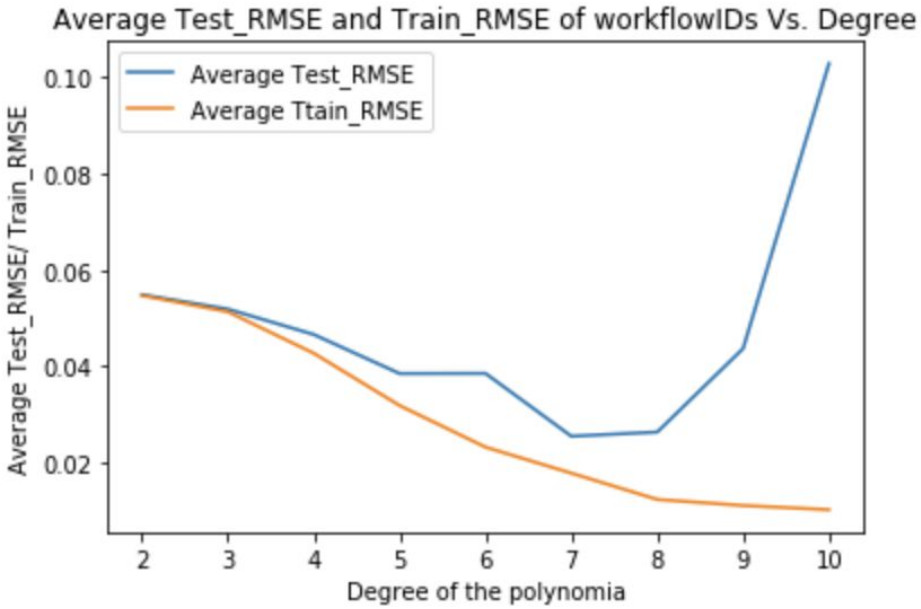


2.4.1.2 Conclusion

By compared these RMSE for each workflow with the RMSE with the whole dataset, we can find that except the RMSE of workflow2 exceed the RMSE of the whole dataset, others' are all lower than the whole dataset. The performance does improve in some workflow.

2.4.2.1 Polynomial Function

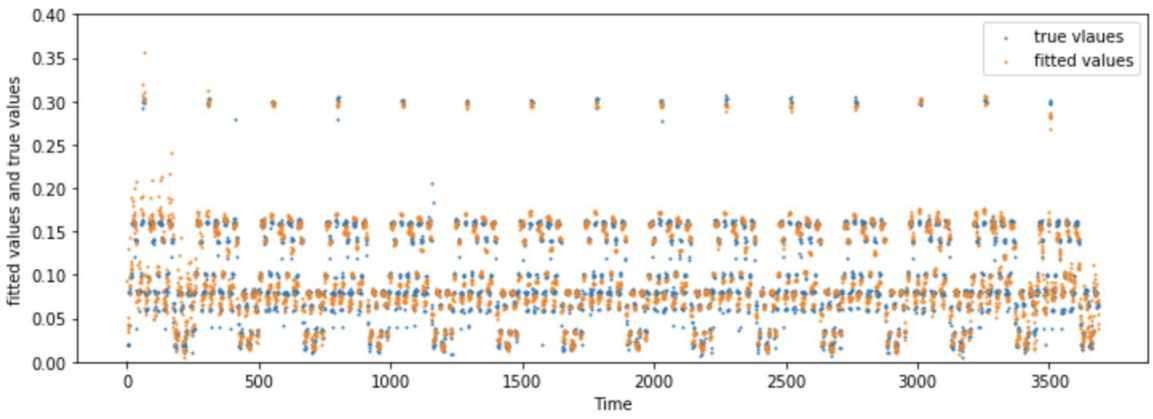
Next, we will use Polynomial function to train and test the data in different workflow, and get the averaged RMSE in different degree. We set the degree from 2 to 10 and draw the averaged RMSE curve to find the best parameter and detect the threshold.



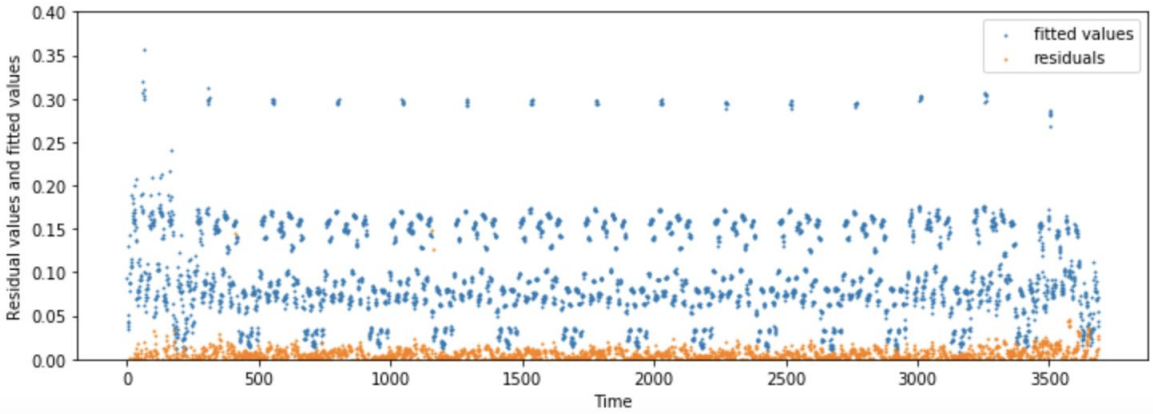
Next, we will use degree equals to 7 to test our model. Again, we will use a 10 fold cross validation to evaluate our results. Plot the average train and test RMSE of the trained model against the degree of the polynomial we use. The results are shown below:

workflow_ID 1
Test RMSE with cross validation: 0.0120145
Train RMSE with cross validation: 0.0091111

Fitted values vs. true values

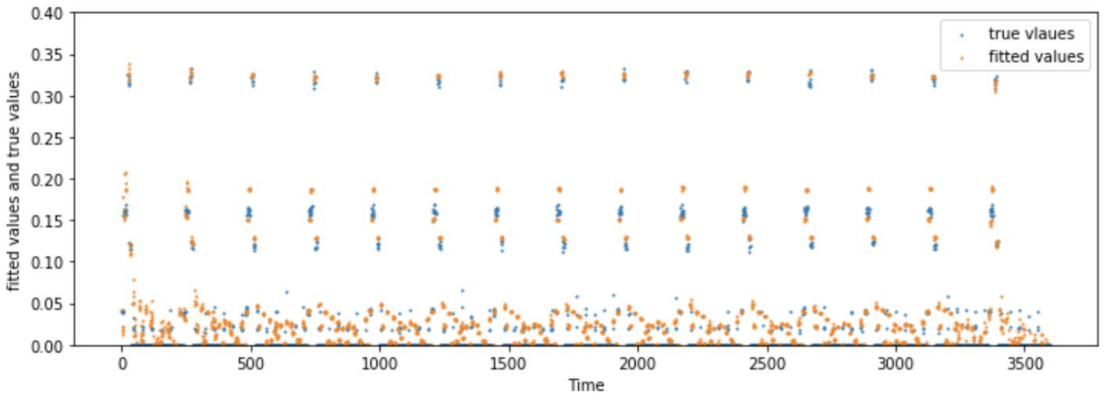


residuals vs. fitted values

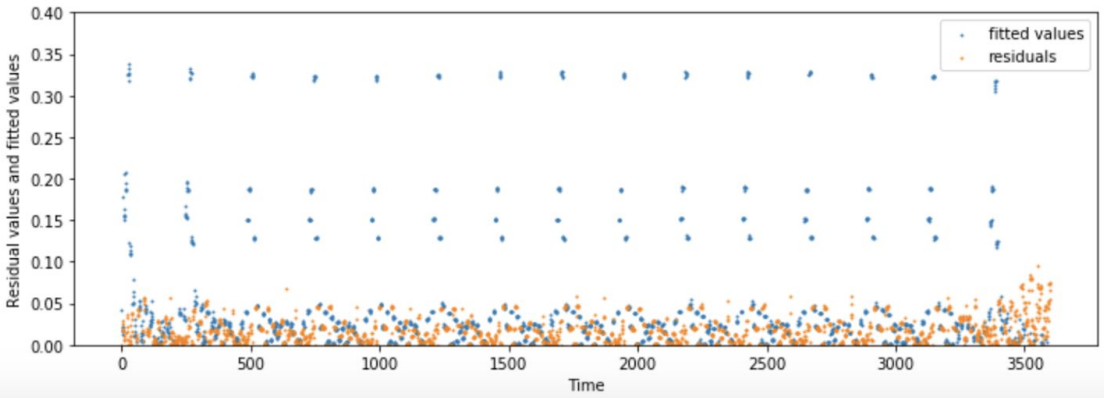


workflow_ID 2
Test RMSE with cross validation: 0.0226095
Train RMSE with cross validation: 0.0207529

Fitted values vs. true values

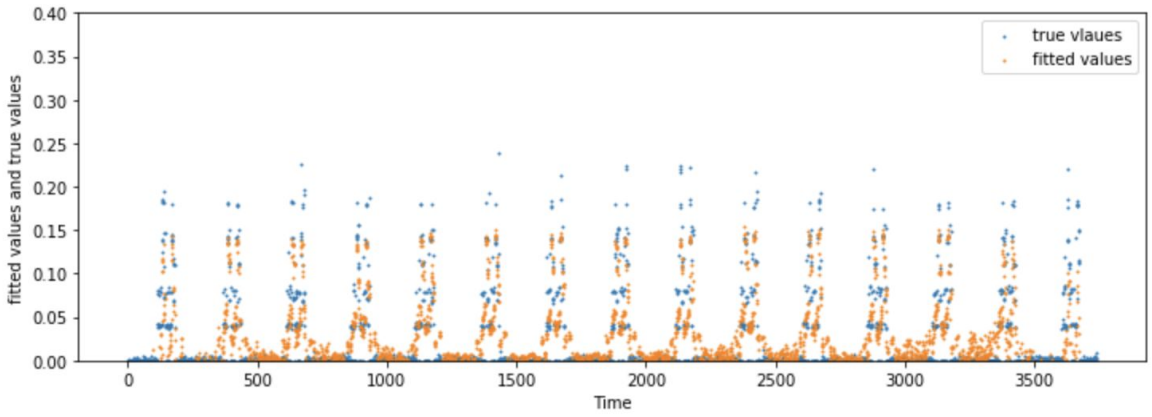


residuals vs. fitted values

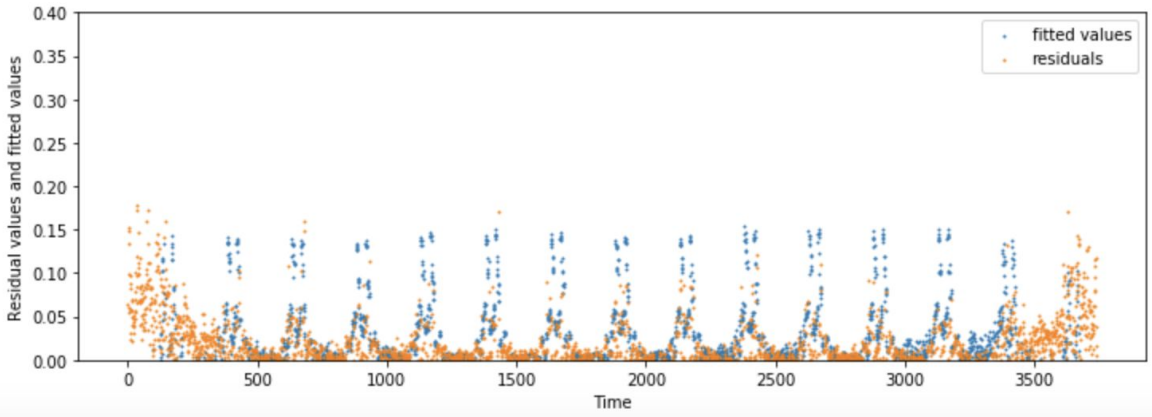


workflow_ID 3
Test RMSE with cross validation: 0.0313264
Train RMSE with cross validation: 0.0209678

```
Fitted values vs. true values
```

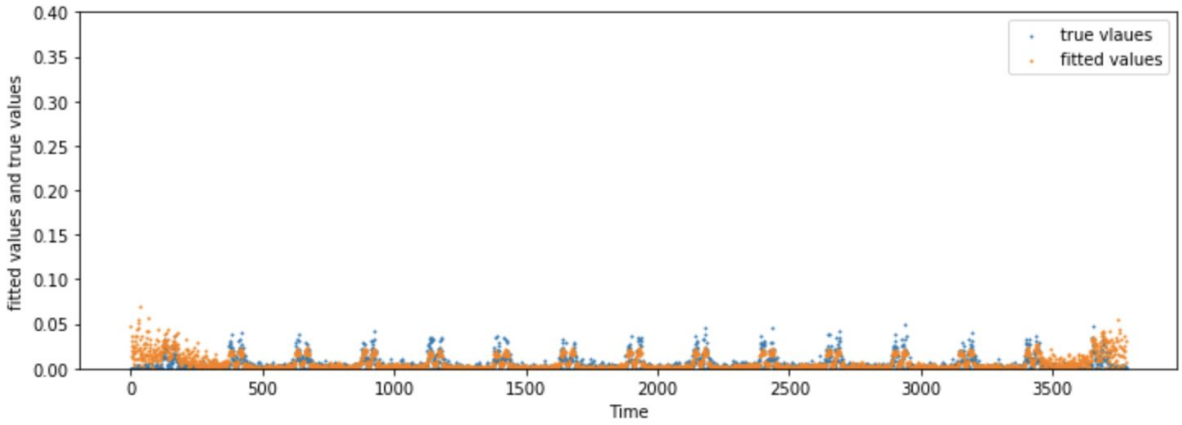



```
residuals vs. fitted values
```

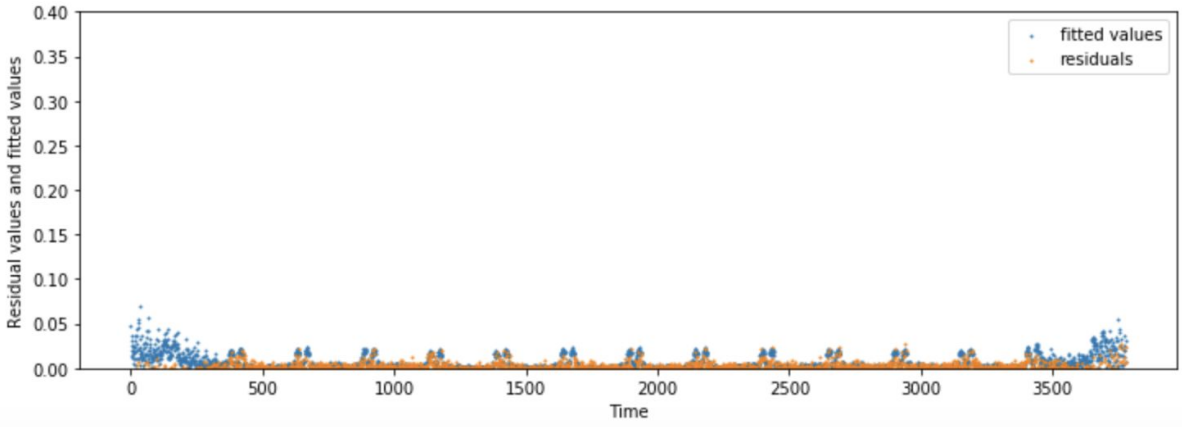


workflow_ID 4
Test RMSE with cross validation: 0.0075901
Train RMSE with cross validation: 0.0044999

```
Fitted values vs. true values
```

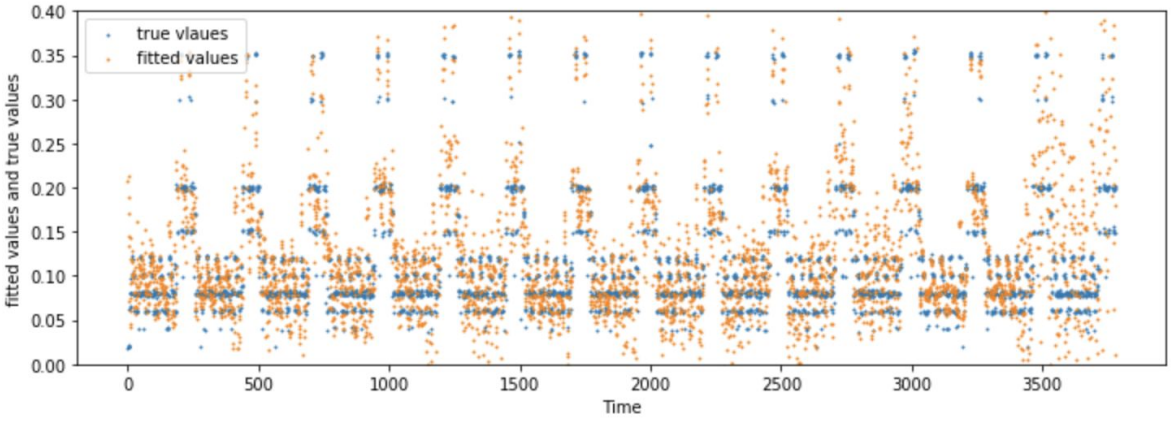



```
residuals vs. fitted values
```

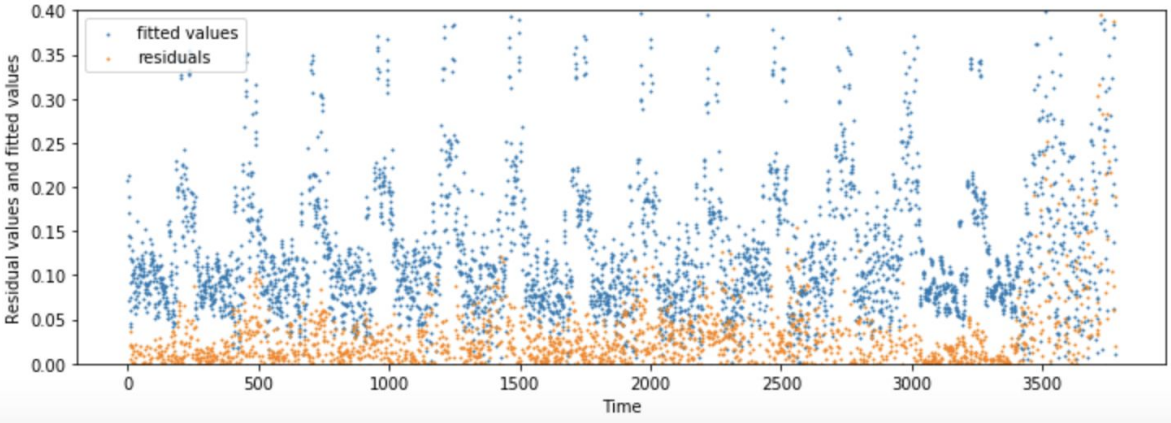


workflow_ID 5
Test RMSE with cross validation: 0.0537094
Train RMSE with cross validation: 0.0335749

Fitted values vs. true values



residuals vs. fitted values



2.4.2.2 Conclusion

From this pic we can observe that when the degree of the polynomial reaches 7, we can get minimum RMSE and if we continue increasing the degree, the Test RMSE will increase because of overfitting.

In cross validation, the model will be less prone to overfitting because a part of the training data itself is used for model estimation. The error calculated using multiple folds of the data will provide a more conservative result. Therefore, Cross Validation provides a more unbiased measure of the error and controls the overfitting . Furthermore, Cross Validation can be applied on several different methods. The least

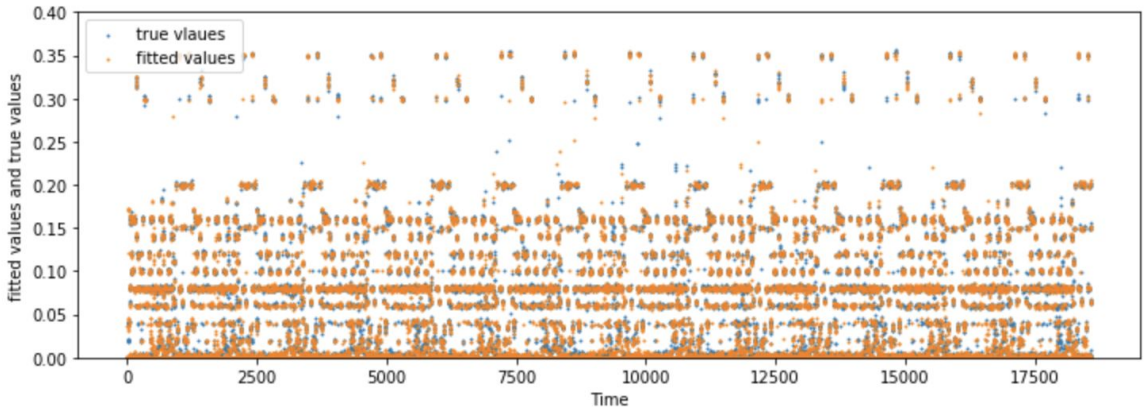
cross validation error can be chosen in these model, which reduce the complexity of the model.

2.5 K-NN Regression

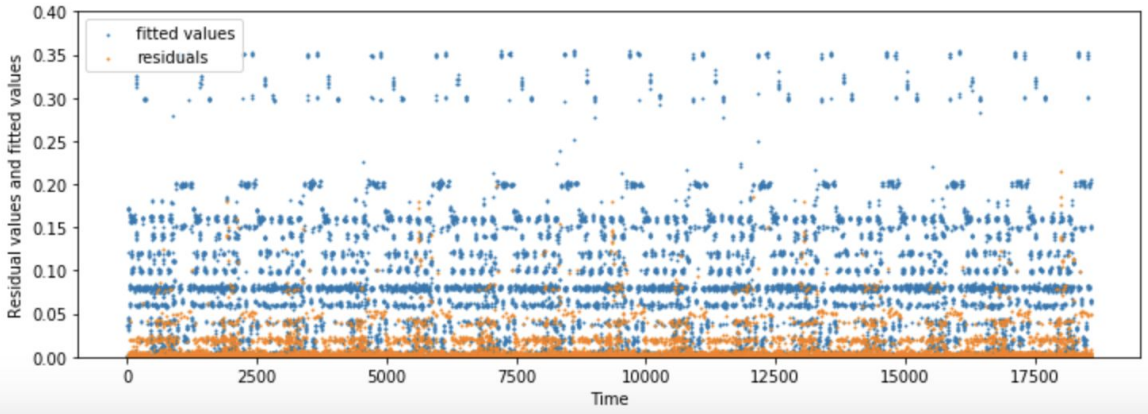
In this part, we will implement K-NN regression to predict. We choose the parameter k from 1 to 5 and get RMSE for each k. By analsize the RMSE we get, we find that when k = 1, we can get the best performance among these different k.

k = 1
Test RMSE with cross validation: 0.0201658

Fitted values vs. true values

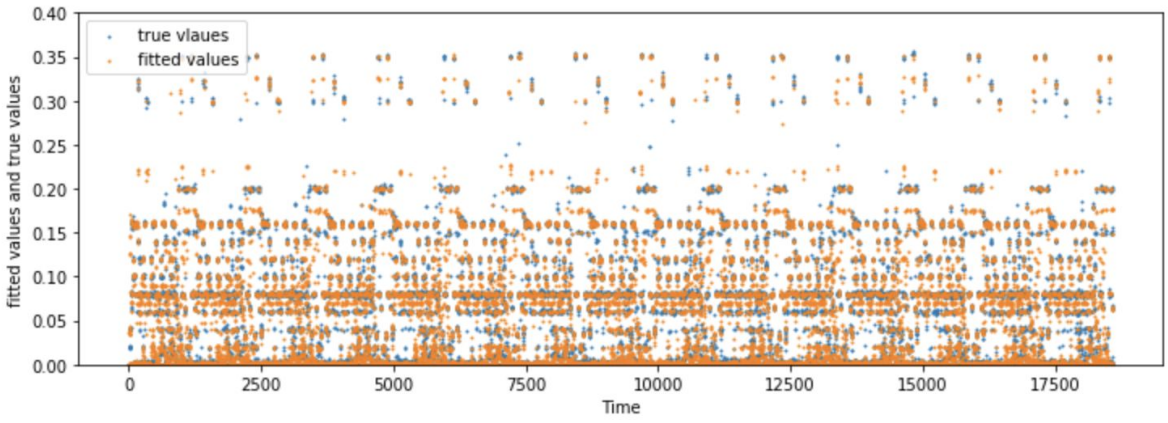


residuals vs. fitted values

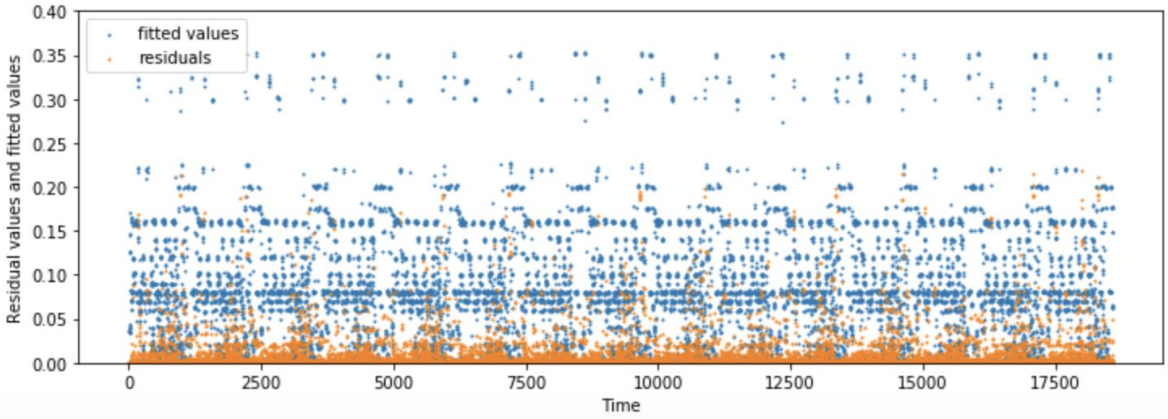


k = 2
Test RMSE with cross validation: 0.0343122
Train RMSE with cross validation: 0.0289562

Fitted values vs. true values

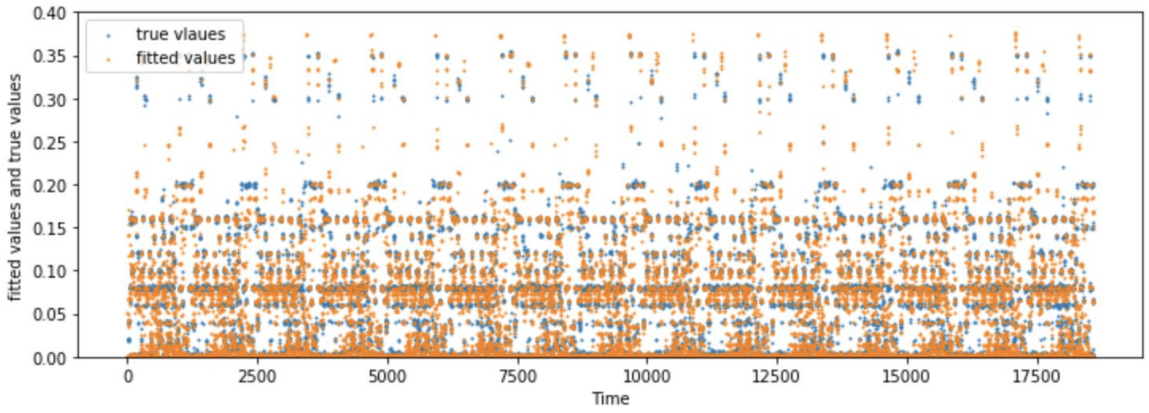


residuals vs. fitted values

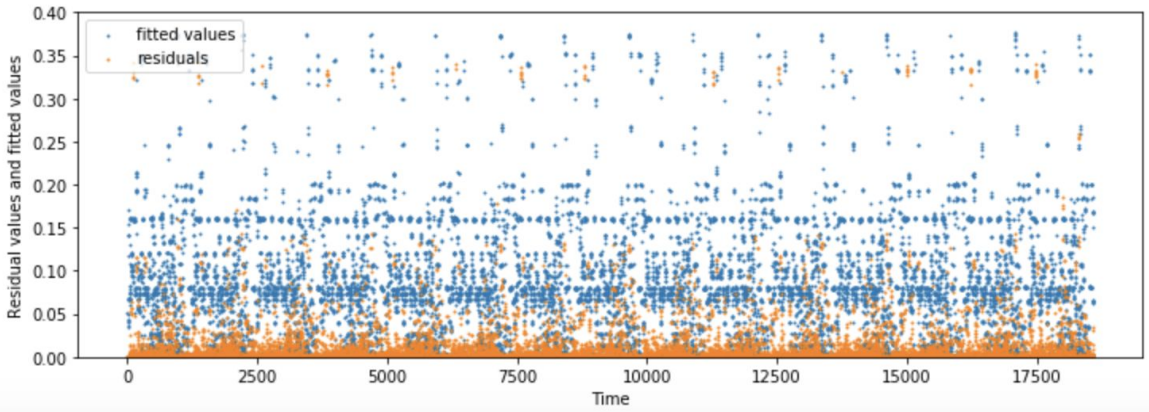


k = 3
Test RMSE with cross validation: 0.0363105
Train RMSE with cross validation: 0.0299732

Fitted values vs. true values

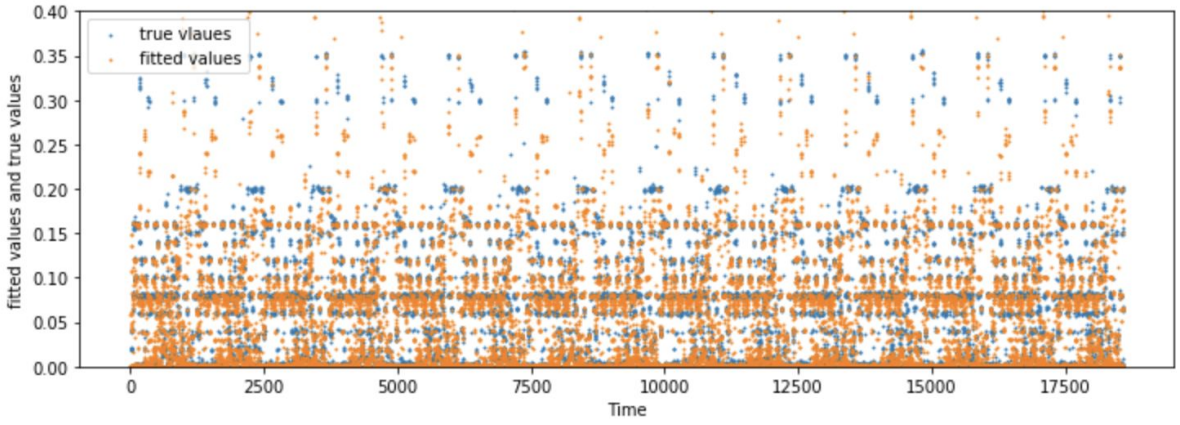


residuals vs. fitted values

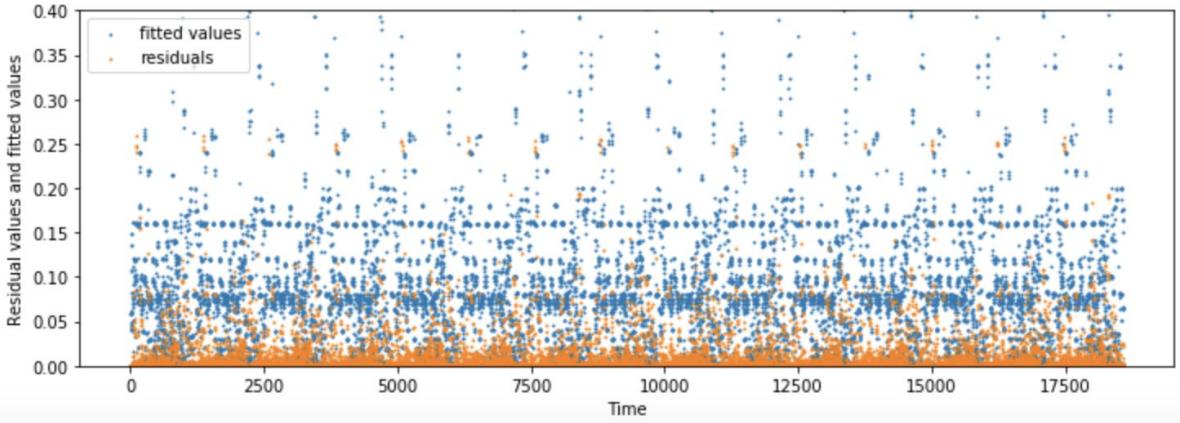


k = 4
Test RMSE with cross validation: 0.0374536
Train RMSE with cross validation: 0.0284126

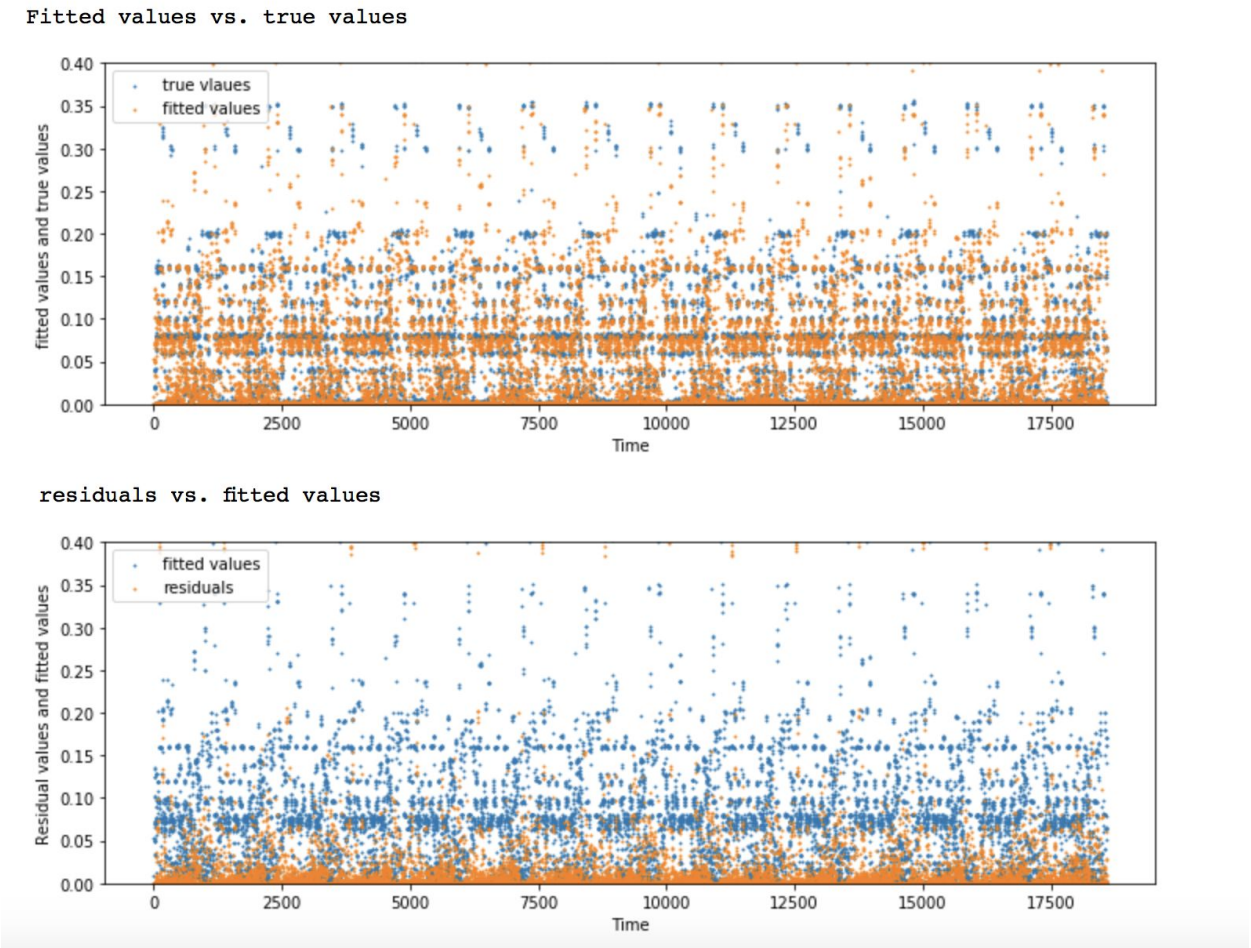
```
Fitted values vs. true values
```




```
residuals vs. fitted values
```



k = 5
Test RMSE with cross validation: 0.0433936
Train RMSE with cross validation: 0.0269626



Problem 3

Conclusion:

In this project, we use 5 model regression analysis. They are linear regression model (with different regularizers), random forest, NN, KNN and polynomial regression when we predict the backup size for each workflow.

Random forest model, polynomial regression model and KNN regression can deal with numerical or scalar encoding categorical variables without much one hot encoding (or totally without one hot encoding). Besides, the linear regression and NN

regression model can better handle sparse features, in other words, the categorical variables that applied one hot encoding.

All the best performances (smallest RMSE and bag of error) are reported in previous part respectively. And the conclusion and findings are also reported in previous part respectively.

From the results we get in this project and from the attributes of these methods, We find that:

The NN regression model is the best model at handling categorical features for its smallest RMSE in best performance. And the polynomial regression is best at handling sparse features. This can be seen because when we predict the backup size separately for each workflow, this model also performs very well.

Take all part into consideration, the NN model performs the best in this case.