Project 4: Regression Analysis

Name: Jianxiong Wang, Yijun Wu, Yanzhao Wang, Yutong Sun

Date: 2019.03.06

Dataset 1

1. Load the dataset

(a) For the first twenty-day period (x-axis unit is day number) plot the backup sizes for all workows (color coded on the y-axis).

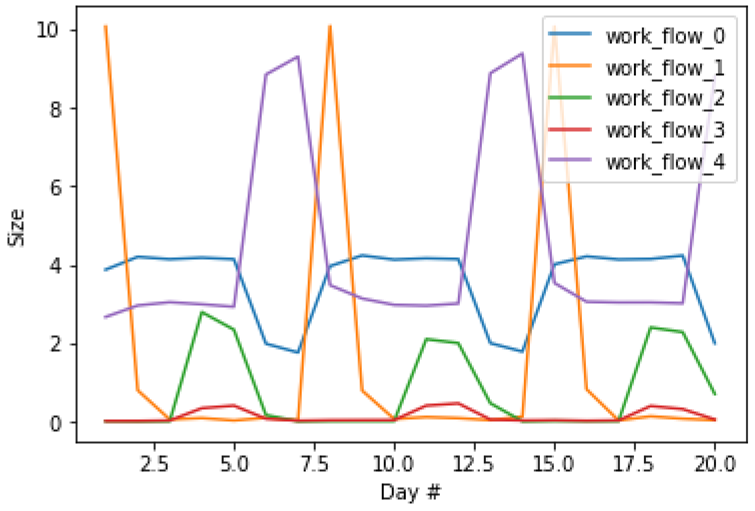


Figure 1. Backup Sizes for All Workows in First 20 Days

(b) Do the same plot for the rst 105-day period.

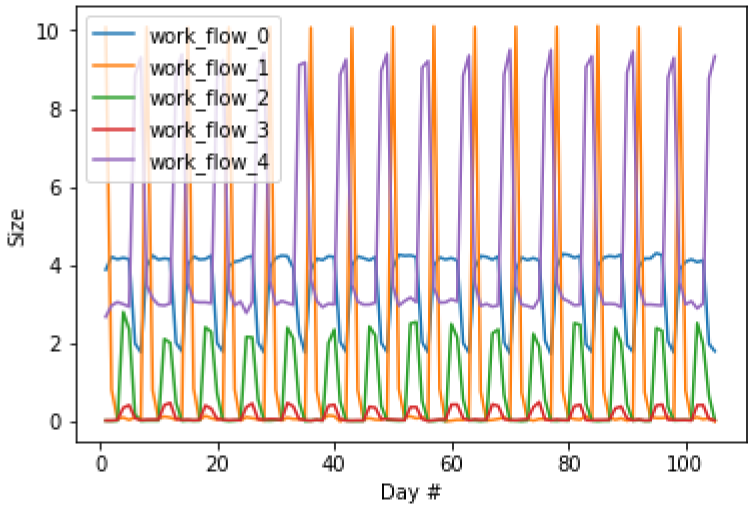


Figure 2. Backup Sizes for All Workows in First 105 Days

(c) Can you identify any repeating patterns?

From the two plots, It is not difficult to conclude from the two plots that the change of backup size, to some extent, shows periodicity. In other words, both 20 days and 105 dyas data have a repeating pattern with period of 7 days. Therefore, it is reasonable to assume that the feature Week number would not make a huge influence on the backup size.

2. Predict the backup size of a file given the other attributes

(a) Linear Regression

i. Plot fitted values against true values as scatter plots

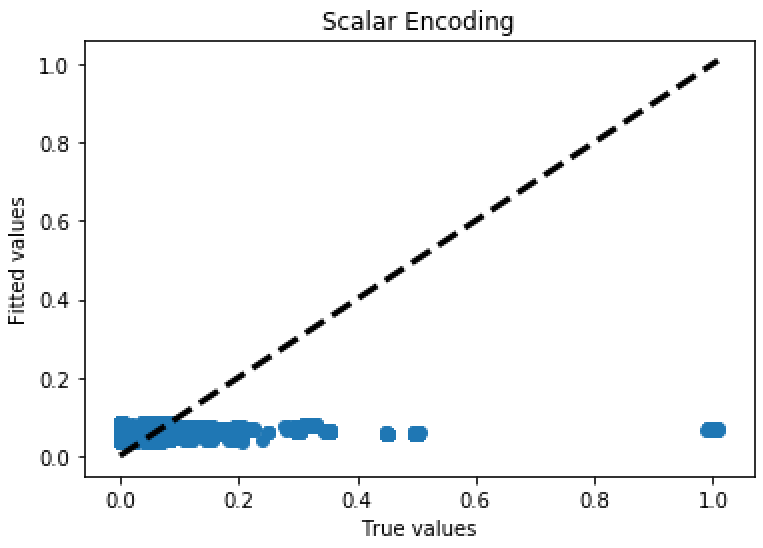


Figure 3. Scalar Encoding Fitted Values vs. True Values Using Linear Regression

ii. Plot residuals versus fitted values as scatter plots

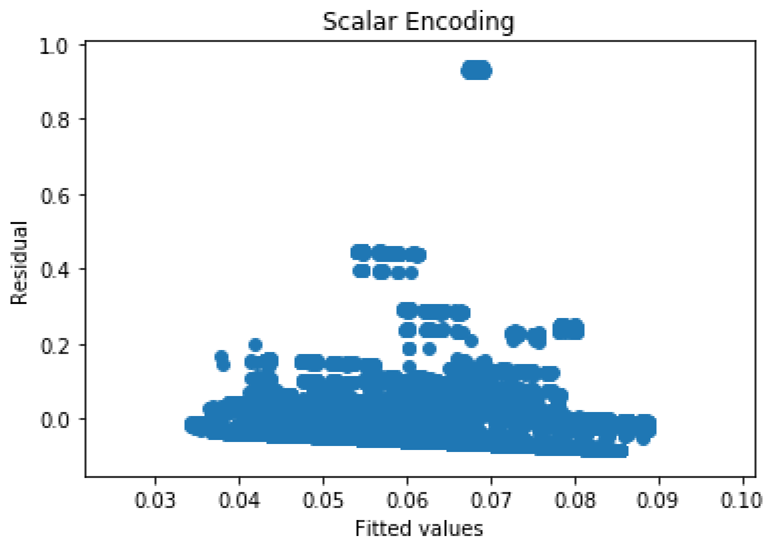


Figure 4. Scalar Encoding Residuals vs. Fitted Values Using Linear Regression

Using linear regression model to fit the data, the average training RMSE is 0.1036, and the average test RMSE is 0.1036.

In this part, we used linear_model in sklearn.linear_model.LinearRegression object to generate a linear regression model and fit it with the backup size data. The fitted value vs. true value scalar encoding plot is shown in Figure 3, the residual value vs. fitted value scalar encoding plot is shown in Figure 4. Based on the above 2 plots, we can conclude that the linear regression model does not fit the training data very well since Fitted vs. True value plot is far from the diagonal dotted line, which indicates a good model, and the Residual vs. Fitted value plot has a lot of points above 0 residual and the residuals and the fitted values are correlated.

(b) Random Forest Regression Model

i. 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.

After 10 fold cross validation, the average Training RMSE we got is 0.0604, the average Test RMSE is 0.0601, and the average Out Of Bag error is 0.3387. The performance of the model can be indicated below in Figure 5 and Figure 6.

The random forest regression model has better performance for the dataset compared to linear regression. In Figure 5, the points in fitted value vs. true value plot are more distributed around the diagonal line, and the points in residual vs. fitted value plot are equally distributed around 0 residual line.

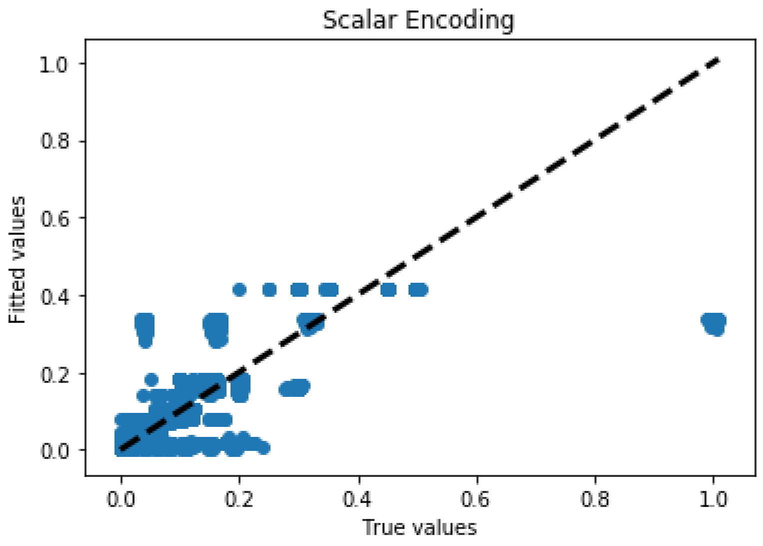


Figure 5. Scalar Encoding Fitted Values vs. True Values Using Random Forest Regression

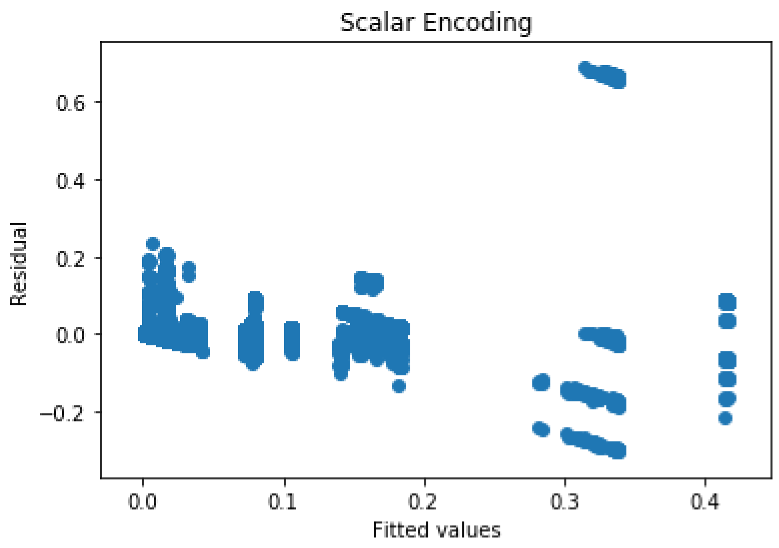


Figure 6. Scalar Encoding Residuals vs. Fitted Values Using Random Forest Regression

ii. Sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5, plot

1) out-of-bag error (y axis) against number of trees (x axis)

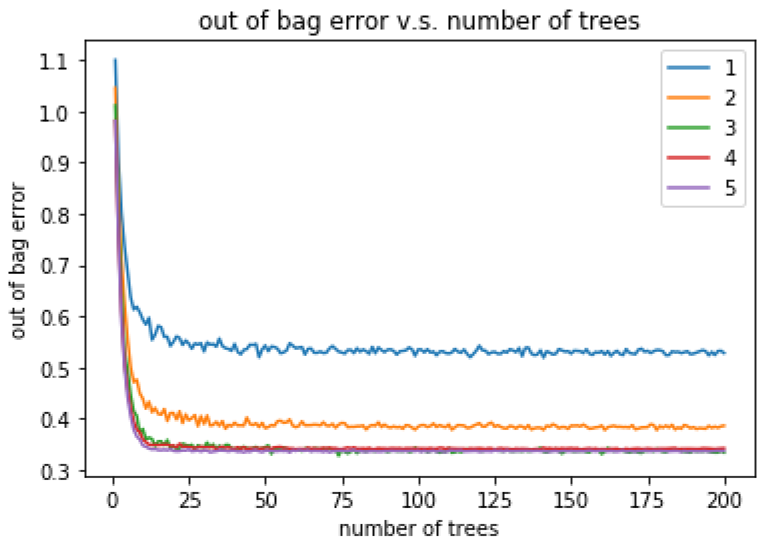


Figure 7. Out of bag error vs. number of trees for different maximum number of features

The maximum number of features	Number of trees at out-of-bag error min	out-of-bag error min
1	140	0.5193763654594519
2	177	0.37815031657124176
3	73	0.3280689933848011
4	136	0.33960185241030394
5	57	0.33468171396527785

Table 1. Result from out-of-bag error vs. number of trees

2) average test-RMSE (y axis) against number of trees (x axis)

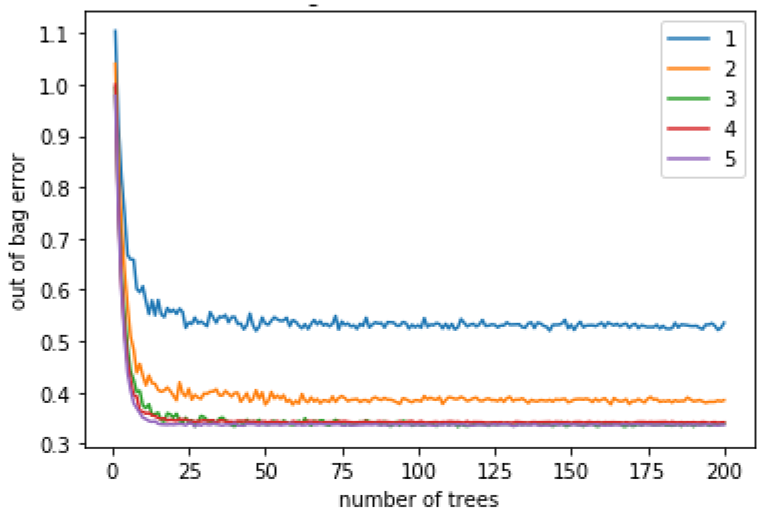


Figure 8. Average test-RMSE vs. number of trees for different maximum number of features

maximum number of features	Number of trees at test RMSE min	Test RMSE min
1	46	0.5200296069804097
2	58	0.3762288421414676
3	112	0.33328895338861453

4	57	0.33914802124810317
5	26	0.33429014936996476

Table 2. Result from average test RMSE vs. number of trees

Based on the results above in Figure 7 and Figure 8, maximum number of features = 3, 4, 5 have similar out-of-bag error and test RMSE results, which are better than maximum number of features = 2 and maximum number of features = 1. If we want to use minimum number of trees, we can choose to use maximum number of features = 5 which has its min out-of-bag error and test RMSE with smallest number of trees. Therefore, we can build the model using maximum number of features = 5.

iii. 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?

We pick depth to be the other parameter to experiment on, and the out-of-bag error vs. depth plot and average test-RMSE vs. depth plot are as shown in Figure 9 and Figure 10.

1) **Plot out-of-bag error (y axis) against depth (x axis)**

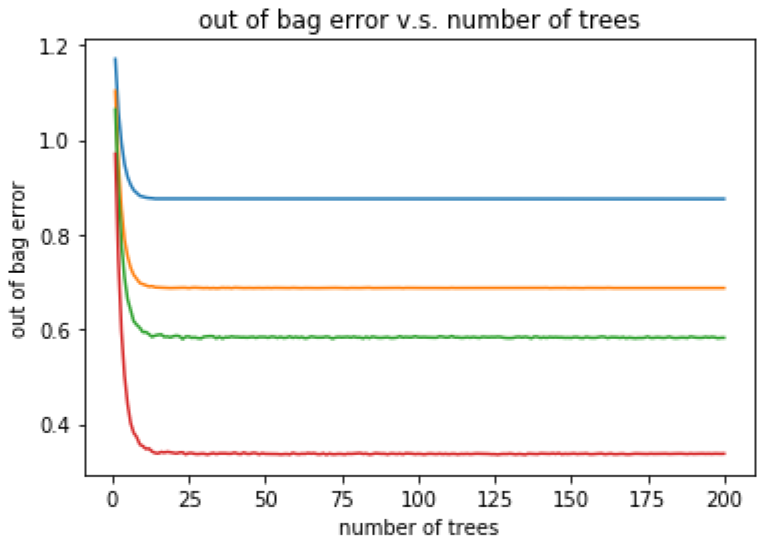


Figure 9. Out of bag error vs. number of trees for different depth

Depth	Depth at out-of-bag error min	out-of-bag error min
1	36	0.8752731419524518
2	33	0.6867492912485055

3	22	0.5795276770593109
4	69	0.33536788269713186

Table 3. Result from out-of-bag error vs. depth

2) Plot average test-RMSE (y axis) against depth (x axis)

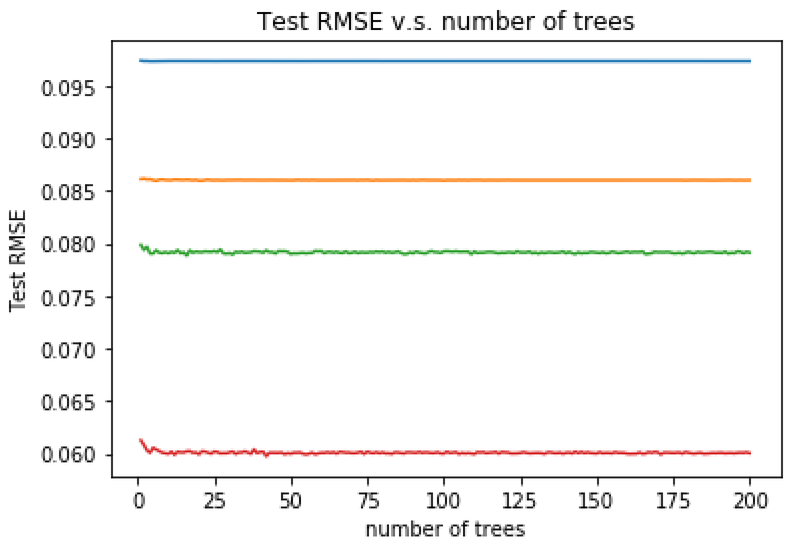


Figure 10. Average test-RMSE vs. number of trees for different depth

Depth	Depth at test RMSE min	Test RMSE min
1	3	0.09731610642189377
2	5	0.08594926512522723
3	15	0.07882772633904289
4	41	0.059747384279583815

Table 4. Result from average test RMSE vs. depth

As we can notice in Table 3 and Table 4, depth = 4 has the best out-of-bag error and test RMSE results. Overall, we the parameters to achieve the best performance are maximum number of features = 5, depth = 4.

Iv. Report the feature importances you got from the best random forest regression you find.

In this part, we used sklearn.ensemble.RandomForestRegressor object's attribute feature_importances_ to get the importance data of each feature. The feature importances ranking from the best random forest regression is as below:

1. feature 4 (0.313193)
2. feature 1 (0.277288)
3. feature 3 (0.255949)
4. feature 2 (0.153563)
5. feature 0 (0.000007)

Therefore, we can say that feature 4 is the most important feature, and the feature 9 is the least important feature for the best random forest regression.

v. 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?

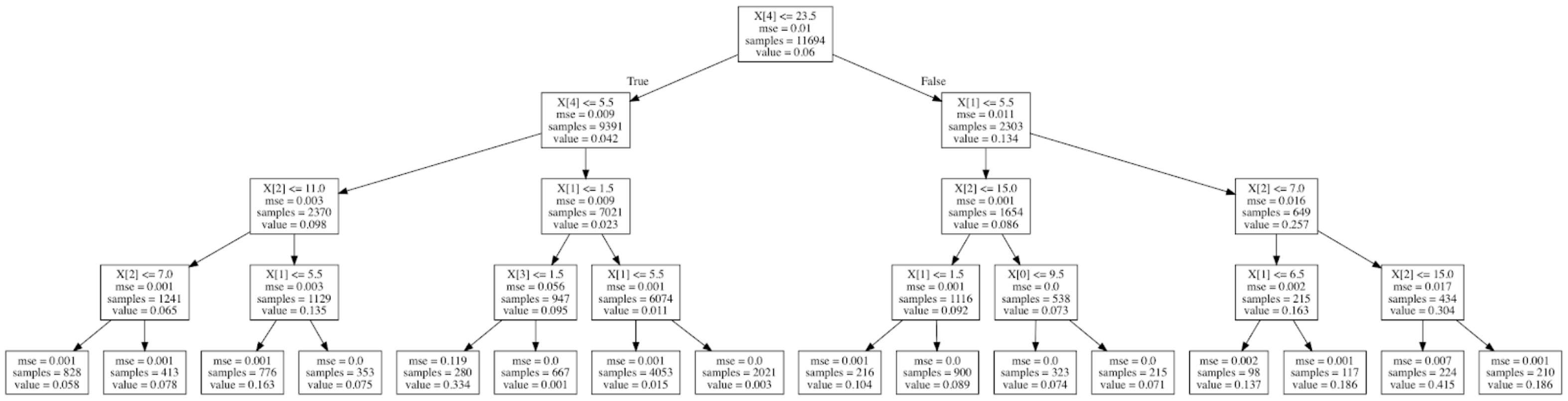


Figure 11. Decision Tree in the Best Random Forest

We first used export_graphviz to get the first tree in the best random forest and output it into DecisionTree.dot file. Then, we converted it into a .png file as shown in Figure 11. The root node of this decision tree is the node which has the following values X[4] <= 23.5, mse = 0.01, samples = 11694, value = 0.06. The most important feature according to the feature importance reported in part iv is consistent with the decision tree since the root node is feature 4 <= 23.5 and feature 4 is the most important feature reported before.

(c) Neural Network Regression Model

Number of hidden units. Try [2, 5, 10, 50, 100, 150, 200,...,600].
Activation function (`relu', `logistic', `tanh')
Plot test-RMSE vs the number of hidden units for each activation function.
Report the best combination.

(1) ReLu activation function:

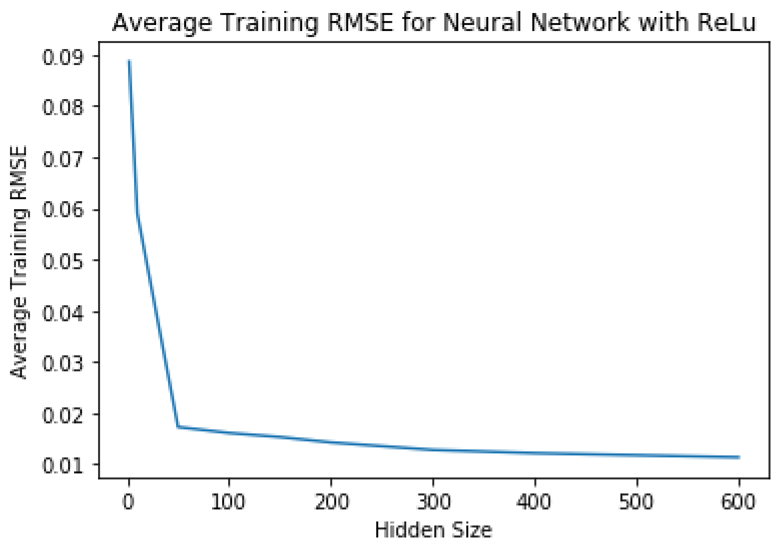


Figure 12. Average Training RMSE Using Neural Network Regression with ReLu

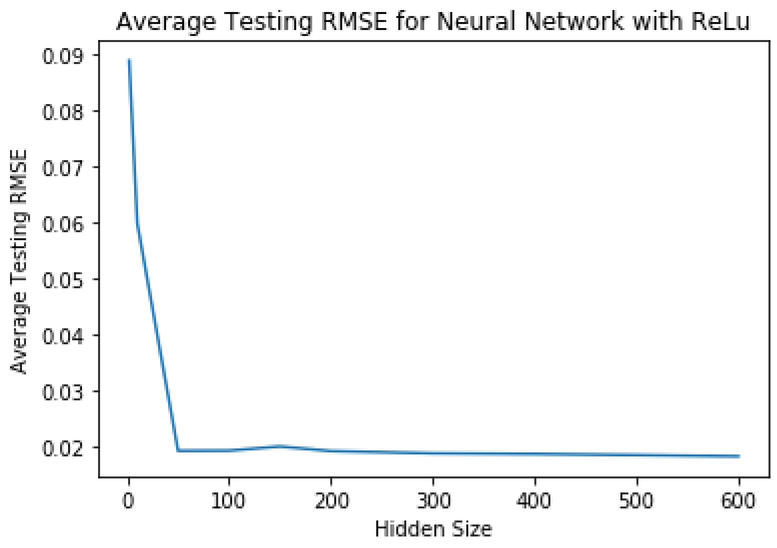


Figure 13. Average Testing RMSE Using Neural Network Regression with ReLu

The training and test RMSE plot have similar shape, so no overfitting happened. As shown in the Figure 12, the test RMSE starts to saturate at hidden size = 50 with a value of

0.019275735714761138, and the test RMSE reaches its best value, 0.01830380345361102, when hidden size is 600.

(2) Logistic activation function:

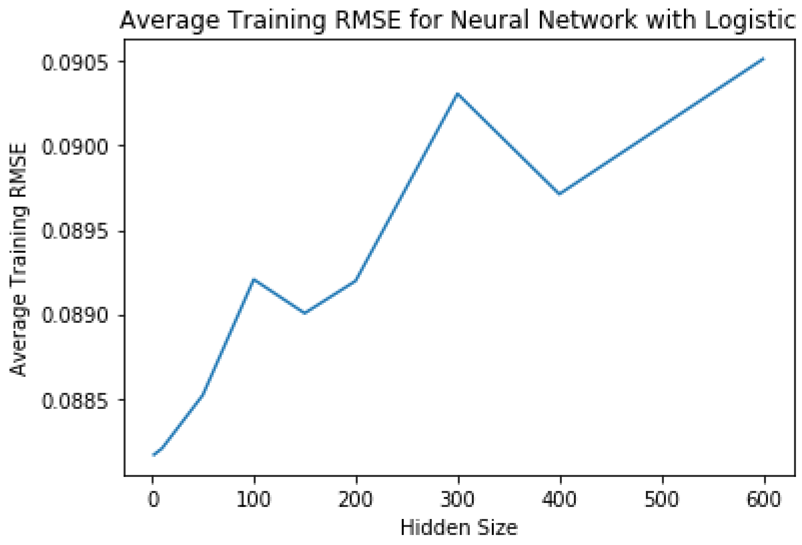


Figure 14. Average Training RMSE Using Neural Network Regression with Logistic

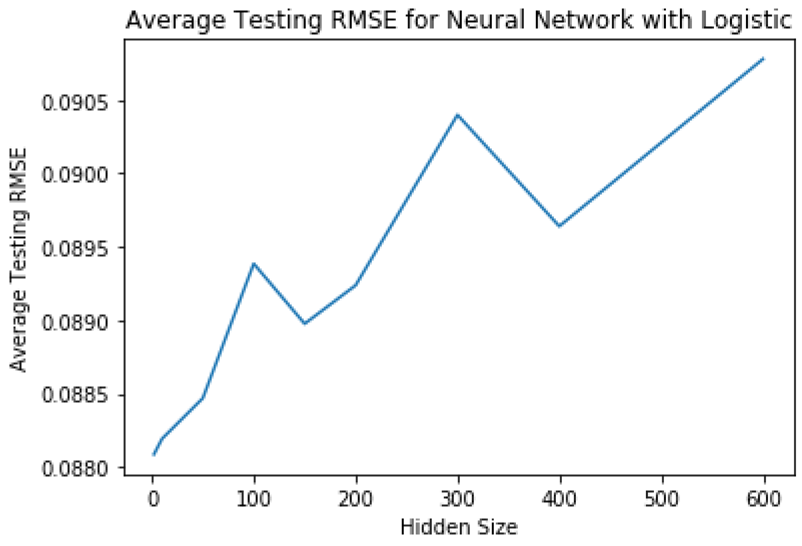


Figure 15. Average Testing RMSE Using Neural Network Regression with Logistic

For logistic function, the training and test RMSE plot have similar shape, so no overfitting happened. The test RMSE and training RMSE both reach their smallest values at hidden size = 2. The smallest test RMSE is 0.08808656059756455.

(3) tanh activation function:

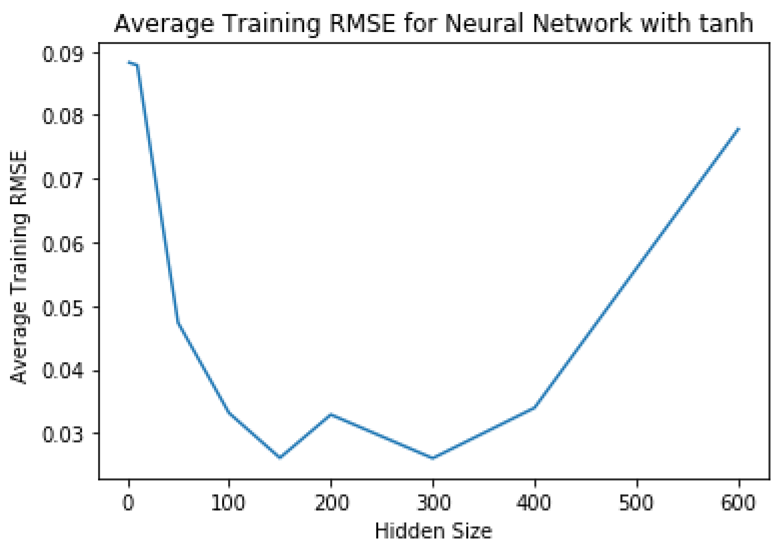


Figure 16. Average Training RMSE Using Neural Network Regression with tanh

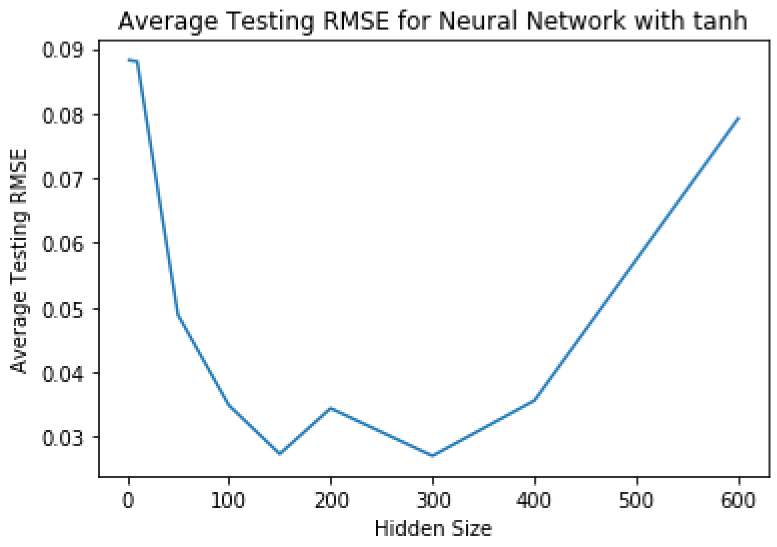


Figure 17. Average Testing RMSE Using Neural Network Regression with tanh

For tanh function, the training and test RMSE plot have similar shape. The smallest test RMSE is 0.027013943536851204 at hidden size = 300.

Therefore, it is easy to conclude that the Relu function has the best performance when hidden size is large. The best combination is Relu activation function with hidden size = 600, and this combination results in an average RMSE of 0.01830380345361102.

(d) Predict the Backup size for each of the workflow

i. Using linear regression model. Explain if the fit is improved?

Workflow 0:

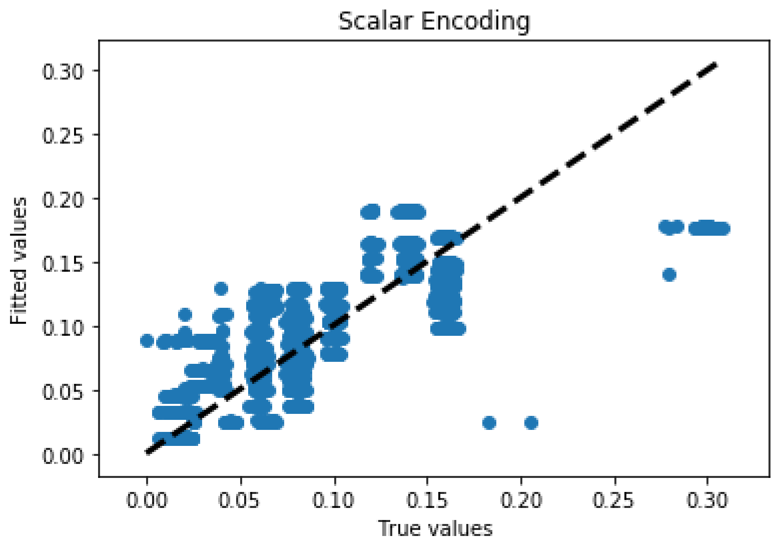


Figure 18. Scalar Encoding for Workflow 0 Fitted Values vs. True Values Using Linear Regression

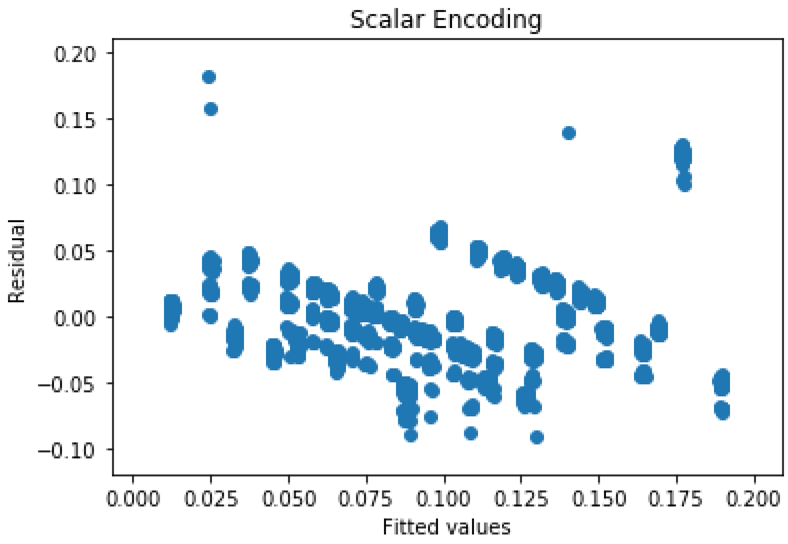


Figure 19. Scalar Encoding for Workflow 0 Residuals vs. True Values Using Linear Regression

For workflow 0, the model is improved significantly compared to the original linear regression model. In Figure 18, the data points are clustered around diagonal line, and the residual values are less correlated to fitted values as shown in Figure 19.

Workflow 1:

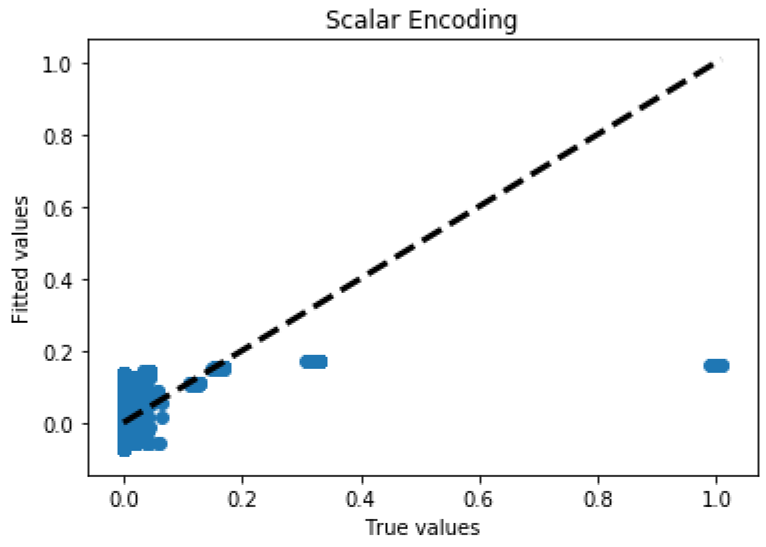


Figure 20. Scalar Encoding for Workflow 1 Fitted Values vs. True Values Using Linear Regression

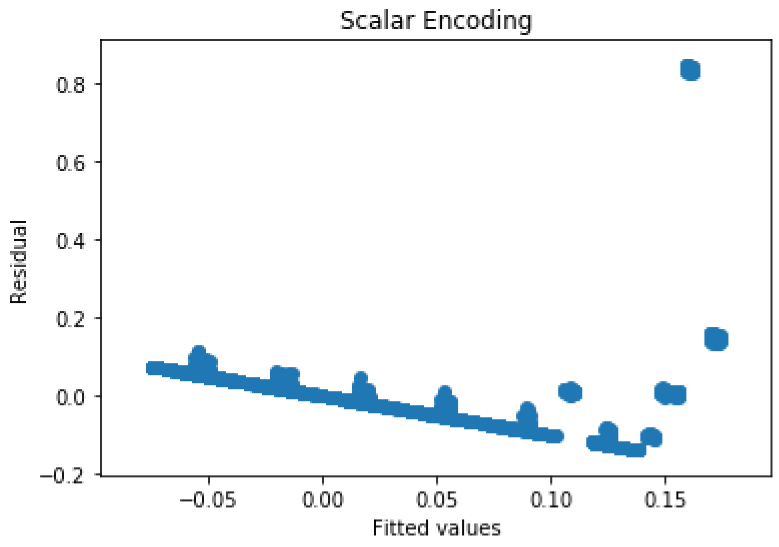


Figure 21. Scalar Encoding for Workflow 1 Residuals vs. True Values Using Linear Regression

Workflow 2:

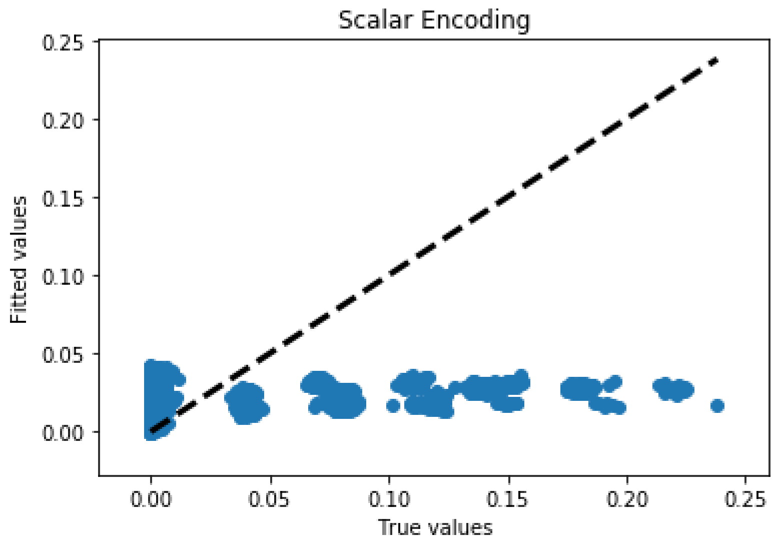


Figure 22. Scalar Encoding for Workflow 2 Fitted Values vs. True Values Using Linear Regression

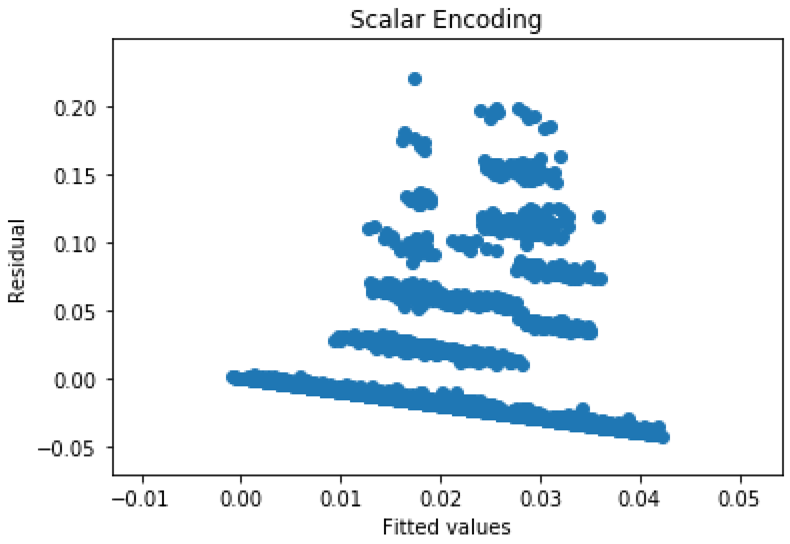


Figure 23. Scalar Encoding for Workflow 2 Residuals vs. True Values Using Linear Regression

Workflow 3:

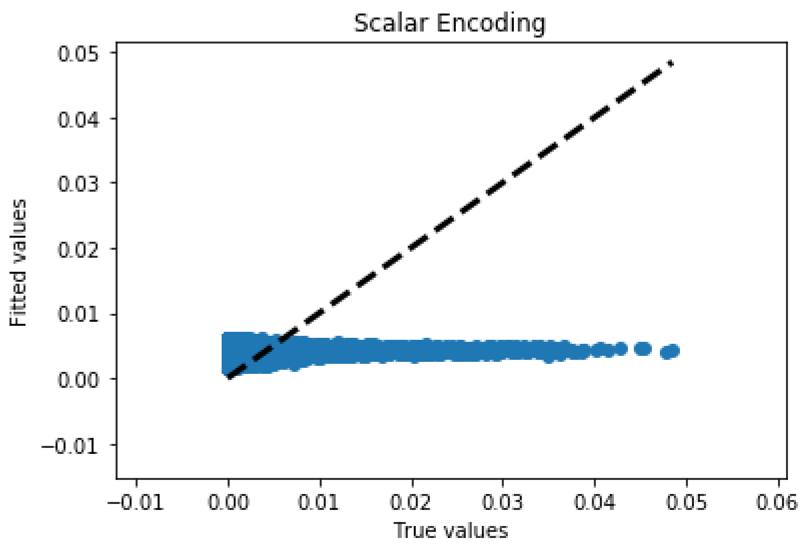


Figure 24. Scalar Encoding for Workflow 3 Fitted Values vs. True Values Using Linear Regression

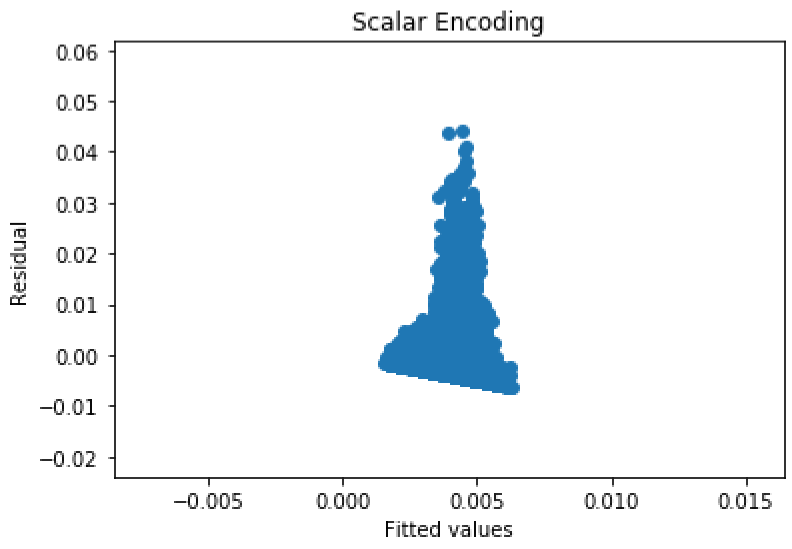


Figure 25. Scalar Encoding for Workflow 3 Residuals vs. True Values Using Linear Regression

Workflow 4:

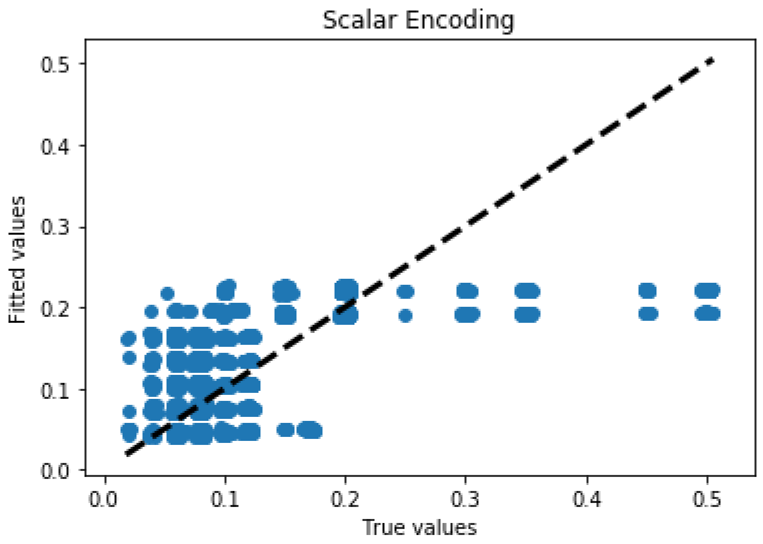


Figure 26. Scalar Encoding for Workflow 4 Fitted Values vs. True Values Using Linear Regression

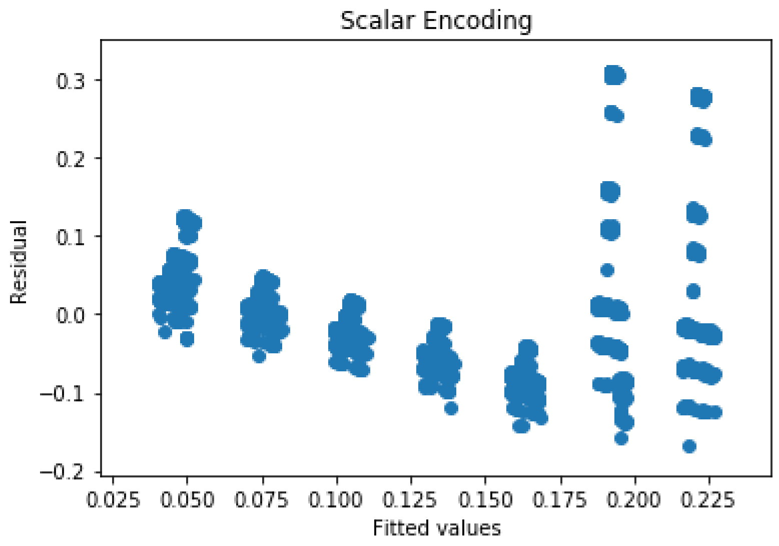


Figure 27. Scalar Encoding for Workflow 4 Residuals vs. True Values Using Linear Regression

Using linear regression only on workflow 0 improves the linear regression model's performance, but does not work for workflow 2, 3, 4. Workflow 1 can be slightly improved.

ii. Try fitting a more complex regression function to your data. Can you find a threshold on the degree of the fitted polynomial beyond which the generalization error of your model gets worse? Can you explain how cross validation helps controlling the complexity of your model?

The polynomial degrees we tried are poly_input = [1,2,3,4,5,6,7,8,9,10,11,12].

Workflow 0:

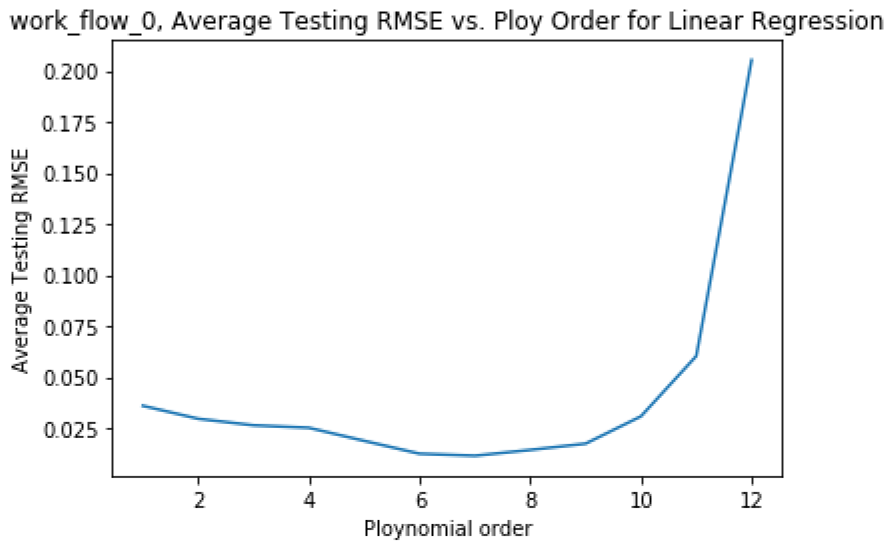


Figure 28. Average Testing RMSE vs. Polynomial Degree for Workflow 0

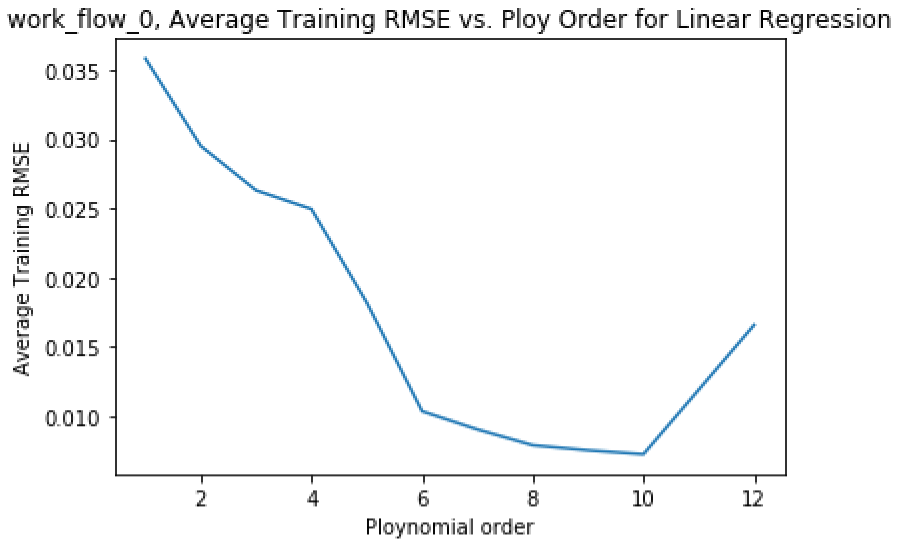


Figure 29. Average Training RMSE vs. Polynomial Degree for Workflow 0

Workflow 1:

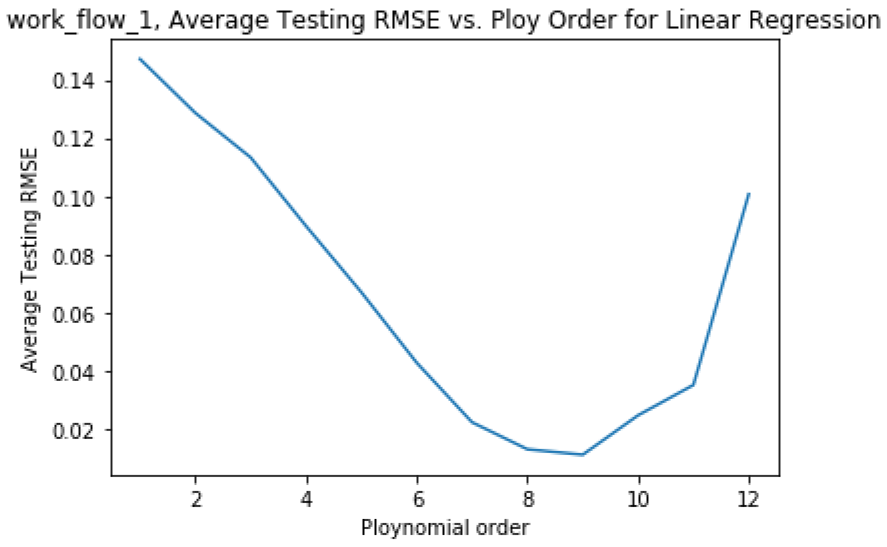


Figure 30. Average Testing RMSE vs. Polynomial Degree for Workflow 1

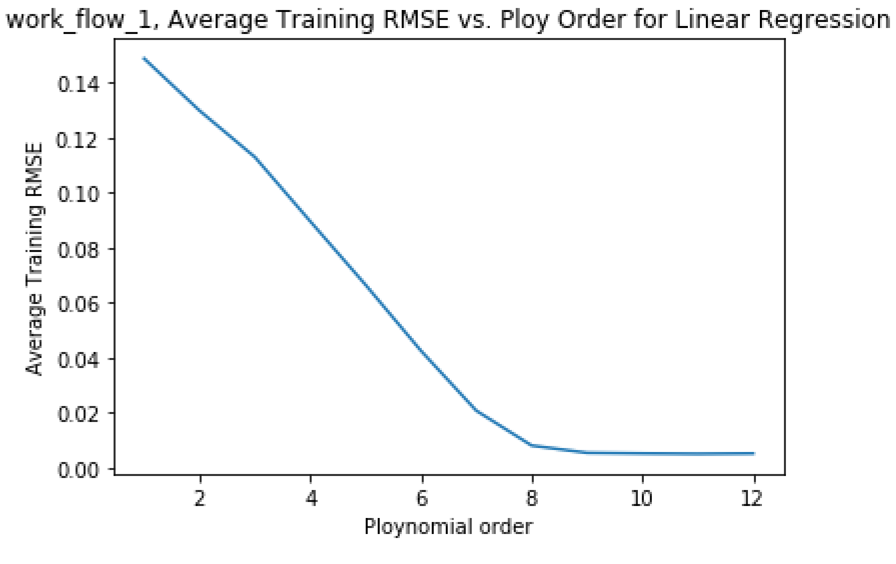


Figure 31. Average Training RMSE vs. Polynomial Degree for Workflow 1

Workflow 2:

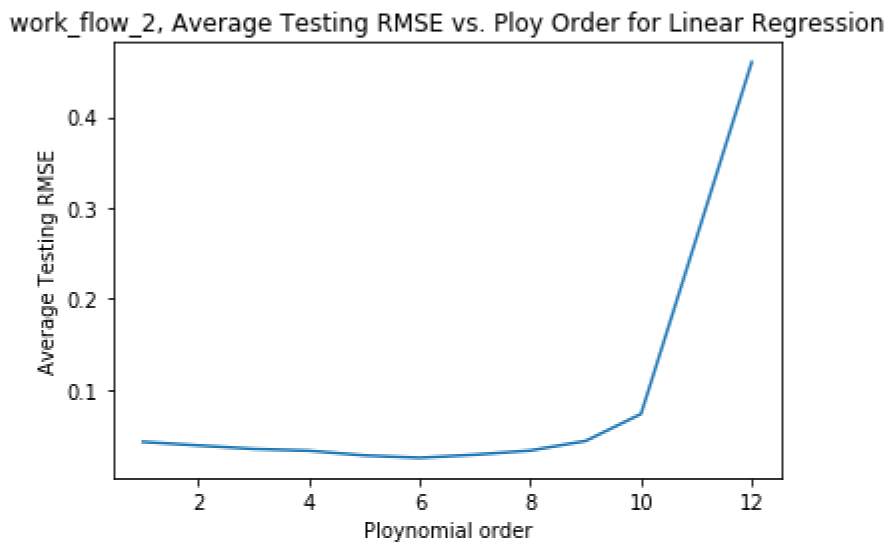


Figure 32. Average Testing RMSE vs. Polynomial Degree for Workflow 2

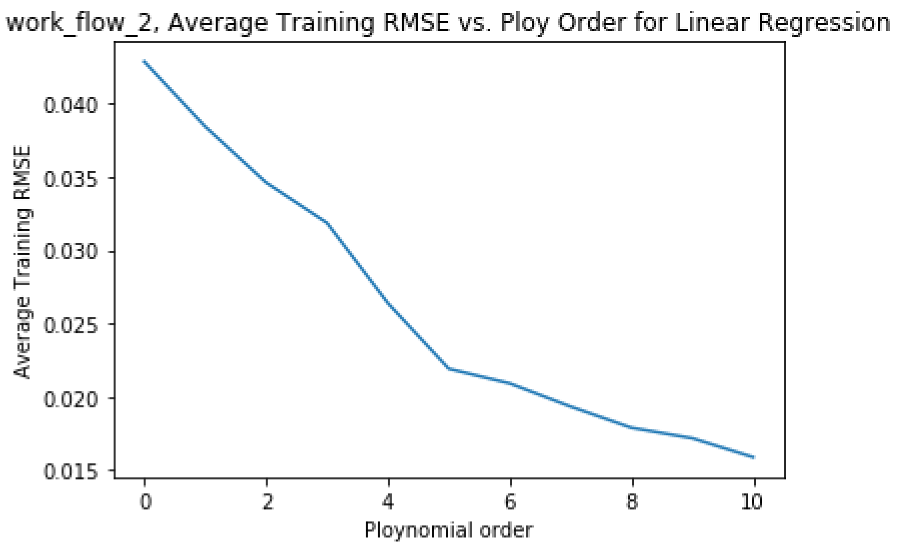


Figure 33. Average Training RMSE vs. Polynomial Degree for Workflow 2

Workflow 3:

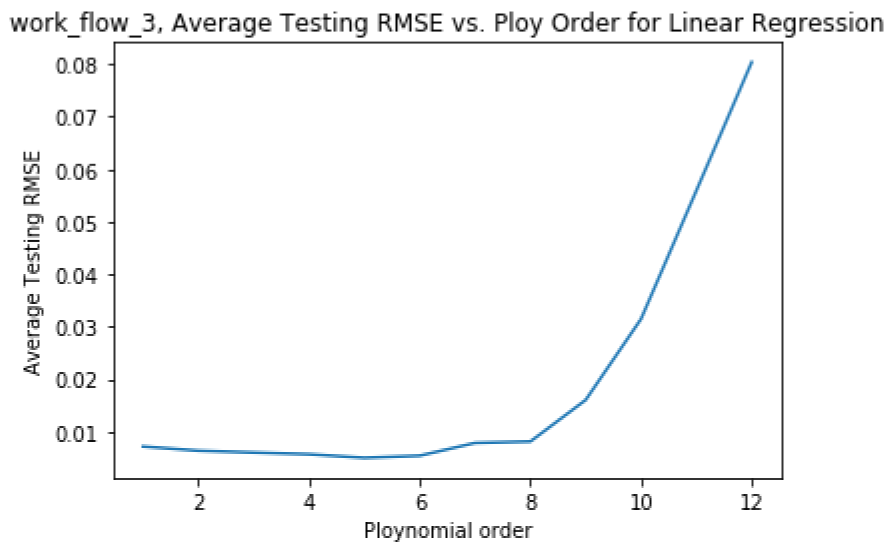


Figure 34. Average Testing RMSE vs. Polynomial Degree for Workflow 3

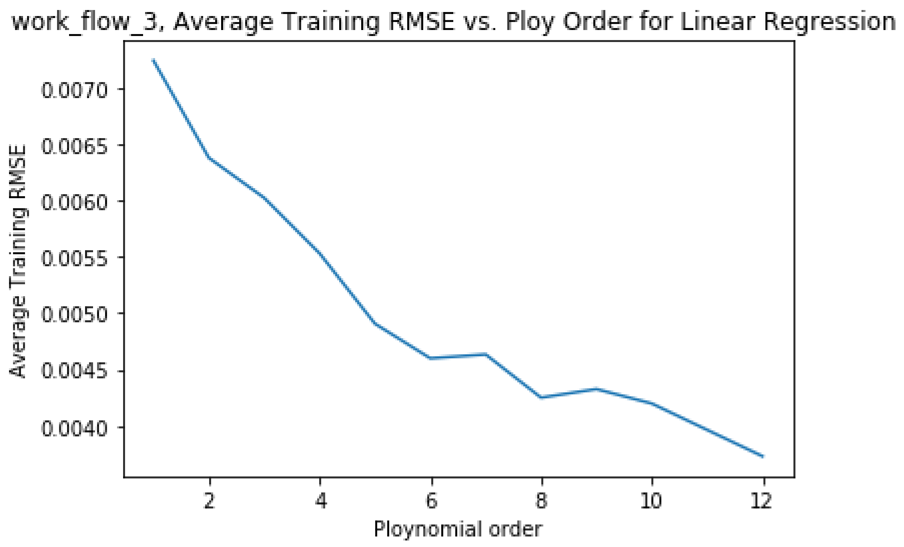


Figure 35. Average Training RMSE vs. Polynomial Degree for Workflow 3

Workflow 4:

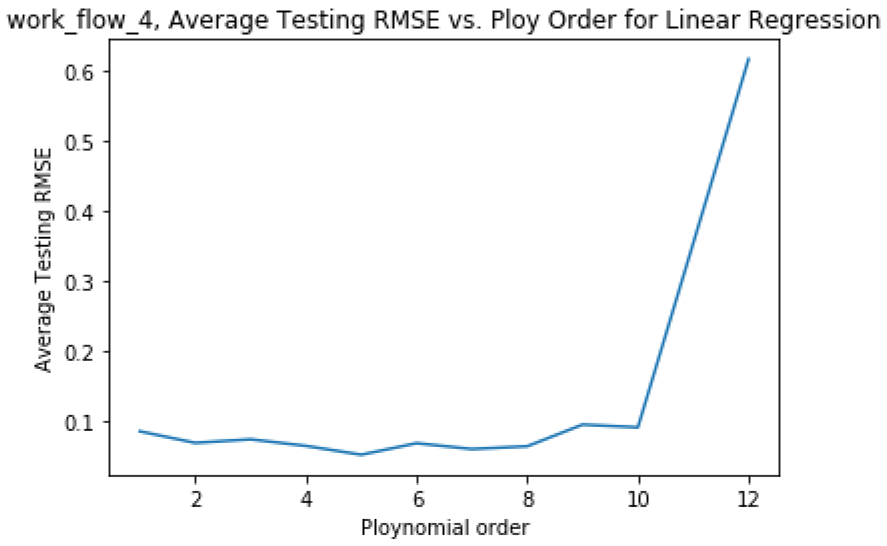


Figure 36. Average Testing RMSE vs. Polynomial Degree for Workflow 4

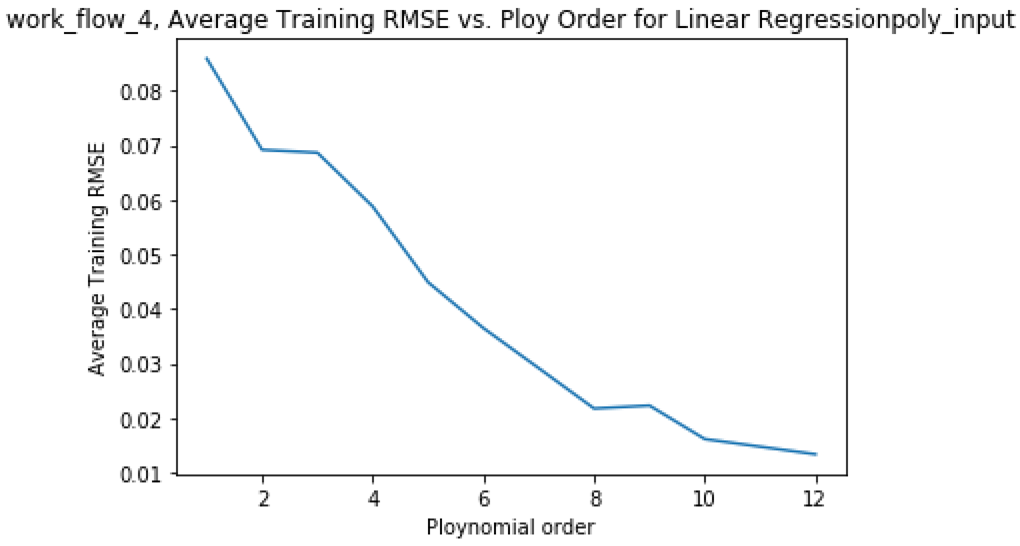


Figure 37. Average Training RMSE vs. Polynomial Degree for Workflow 4

The threshold on the degree of the fitted polynomial beyond which the generalization error of your model gets worse is around polynomial degree of 8. After this point, the test RMSE will start increasing when polynomial degree increase for workflow 0 - 4. The training RMSE will always decrease when we use higher degree of polynomial. Therefore, the polynomial degree of model results in best performance is 8.

(e) k-nearest neighbor regression

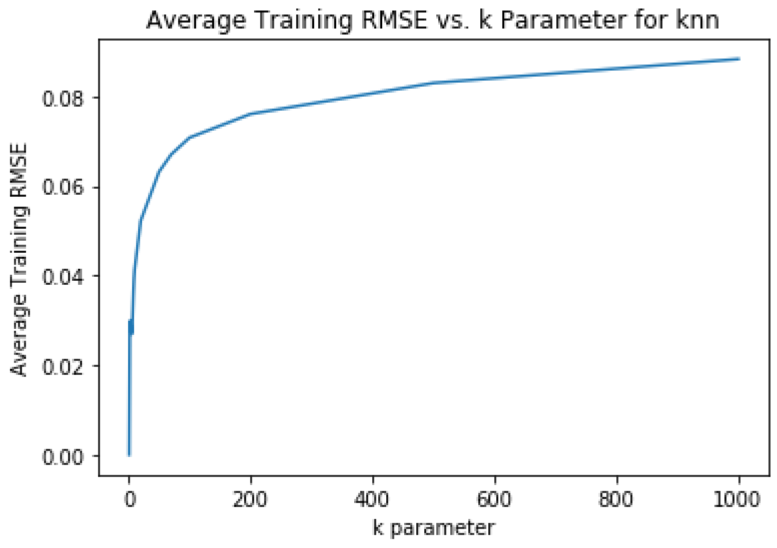


Figure 38. Average Training RMSE vs. k Parameter Using k-nearest Neighbor Regression

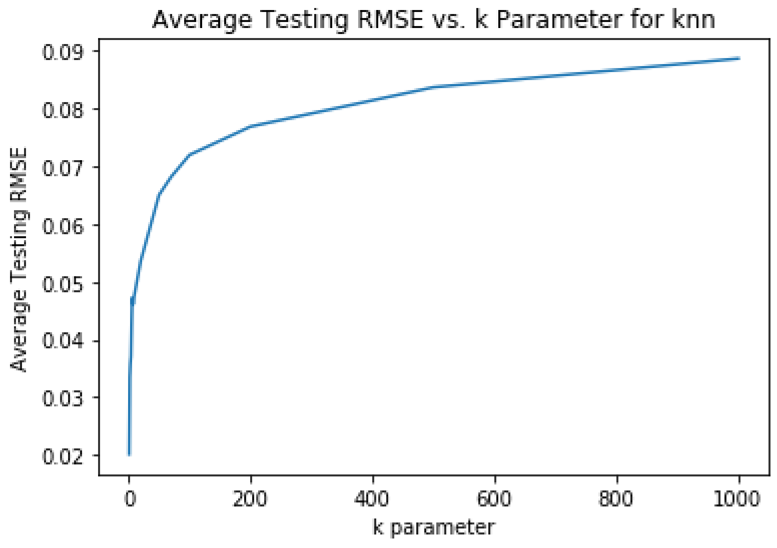


Figure 39. Average Testing RMSE vs. k Parameter Using k-nearest Neighbor Regression

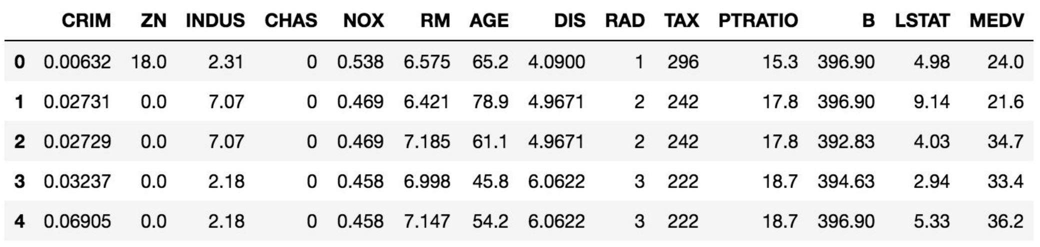
In this part, k parameter of the knn model is sweeped within 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 70, 100, 200, 500, 1000, and the performance of these models is shown in Figure 38 and Figure 39. In Figure 39, we can tell that the smallest average test RMSE is achieved using k = 1 and RMSE keeps increasing after k = 1 , so the best k parameter is k = 1 for this dataset.

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

From the results of all the regression models, we can conclude that neural network regression model with ReLu function is the best model at handling categorical features since it has the best average test RMSE. Polynomial regression model is the best model at handling sparse features, especially when we predict backup size separately for each workflow. Overall, neural network regression model generates the best results.

Dataset 2: Boston Housing Dataset

1. Load the dataset:



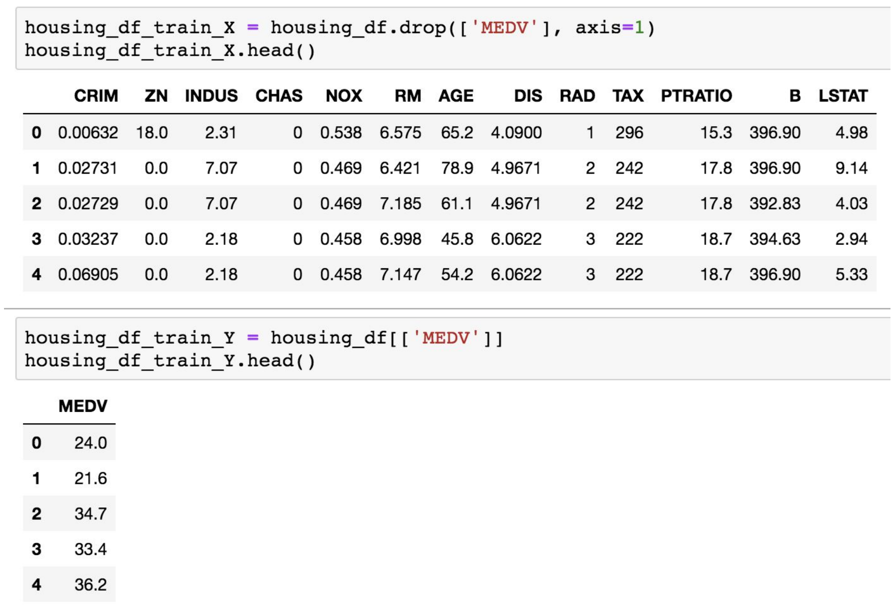
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Figure 40. Boston Housing Dataset

2. Fit a linear regression model:

(a) Set MEDV as the target variable and the other attributes as the features and ordinary least square as the penalty function.

In this part, we selected the features and target variable (MEDV) and store them into housing_df_train_X, and housing_df_train_Y as shown in Figure 40. Then, we used the OLS model in statsmodels.api to fit the dataset and the results and coefficients of the trained model is in Figure 41.



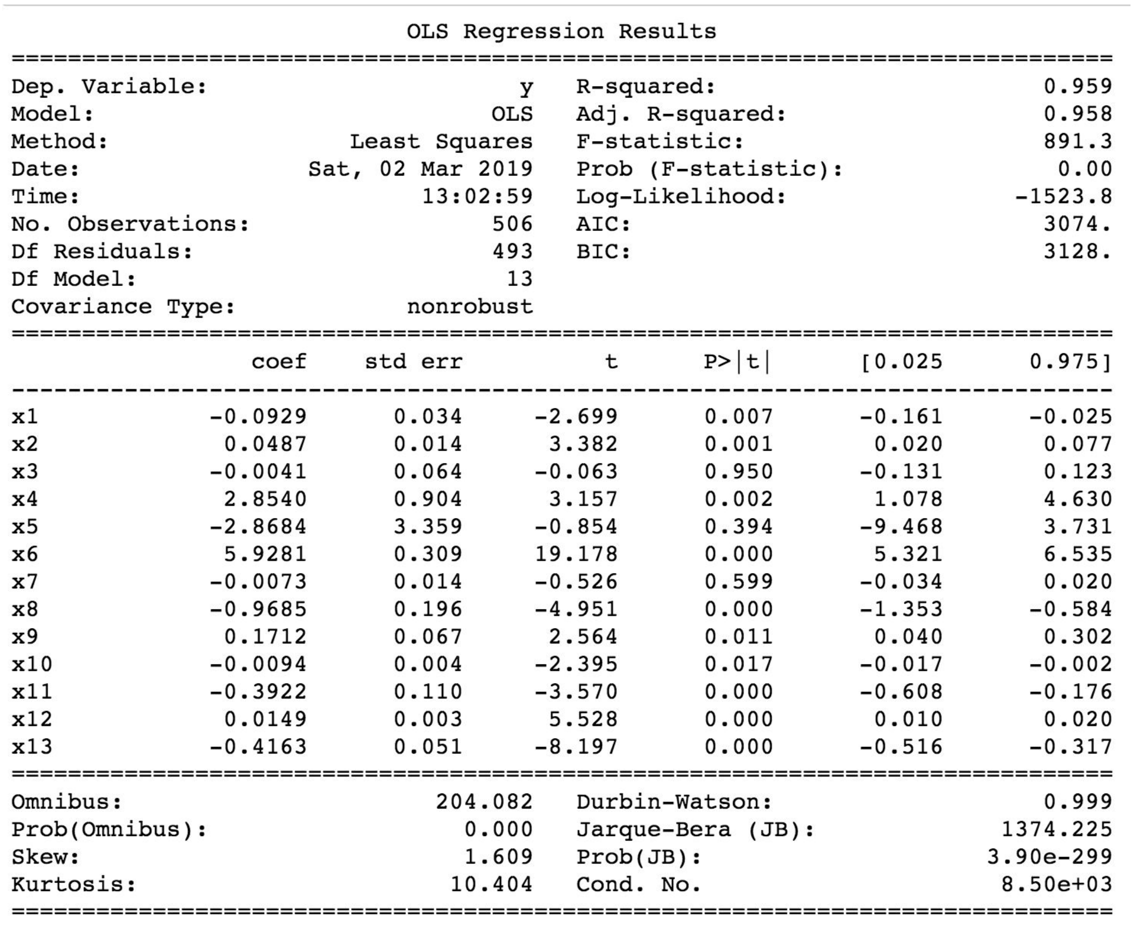
```
housing_df_train_X = housing_df.drop(['MEDV'], axis=1)
housing_df_train_X.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

```
housing_df_train_Y = housing_df[['MEDV']]
housing_df_train_Y.head()
```

	MEDV
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

Figure 41. Boston Housing Features and Target Variable



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.959
Model:                            OLS   Adj. R-squared:                  0.958
Method:                 Least Squares   F-statistic:                     891.3
Date:                Sat, 02 Mar 2019   Prob (F-statistic):               0.00
Time:                        13:02:59   Log-Likelihood:                -1523.8
No. Observations:                 506   AIC:                             3074.
Df Residuals:                     493   BIC:                             3128.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0929      0.034     -2.699      0.007      -0.161      -0.025
x2             0.0487      0.014      3.382      0.001       0.020       0.077
x3            -0.0041      0.064     -0.063      0.950      -0.131       0.123
x4             2.8540      0.904      3.157      0.002       1.078       4.630
x5            -2.8684      3.359     -0.854      0.394      -9.468       3.731
x6             5.9281      0.309     19.178      0.000       5.321       6.535
x7            -0.0073      0.014     -0.526      0.599      -0.034       0.020
x8            -0.9685      0.196     -4.951      0.000      -1.353      -0.584
x9             0.1712      0.067      2.564      0.011       0.040       0.302
x10           -0.0094      0.004     -2.395      0.017      -0.017      -0.002
x11           -0.3922      0.110     -3.570      0.000      -0.608      -0.176
x12            0.0149      0.003      5.528      0.000       0.010       0.020
x13           -0.4163      0.051     -8.197      0.000      -0.516      -0.317
==============================================================================
Omnibus:                      204.082   Durbin-Watson:                   0.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1374.225
Skew:                           1.609   Prob(JB):                    3.90e-299
Kurtosis:                      10.404   Cond. No.                     8.50e+03
==============================================================================
```

Figure 42. OLS Regression Results for Housing Dataset

(b) Perform a 10 fold cross validation, analyze the significance of different variables with the statistics obtained from the model you have trained, and the averaged Root Mean Squared Error (RMSE), and plot 1) fitted values against true values as scatter plots using the whole dataset; 2) residuals versus fitted values as scatter plots using the whole dataset.

As Figure 42 shows, the p-values of variable x3 (INDUS: proportion of non-retail business acres per town), x5 (NOX: nitric oxides concentration (parts per 10 million)) and x7 (AGE: proportion of owner-occupied units built prior to 1940) are high (>0.05), which means we cannot reject the null hypothesis and these variables are not significant in the linear model. While other variables have low p-values (<0.05), we cannot reject the null hypothesis and these variables are more significant in the model.

The average training RMSE of the trained model is 4.670454144364911, and average testing RMSE is 4.7929925360045855. The performance of the model is good as the points on fitted vs. true value plot is around diagonal line and points on the residual vs. fitted value plot are evenly distributed.

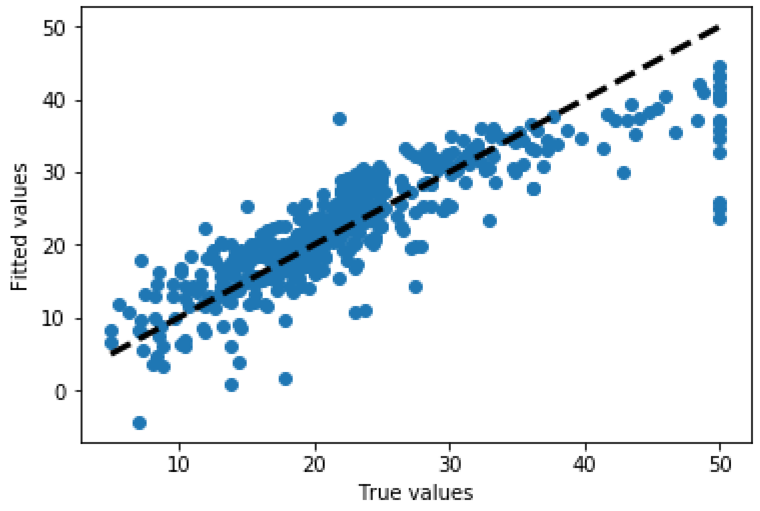


Figure 43. Scalar Encoding for Housing Dataset, Fitted vs. True Values Using Linear Regression

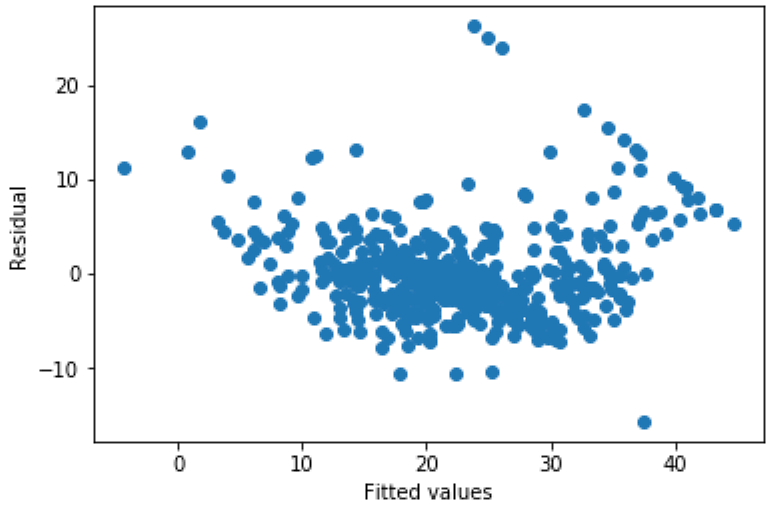


Figure 44. Scalar Encoding for Housing Dataset, Residuals vs. True Values Using Linear Regression

3. In this part, we try to control overfitting via regularization of the parameters.

(a) You are asked to try the following regularizations with suitable parameters.

1. Ridge Regularizer

2. Lasso Regularizer

3. Elastic Net Regularizer (optional)

Optimize over choices of ; 1; 2 to pick one good model, report the best RMSE obtained via 10-fold cross validation. Compare the values of the estimated coefficients for these regularized good models, with the unregularized best model.

1. Ridge Regularizer

alpha	Average training RMSE	Average testing RMSE
1	5.545176844466782	5.579532934486909
0.1	4.751909135062023	4.842625212247374
0.01	4.672847000708183	4.790628723005426
0.001	4.670483185518561	4.792508850868982

Table 5. RMSE results with different alpha using Ridge Regularizer

From the experiments result we find when alpha = 0.01, the average testing RMSE is smallest.
Best Alpha value for Ridge Regression : 0.01
Best test RMSE for corresponding Alpha = 4.790628723005426

2. Lasso Regularizer

alpha	Average training RMSE	Average testing RMSE
1	9.187066923162694	9.1750680568801529
0.1	5.8631153265534115	5.8899258054840615
0.01	4.8473396474252475	4.955764682248477
0.001	4.673439627316392	4.787186640815802
0.0005	4.671258450235468	4.789598782525104
0.0001	4.6704879302046765	4.792236760244802

Table 6. RMSE results with different alpha using Lasso Regularizer

From the experiments result we find when alpha = 0.001, the average testing RMSE is smallest.
Best Alpha value for Lasso Regularization : 0.001
Best test RMSE for corresponding Alpha = 4.787186640815802

3. Elastic Net Regularizer

alpha	L1_ratio	Average training RMSE	Average testing RMSE
1	0	5.116749315249062	5.223342978336722
	0.2	5.128575442260425	5.236918138114297
	0.4	5.136341450514844	5.245727211600449
	0.6	5.145774999038599	5.256421912306559
	0.8	5.157024630992626	5.270591362996316
	1	5.170088094751252	5.294695089614015

0.1	0	4.814570102178061	4.930576076156072
	0.2	4.8093482451138625	4.926686866275611
	0.4	4.805051254175773	4.924159874802844
	0.6	4.801258362837208	4.922411617413
	0.8	4.795824856433062	4.92005231586951
	1	4.7917697372284405	4.921197491664635
0.01	0	4.731737276528432	4.85474856650007
	0.2	4.726586004684139	4.849880978204611
	0.4	4.7196810110338925	4.843202502343379
	0.6	4.709902642623378	4.833395780358748
	0.8	4.695189235542577	4.818526506441823
	1	4.674366116730136	4.797535166240188
0.001	0	4.676754600637215	4.799120287950053
	0.2	4.675049085857777	4.79740117133034
	0.4	4.673470683580314	4.795827535807279
	0.6	4.672099613047386	4.79448791985361
	0.8	4.671051347267063	4.793513004501272
	1	4.670493416939514	4.793080855250757
0.0001	0	4.6705585150684445	4.793024848595755
	0.2	4.670523788092535	4.793003940318688
	0.4	4.670495750862608	4.792990458366427
	0.6	4.670474673454744	4.79298469141265
	0.8	4.670460837868546	4.792986959523869
	1	4.6704545370419215	4.79299758740464

Table 7. RMSE results with different alpha and L1 ratio using Elastic Net Regularizer

From the experiments result we find when alpha = 0.0001, L1 ratio = 0.6, the average testing RMSE is smallest.
Best Alpha value for Elastic Net Regularization : 0.0001
Best L1 ratio for Elastic Net Regularization: 0.6

In this question we use the ElasticNet model in sklearn.linear model library. The object function we are going to minimize is :

```
1 / (2 * n_samples) * ||y - Xw||^2_2 + alpha * l1_ratio * ||w||_1 + 0.5 * alpha
* (1 - l1_ratio) * ||w||^2_2
```

Hence we have λ1 = 0.00006 and have λ2 = 0.00002 for the best model
Best test RMSE for corresponding parameters: 4.79298469141265
(More experiments should be carried out on parameter combinations if time permits)

The estimated coefficients for the three regularized models and the unregularized model are in the table below.

Regularizer	x1	x2	x3	x4	x5	x6	x7
N/A	-1.08011358e-01	4.64204584e-02	2.05586264e-02	2.68673382e+00	-1.77666112e+01	3.80986521e+00	6.92224640e-04
Ridge	-1.03542237e-01	4.34058197e-02	5.19960647e-03	2.74630656e+00	-1.66255959e+01	3.86518807e+00	-3.41085552e-04
Lasso	-1.00549239e-01	4.23750555e-02	0.00000000e+00	2.69026297e+00	-1.65258561e+01	3.85232711e+00	-0.00000000e+00
Elastic Net	-9.94116898e-02	4.07115899e-02	-6.62321503e-03	2.78881844e+00	-1.55715998e+01	3.91033918e+00	-1.10042411e-03

Regularizer	x8	x9	x10	x11	x12	x13
N/A	-1.47556685e+00	3.06049479e-01	-1.23345939e-02	-9.52747232e-01	9.31168327e-03	-5.24758378e-01
Ridge	-1.41355030e+00	2.69158524e-01	-1.05767047e-02	-9.34595971e-01	9.28758725e-03	-5.15910557e-01
Lasso	-1.41597570e+00	2.63004585e-01	-1.02547310e-02	-9.33489599e-01	9.08023522e-03	-5.22501984e-01
Elastic Net	-1.35109532e+00	2.38197768e-01	-9.17103894e-03	-9.17641474e-01	9.24664786e-03	-5.07702523e-01

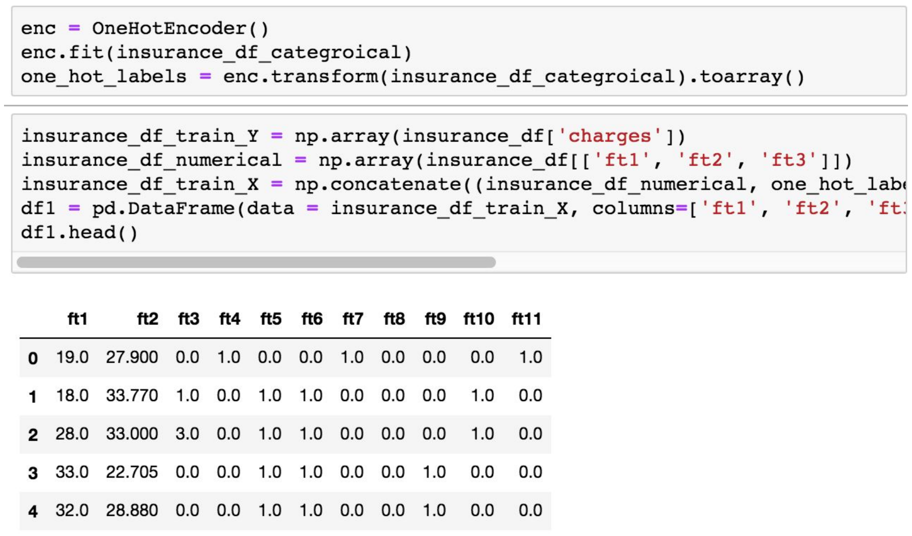
Table 8. Coefficients for different regularized models

Dataset 3:

1. Feature Preprocessing

For each model,
1) Report the average training RMSE and average test RMSE for 10 fold cross validation
2) Plot fitted values against true values as scatter plots using the whole dataset
3) Plot residuals versus fitted values as scatter plots using the whole dataset

(a) Feature Encoding: Use one-hot-encoding for the following 3 categorical features: ft4, ft5, ft6. Use the encoded features and the numerical features to fit a linear regression model.



```
enc = OneHotEncoder()
enc.fit(insurance_df_categroical)
one_hot_labels = enc.transform(insurance_df_categroical).toarray()
```

```
insurance_df_train_Y = np.array(insurance_df['charges'])
insurance_df_numerical = np.array(insurance_df[['ft1', 'ft2', 'ft3']])
insurance_df_train_X = np.concatenate((insurance_df_numerical, one_hot_labe
df1 = pd.DataFrame(data = insurance_df_train_X, columns=['ft1', 'ft2', 'ft
df1.head()
```

	ft1	ft2	ft3	ft4	ft5	ft6	ft7	ft8	ft9	ft10	ft11
0	19.0	27.900	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
1	18.0	33.770	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
2	28.0	33.000	3.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3	33.0	22.705	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
4	32.0	28.880	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0

Figure 45. Car Insurance Data Features
(with one-hot encoding on original feature 4, 5 and 6)

Performing 10-fold cross validation on a linear regression model we get statistics as follow:
Average training RMSE: 6040.352471670377
Average testing RMSE: 6066.1961592429025

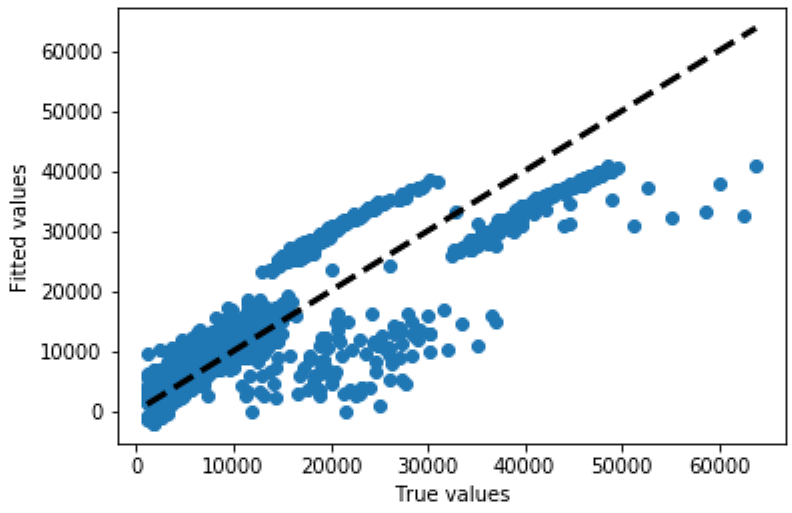


Figure 46. Scalar Encoding for Insurance Dataset with One Hot Encoding, Fitted vs. True Values Using Linear Regression

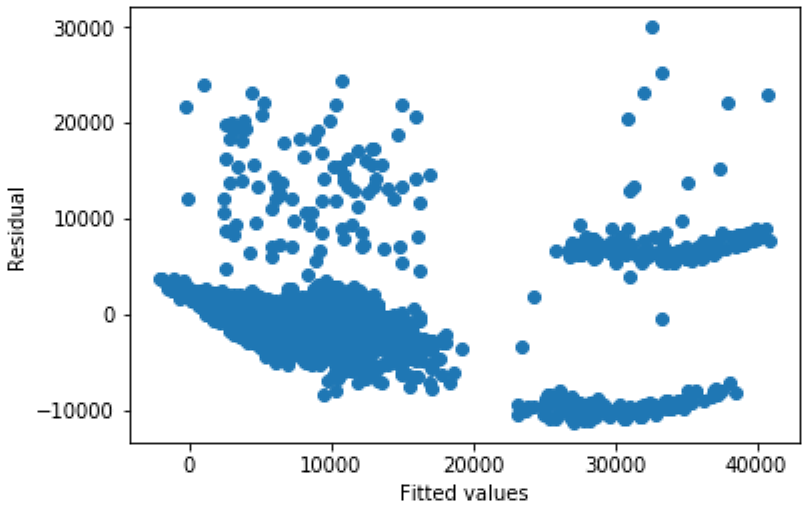
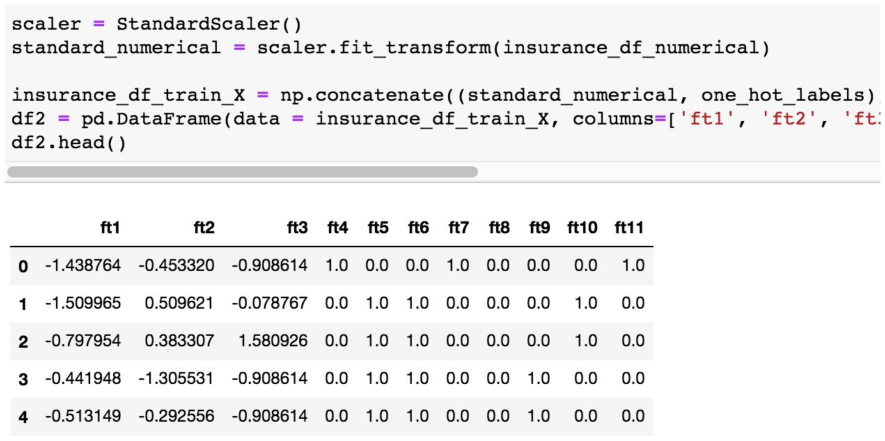


Figure 47. Scalar Encoding for Insurance Dataset with One Hot Encoding, Residuals vs. True Values Using Linear Regression

(b) Standardization: Standardize (see the Useful Functions Section) all these numerical features and keep the one-hot-encoded features. Fit a linear regression model.



```
scaler = StandardScaler()
standard_numerical = scaler.fit_transform(insurance_df_numerical)

insurance_df_train_X = np.concatenate((standard_numerical, one_hot_labels)
df2 = pd.DataFrame(data = insurance_df_train_X, columns=['ft1', 'ft2', 'ft
df2.head()
```

	ft1	ft2	ft3	ft4	ft5	ft6	ft7	ft8	ft9	ft10	ft11
0	-1.438764	-0.453320	-0.908614	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
1	-1.509965	0.509621	-0.078767	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
2	-0.797954	0.383307	1.580926	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3	-0.441948	-1.305531	-0.908614	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
4	-0.513149	-0.292556	-0.908614	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0

Figure 48. Car Insurance Data Features
(with one-hot encoding on original feature 4, 5 and 6 and standardizing on feature 1, 2, and 3)

Performing 10-fold cross validation on a linear regression model we get statistics as follow:
Average training RMSE: 6040.884853215843
Average testing RMSE: 6072.56145214846

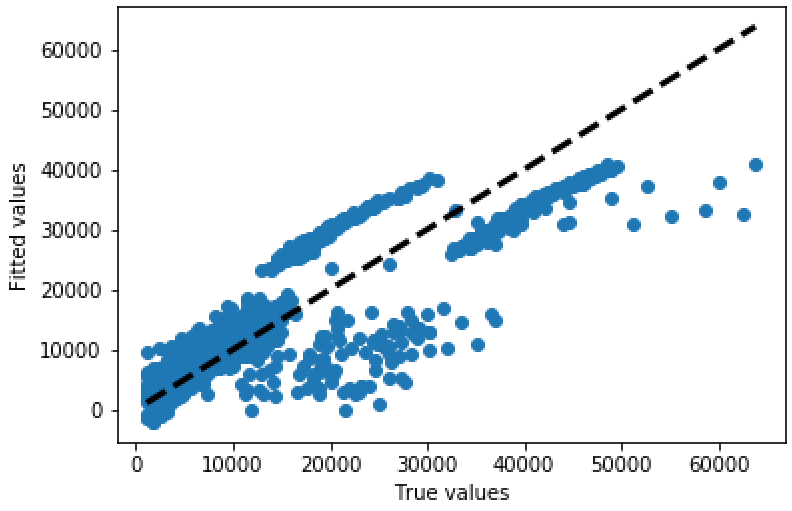


Figure 49. Scalar Encoding for Data with Standardizing, One Hot Encoding, Fitted vs. True Values Using Linear Regression

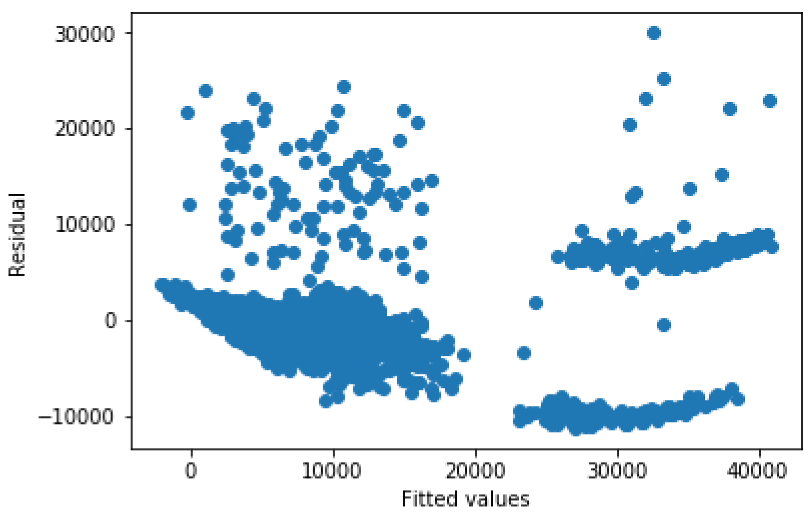
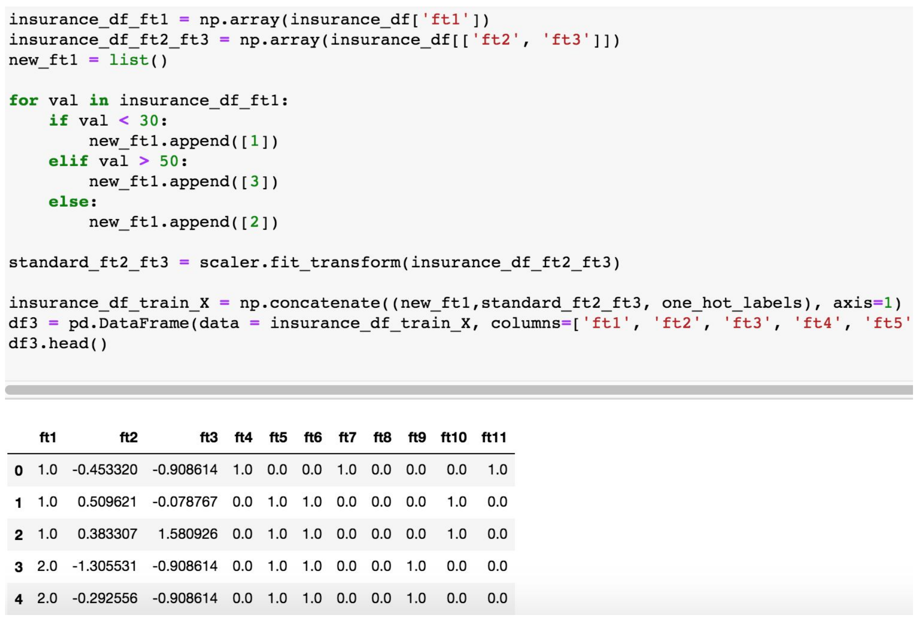


Figure 50. Scalar Encoding for Data with Standardizing, One Hot Encoding, Residual vs. Fitted Values Using Linear Regression

(c) Divide ft1 into 3 ranges: < 30, [30,50] and > 50. That is, set the new values to 1 for original values below 30, 2 for values between 30 and 50 and 3 for values above 50. Standardize ft2 and ft3 feature. One-hot encoding the rest three categorical features. Fit a linear regression model.



```
insurance_df_ft1 = np.array(insurance_df['ft1'])
insurance_df_ft2_ft3 = np.array(insurance_df[['ft2', 'ft3']])
new_ft1 = list()

for val in insurance_df_ft1:
    if val < 30:
        new_ft1.append([1])
    elif val > 50:
        new_ft1.append([3])
    else:
        new_ft1.append([2])

standard_ft2_ft3 = scaler.fit_transform(insurance_df_ft2_ft3)

insurance_df_train_X = np.concatenate((new_ft1,standard_ft2_ft3, one_hot_labels), axis=1)
df3 = pd.DataFrame(data = insurance_df_train_X, columns=['ft1', 'ft2', 'ft3', 'ft4', 'ft5'
df3.head()
```

	ft1	ft2	ft3	ft4	ft5	ft6	ft7	ft8	ft9	ft10	ft11
0	1.0	-0.453320	-0.908614	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
1	1.0	0.509621	-0.078767	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
2	1.0	0.383307	1.580926	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3	2.0	-1.305531	-0.908614	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
4	2.0	-0.292556	-0.908614	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0

Figure 51. Car Insurance Data Features
(with one-hot encoding on original feature 4, 5 and 6 and standardizing on feature 2 and 3, divide feature 1 into three ranges)

Performing 10-fold cross validation on a linear regression model we get statistics as follow:
Average training RMSE: 6199.837835214862
Average testing RMSE: 6226.071276987422

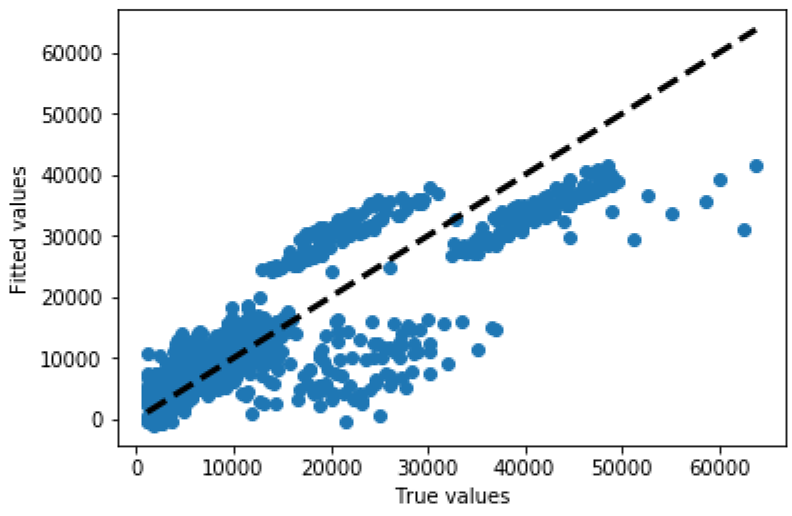


Figure 52. Scalar Encoding for Dividing ft1 into 3 ranges, Standardize ft2 ft3, One Hot Encoding, Fitted vs. True Values Using Linear Regression

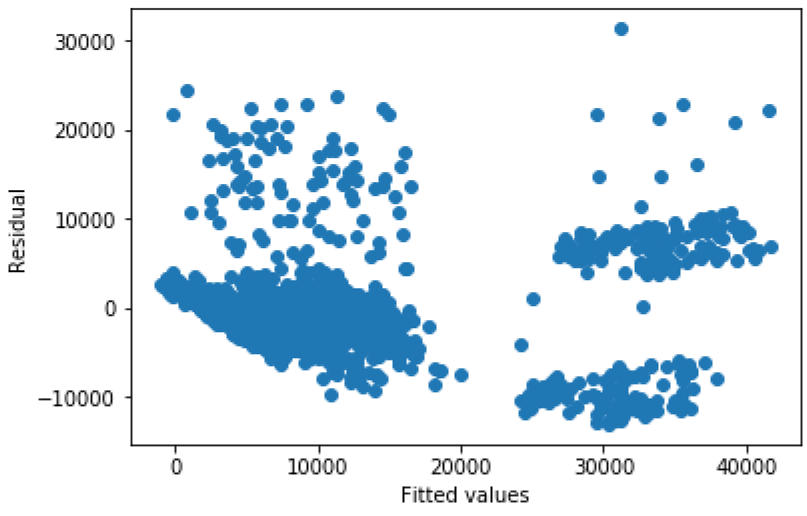


Figure 53. Scalar Encoding for Dividing ft1 into 3 ranges, Standardize ft2 ft3, One Hot Encoding, Residual vs. Fitted Values Using Linear Regression

2. Correlation exploration:

(a) Convert each categorical feature into a one dimensional numerical value. Now we have 6 numerical features. Use f regression and mutual information regression measure to select two most important variables respectively. Report the two most important variables you find.

Perform F regression and mutual information regression. high f value, low p value means significant, high mutual info value means significant.

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6
F value	1.31174013e+02	5.47093081e+01	6.20603705e+00	4.39970170e+00	2.17761487e+03	5.14943381e-02
P value	4.88669333e-029	2.45908554e-013	1.28521285e-002	3.61327210e-002	8.27143584e-283	8.20517836e-001
Mutual info value	1.49792104	0.07358636	0.16199842	0.17665987	0.36917105	0.07622114

Table 7. Importance Measurements of 6 Features

Based of the f value, p value, and mutual info value in Table 1, the 2 most important variables are feature 5 and feature 1, as feature 5 and feature 1 have the highest f value, lowest p value and highest mutual information value.

(b) Scatter plot charges (y axis) vs ft2 (x axis), and color points based on ft5 (Yes or No).

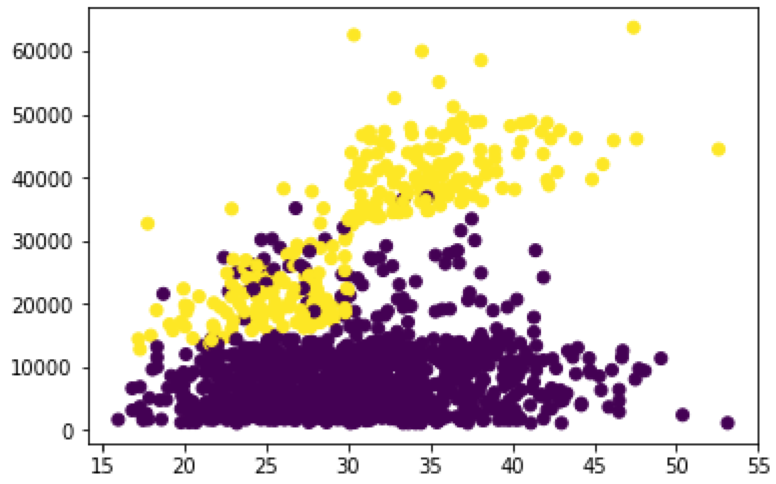


Figure 54. Scatter plot - charges vs. feature 2 (ft5 = no is purple, ft5 = yes is yellow)

(c) Scatter plot charges (y axis) vs ft1 (x axis), and color points based on ft5 (Yes or No).

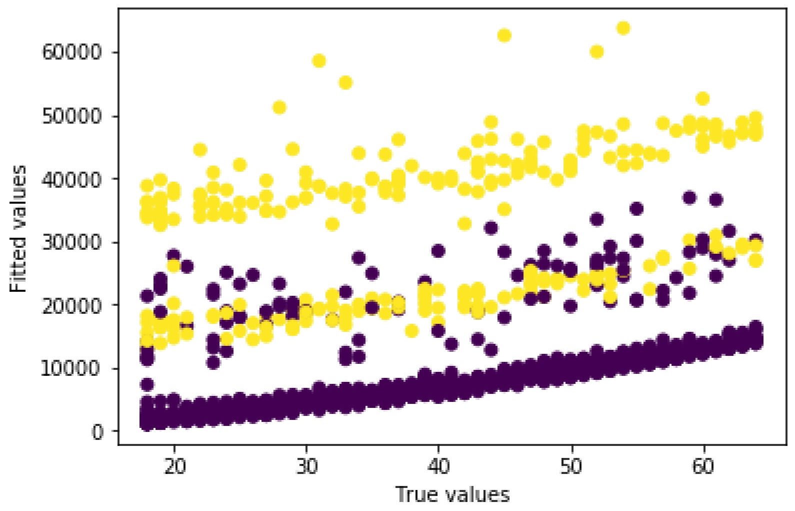


Figure 55. Scatter plot - charges vs. feature 1 (ft5 = no is purple, ft5 = yes is yellow)

3. Modify the target variable

(a) Pick one method of feature preprocessing from question 1 to train a linear regression model on this new target. Does the performance improve?

For the feature preprocessing, we used one hot encoding for feature 4 to feature 6, and used the original values for feature 1 to feature 3.

Performing 10-fold cross validation on a linear regression model we get statistics as follow:
Average training RMSE: 8376.83969997326
Average testing RMSE: 8399.363589718047
The performance gets worse. It is probably because applying a log function on the target variable makes the difference between target variables smaller, hence it is harder for the linear regressor to find the relationship between features and targets, and the error gets higher.

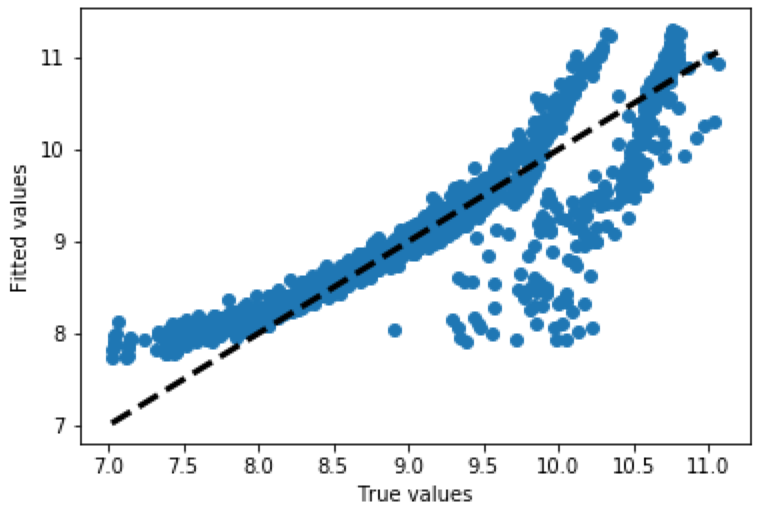


Figure 56. Scalar Encoding for Data with One Hot Encoding and Log Target, Fitted vs. True Values

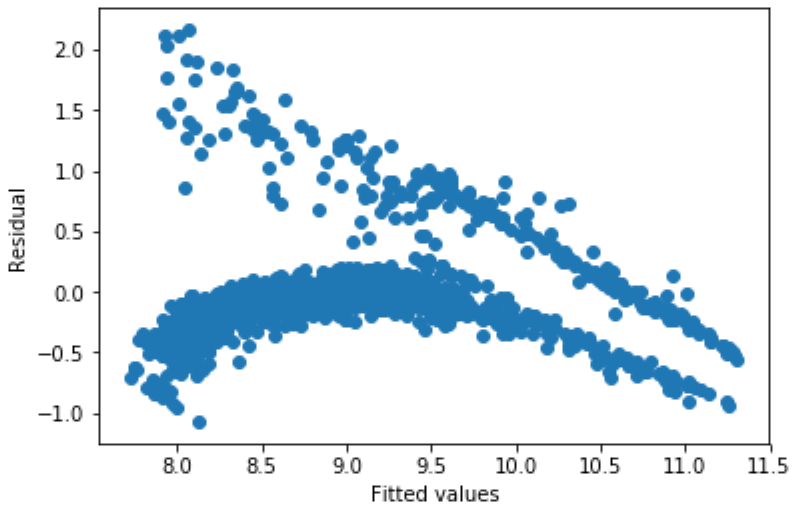


Figure 57. Scalar Encoding for Data with One Hot Encoding and Log Target, Residual vs. Fitted Values

(b) Repeat the correlation exploration part for the new target.

Repeat Correlation Exploration by fitting log(y) to the variable target. High f value, low p value means significant, and high mutual info value means significant.

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6
F value	5.15977081e+02	2.39364692e+01	3.57046705e+01	4.23764137e-02	1.06212392e+03	2.43916473e+00
P value	7.47738522e-097	1.11667665e-006	2.94069066e-009	8.36935289e-001	6.30764636e-172	1.18576054e-001
Mutual info value	1.4994657	0.06765267	0.16101784	0.17628529	0.36939335	0.07753486

Table 8. Importance Measurements of 6 Features

Based of the f value, p value, and mutual info value in Table 1, the 2 most important variables are feature 5 and feature 1 as feature 5 and feature 1 have the highest f value, lowest p value and highest mutual information value.

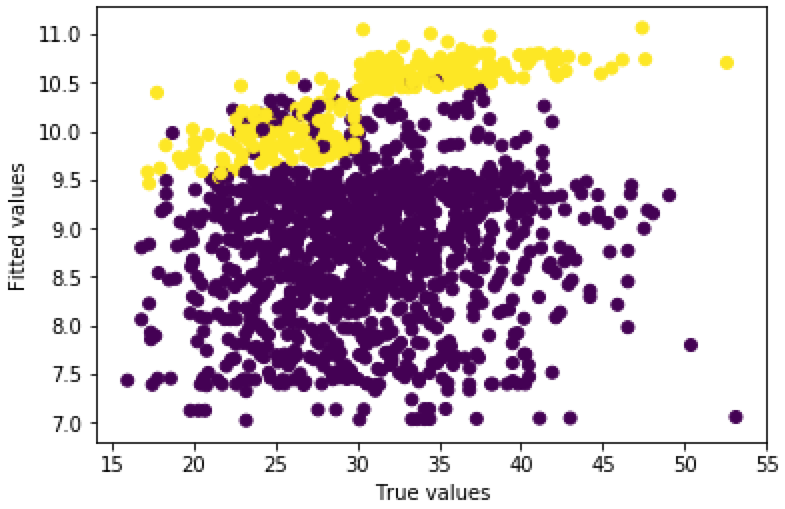


Figure 58. Scatter Plot for Data with One Hot Encoding and Log Target - Charges vs. Feature 2 (ft5 = no is purple, ft5 = yes is yellow)

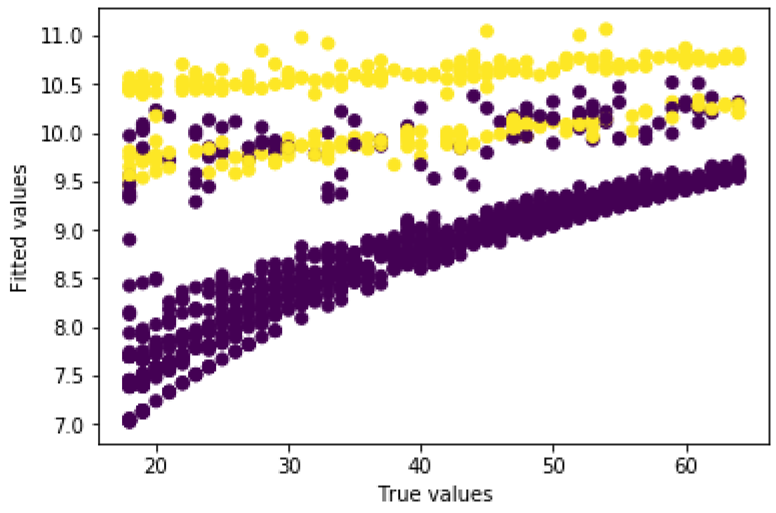
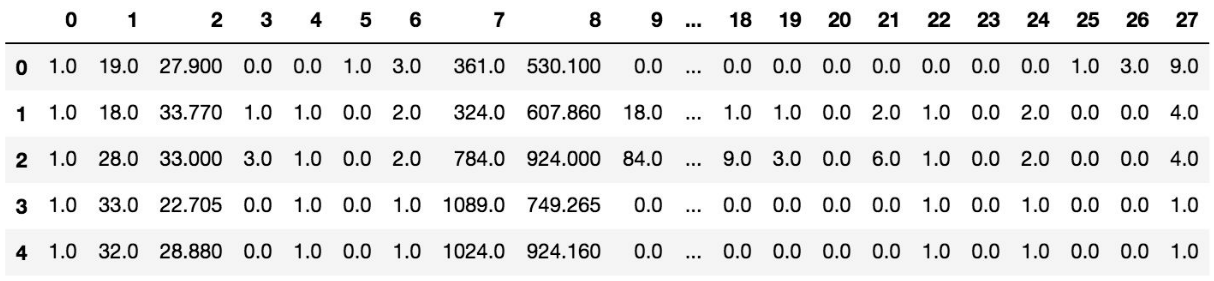


Figure 59 . Scatter Plot for Data with One Hot Encoding and Log Target - Charges vs. Feature 1(ft5 = no is purple, ft5 = yes is yellow)

4. Bonus questions:

(a) Considering current results, can you further improve your results by better feature encoding? You can try polynomial features or different combinations of encoding methods.

For this question, we first converted each categorical feature into a numerical value and applied polynomial feature transformation of degree 2 on all the features. We get the new dataframe as follow.



	0	1	2	3	4	5	6	7	8	9	...	18	19	20	21	22	23	24	25	26	27
0	1.0	19.0	27.900	0.0	0.0	1.0	3.0	361.0	530.100	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0	9.0
1	1.0	18.0	33.770	1.0	1.0	0.0	2.0	324.0	607.860	18.0	...	1.0	1.0	0.0	2.0	1.0	0.0	2.0	0.0	0.0	4.0
2	1.0	28.0	33.000	3.0	1.0	0.0	2.0	784.0	924.000	84.0	...	9.0	3.0	0.0	6.0	1.0	0.0	2.0	0.0	0.0	4.0
3	1.0	33.0	22.705	0.0	1.0	0.0	1.0	1089.0	749.265	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
4	1.0	32.0	28.880	0.0	1.0	0.0	1.0	1024.0	924.160	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0

Figure 60. Car Insurance Data Features
(with polynomial transformation of degree 2 on all features)

Performing 10-fold cross validation on a linear regression model we get statistics as follow:
Average training RMSE: 4737.268113280409
Average testing RMSE: 4799.386560914034

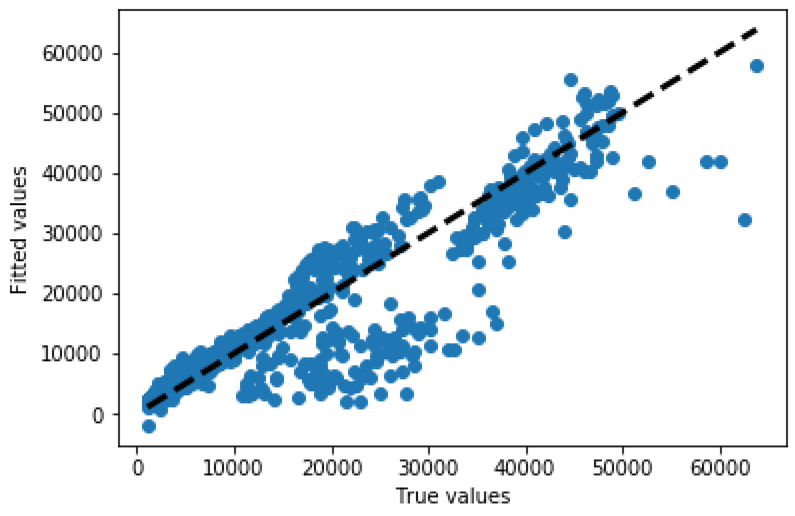


Figure 61. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Linear Regression

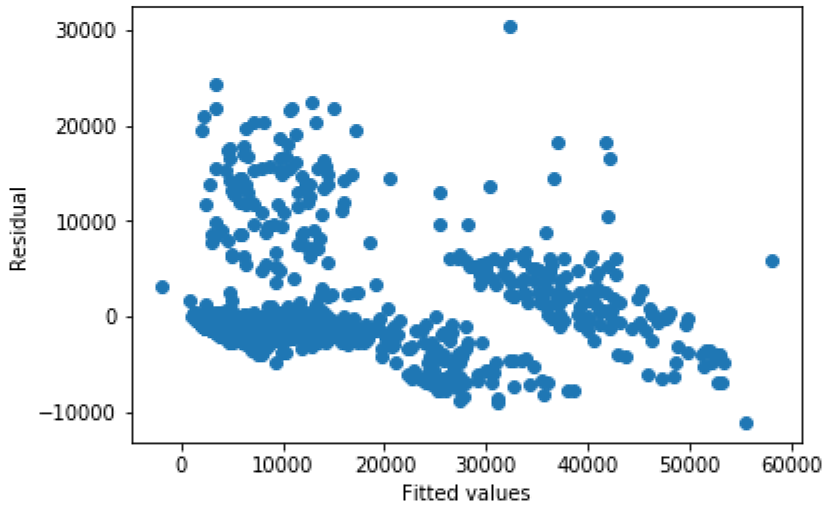


Figure 62. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Residual vs. Fitted Values Using Linear Regression

(b) Can you further improve your results by picking a better model?

i. You should try at least three different types of models (e.g. random forest, neural network, gradient boosting tree)

ii. You can also try to modify the hyper-parameters of the models you have tried. Hint: you can use GridSearchCV as in project 1.

1. Random Forest

We used a Random Forest Regressor in scikit-learn with n_estimators = 200, max_depth = 5, bootstrap = True and oob_score = True. Performing 10-fold cross validation on a random forest regressor we get statistics as follow:

Average training RMSE: 4051.3822315291645

Average testing RMSE: 4493.500041118015

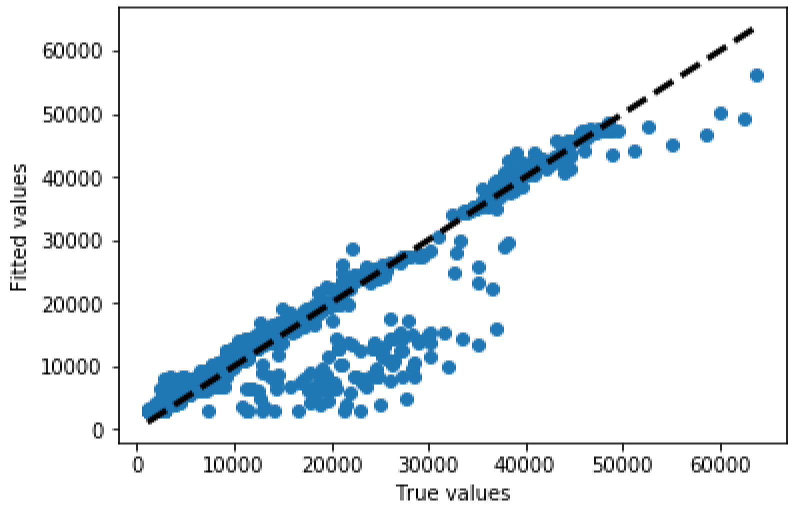


Figure 63. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Random Forest Regressor

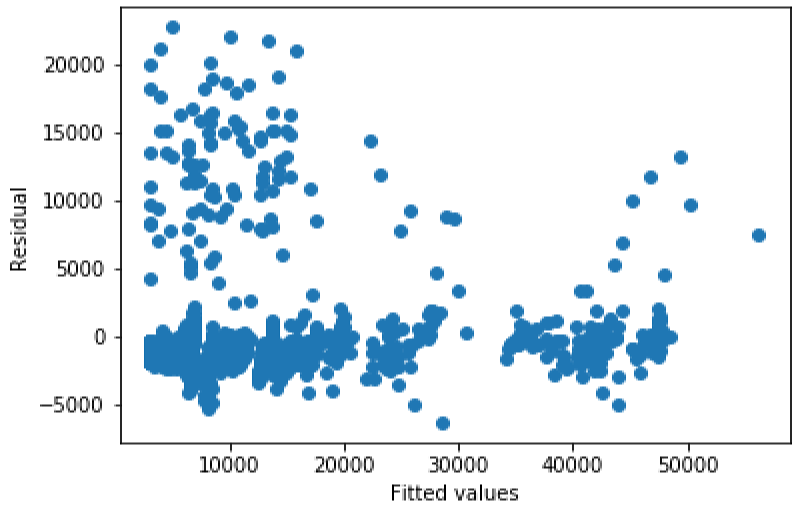


Figure 64. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Residual vs. Fitted Values Using Random Forest Regressor

2. Neural Network

We used a Multi-layer Perceptron Regressor in scikit-learn with two hidden layers with size = [200,200]. Performing 10-fold cross validation on a multi-layer perceptron regressor we get statistics as follow:

Average training RMSE: 5152.737957716946

Average testing RMSE: 5198.432717488813

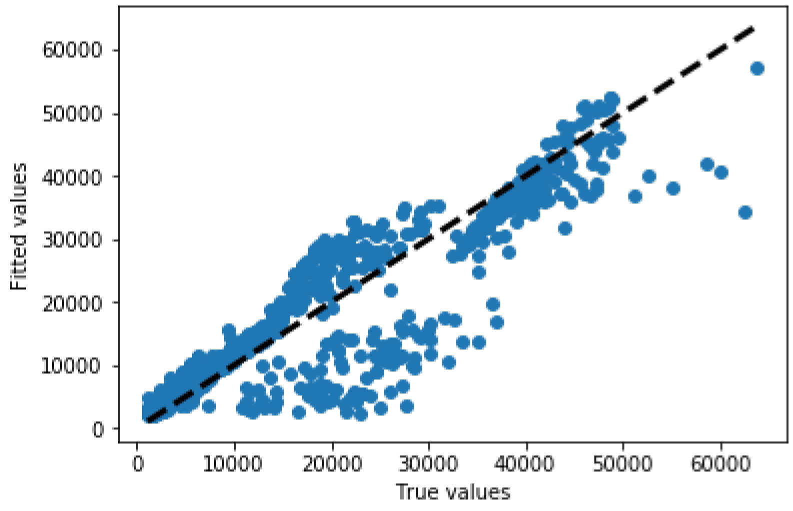


Figure 65. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Multi-layer Perceptron Regressor

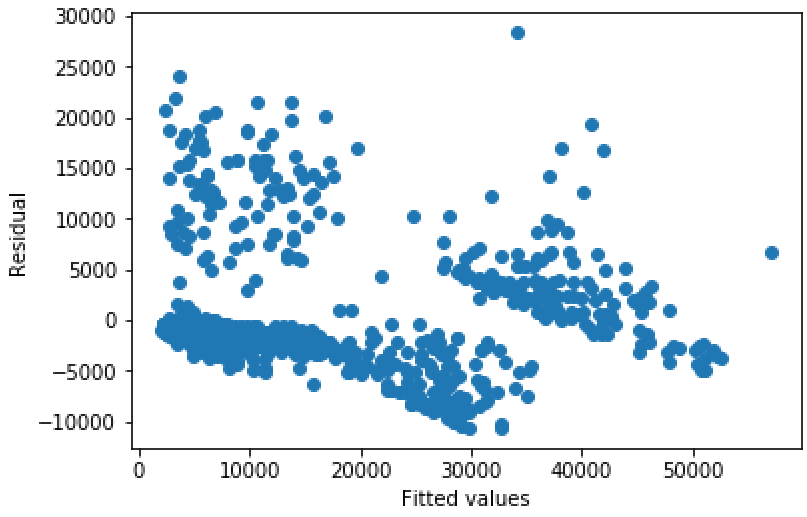


Figure 66. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Multi-layer Perceptron Regressor

3. Gradient Boosting Regressor

We used a Gradient Boosting Regressor in scikit-learn withn_estimators=50 and learning_rate = 0.1. Performing 10-fold cross validation on a multi-layer perceptron regressor we get statistics as follow:

Average training RMSE: 3890.481261454542

Average testing RMSE: 4489.036661482436

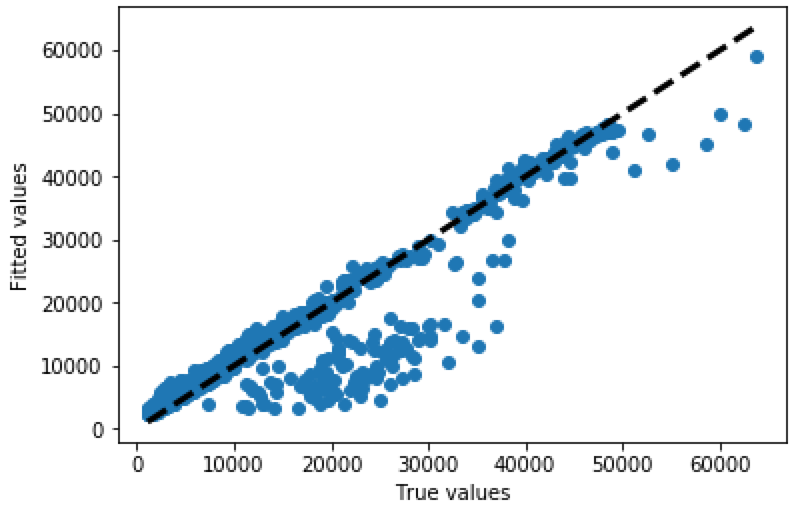


Figure 67. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Gradient Boosting Regressor

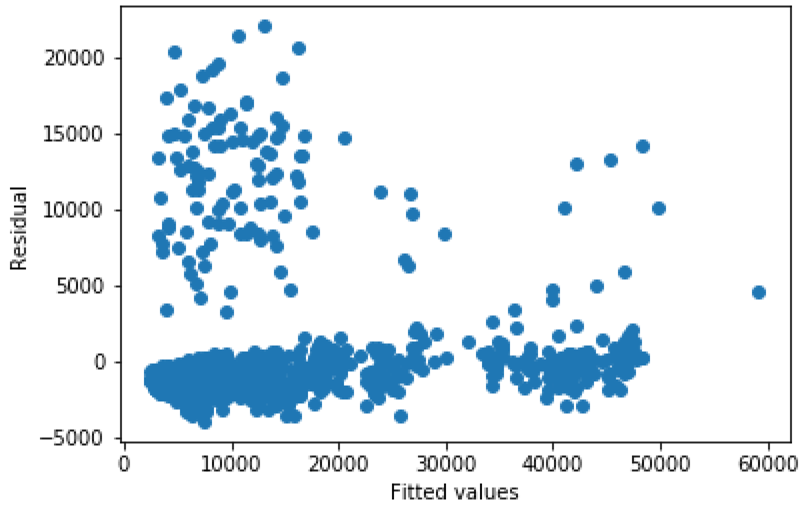


Figure 68. Scalar Encoding for Insurance Dataset with Polynomial Feature Transformation, Fitted vs. True Values Using Gradient Boosting Regressor

Comparing the results of three regressors, the best model is the Gradient Boosting Regressor with average testing RMSE = 4489.036661482436

Conclusion

Until now, we have experimented on three different datasets. Based on the experiences, comment on what kinds of model is more suitable for what specific type of data and what is the best feature preprocessing stage.

From dataset 1, we can conclude that neural network regression model is the best model for 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.

From dataset 2, we find that Lasso Regression performs better than linear regression, ridge regression and elastic net regression when handling a large number of numerical features as in Boston Housing dataset.

From dataset 3, the best feature preprocessing method is converting all features in to numerical features and applying polynomial feature transformation, which significantly reduces the RMSE in our experiment. Then gradient boosting regressor and random forest regressor work best on the preprocessed data with numerical values. Linear regression works not very well on the Car Insurance dataset with both categorical and numerical features, and using one-hot encoding and standardization on features will decrease the performance, and using the log value of target variable will make the result even worse.