Regression Analysis

University of California, Los Angeles
ECE 219 Project 4

Joshua Hannan (UID: 805221851)
James (Zach) Harris (UID: 105221897)
Dennis Wang (UID: 105229662)
Akash Deep Singh (UID: 405228294)

jhannan@g.ucla.edu
jzharris@g.ucla.edu
dmwang626@g.ucla.edu
akashdeepsingh@g.ucla.edu

Introduction

So far, we have dealt with algorithms that perform classification of our data. In this project, we will look into another machine learning method called regression that produces relationships between the output variable and the other variables that produce this output. We explore basic regression models over 3 given datasets and try various methods like cross-validation and regularization to handle overfitting.

Dataset 1

The first dataset is called the 'Network Backup Dataset', which is a simulated dataset comprised of traffic data on a backup system over a network. The dataset has around 18000 points and the following columns:

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

Question 1

(a) In this section, we plot the data for each workflow ID separately for the first 20 days. Since, the dataset does not contain the number of days, we us the following algorithm to select the data for the first 20 days:

- We know that every week has 7 days.
- So, 20^{th} day will be one day before the end of 3^{rd} week.
- In the dataset, Monday is the first day of the week. So Sunday will be the last day.
- Hence, we need data till just before the Sunday of the 3^{rd} week.

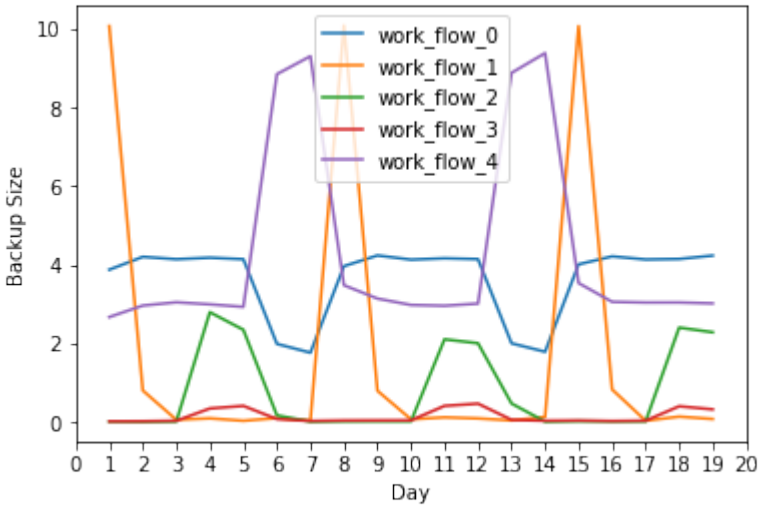

Fig. 1. Backup sizes in GB for the first 20-day period for all workflows

(b) In this section, we plot the dataset for the 105 day period. Day index is found by incrementing a variable every time the 'Day of the week changes' in the data.

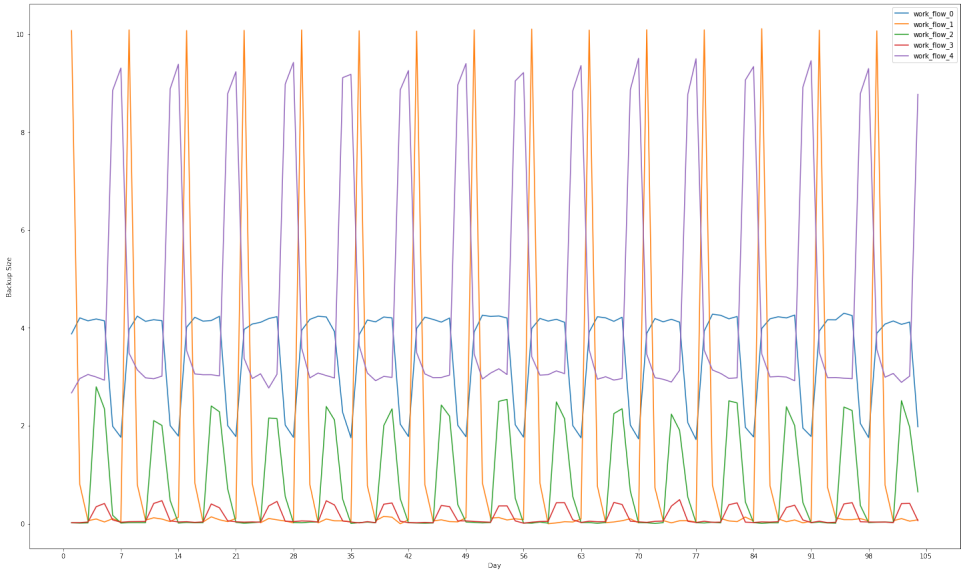

Fig. 2. Backup sizes in GB for the first 105-day period for all workflows

(c) Looking at the 2 plots above, we notice the following patterns very clearly:

- **work_flow_0:** Fluctuates slightly above $4GB$ during the weekdays but goes back down to $2GB$ during weekends.
- **work_flow_1:** Shows the highest value - $10GB$ on the 1^{st} day of the week, but decreases to around $1GB$ on the 2^{nd} day and eventually goes to $0GB$ on the 3^{rd} day of the week and stays there for the second half of the week.
- **work_flow_2:** Size of this backup is around $2GB$ for the 4^{th} and 5^{th} days of the week otherwise it is almost $0GB$.
- **work_flow_3:** Shows some values on the 4^{th} and 5^{th} days of the week, otherwise is almost $0GB$. Peak is around $0.4GB$
- **work_flow_4:** On weekdays, it is around $2GB$ and $3GB$ but picks up over the weekend and peaks at around $9GB$

Question 2

In this section, we will use different regression algorithms to predict the backup size of a file using all attributes apart from back time. Before we proceed, we have done scalar encoding of our data. For later section, we have also performed one-hot encoding.

(a) Linear Regression Model:
We perform a 10-fold cross validation on our data and the report the average Root Mean Squared Error (RMSE) for both the training and test sets.

- **Average Training RMSE:** 0.1035877
- **Average Testing RMSE:** 0.1036301

We have split the training and testing data in 9:1 ration and plotted the scatter plots below: (testing data is in green while training data is in blue)

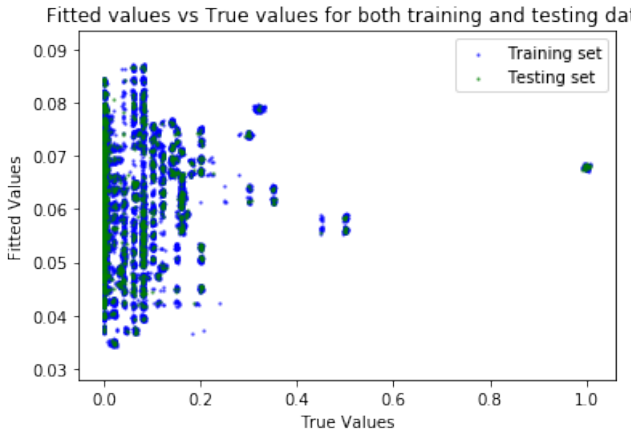

Fig. 3. Fitted values vs true values for linear regression model

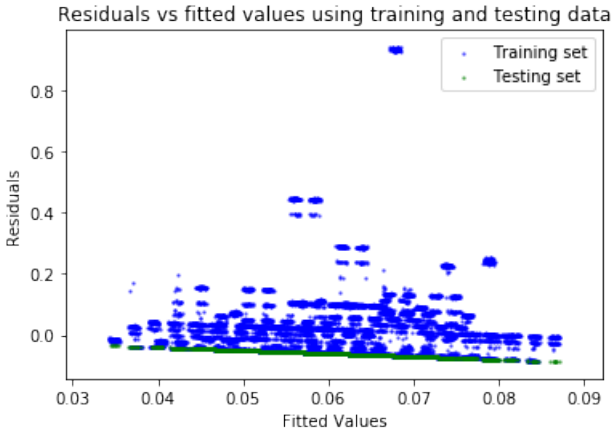

Fig. 4. Residuals vs fitted values for linear regression model

(b) Random Forest Regression
We initialize our model with the following parameters:

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

(i) Then, we have done a 10-fold cross validation of the data and obtained the followinf results:

- **Average Training RMSE:** 0.0606359
- **Average Testing RMSE:** 0.0607636
- **Average Out of Bag error:** 0.3410015

(ii) Now, we sweep over number of trees from 1 to 200 and number of features from 1 to 5 and obtain the following plots:

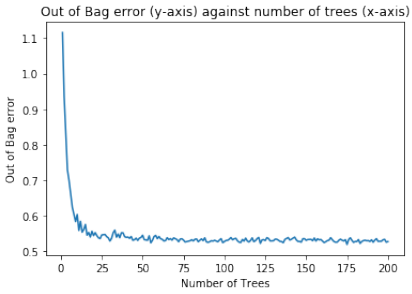

Fig. 5. out-of-bag error (y axis) against number of trees. Max number of features = 1

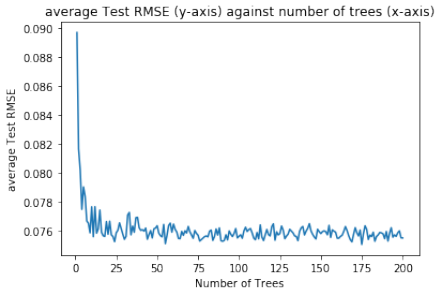

Fig. 6. average test RMSE (y axis) against number of trees. Max number of features = 1

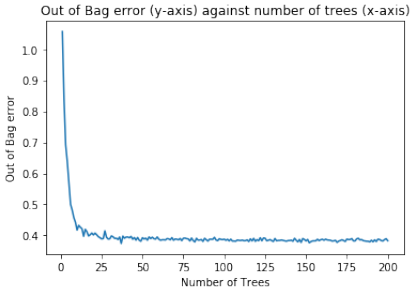

Fig. 7. out-of-bag error (y axis) against number of trees. Max number of features = 2

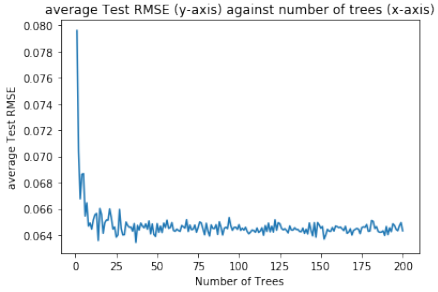

Fig. 8. average test RMSE (y axis) against number of trees. Max number of features = 2

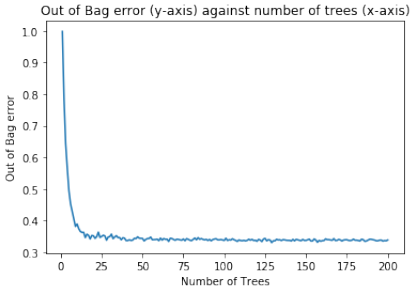

Fig. 9. out-of-bag error (y axis) against number of trees. Max number of features = 3

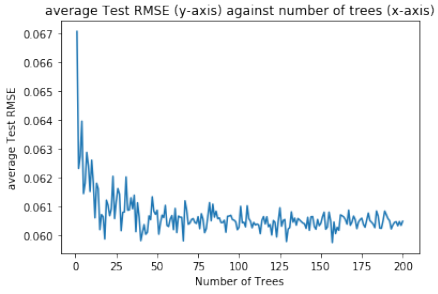

Fig. 10. average test RMSE (y axis) against number of trees. Max number of features = 3

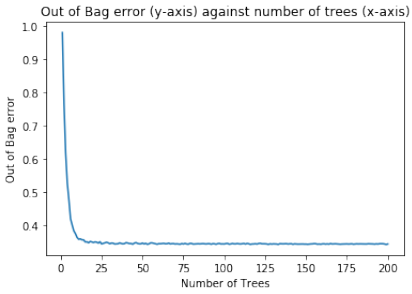

Fig. 11. out-of-bag error (y axis) against number of trees. Max number of features = 4

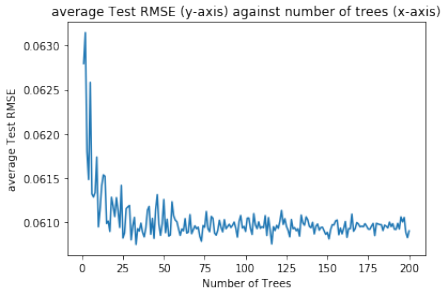

Fig. 12. average test RMSE (y axis) against number of trees. Max number of features = 4

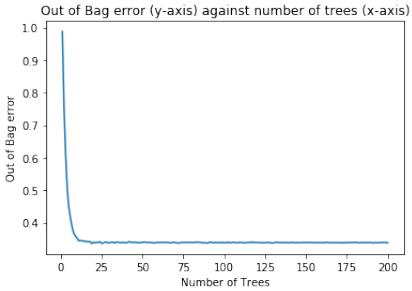

Fig. 13. out-of-bag error (y axis) against number of trees. Max number of features = 5

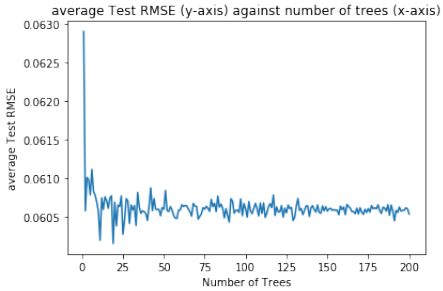

Fig. 14. average test RMSE (y axis) against number of trees. Max number of features = 5

We choose the parameters that produced the lowest value of test RMSE and the lowest value of OOB error. Based on the above results, we find that the best values of our sweeped parameters is:

- **Number of trees: 19**
- **Maximum number of features: 5**

(iii) Now, we set the best values found in the last part and sweep over another parameter: Depth of each tree. We sweep it from 1 to 40 and plot the following figures:

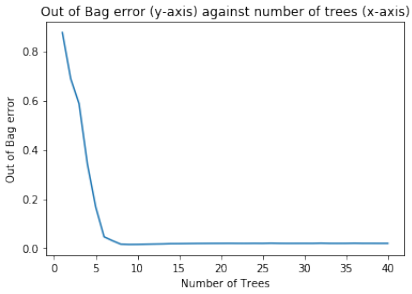

Fig. 15. out-of-bag error (y axis) against number of trees. Max number of features = 5, Number of trees = 19

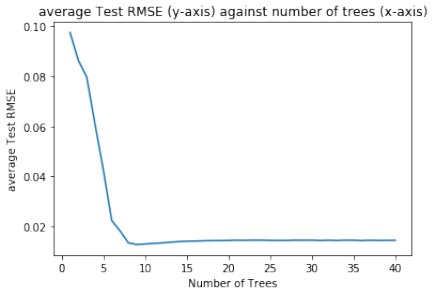

Fig. 16. average test RMSE (y axis) against number of trees. Max number of features = 5, Number of trees = 19

We choose the parameters that produced the lowest value of test RMSE and the lowest value of OOB error. Based on the above graphs, we find that in order to achieve the best performance, we should select the following values of our parameters:

- **Number of trees: 19**
- **Maximum number of features: 5**
- **Depth of each tree: 9**

For the best Random Forest Regressor, we find the following values:

- **Average Training RMSE:** 0.01156
- **Average Testing RMSE:** 0.01284
- **Average Out of Bag error:** 0.01542

We have split the training and testing data in 9:1 ration and plotted the scatter plots for our best model below: (testing data is in green while training data is in blue)

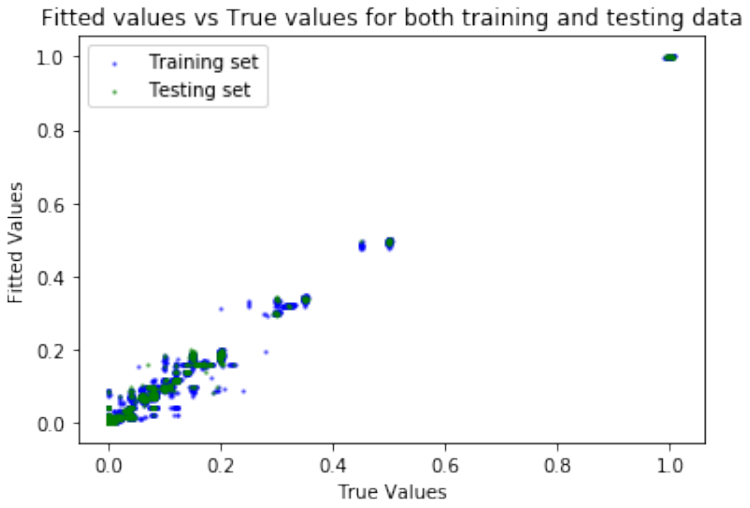

Fig. 17. Fitted values vs true values for best random forest regression model

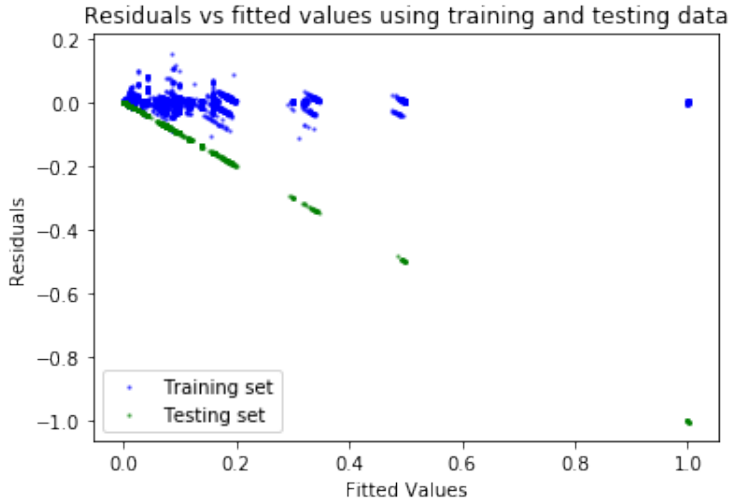

Fig. 18. Residuals vs fitted values for best random forest regression model

(iv) Feature Importances:
In this section we find feature importances for our best Random Forest Regressor:

Feature	Importance
Week #	0.00164
Day of week	0.20033
Backup Start Time - Hour of Day	0.39730
Work-Flow-ID	0.15255
File Name	0.24817

Table 1. Feature Importance for the best Random Forest Regressor

From table 1, it is clear that the most important feature is 'Backup Start Time - Hour of Day' while the least important is 'Week #'. This is in line with what we can see from the data plots, since the backup size follows a repeating pattern very week, week # is least important whereas the time at which the backup starts is most important.

(v) Even though the best depth we got was 9, for the visualization part, we have limited it to 4 as stated in the project description.

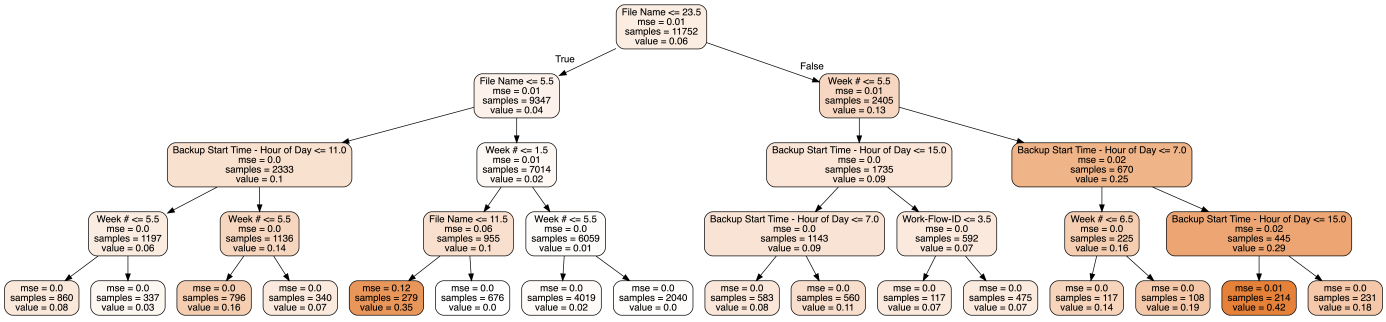

Fig. 19. Decision tree visualization

The root node of the decision tree is 'File Name'. It is not the most important but the second most important feature according to feature importance table 1. It is not necessary that the root node of a decision tree be the most important feature.

(c) Neural Network Regression

For our neural network model, we used a single layer neural network with various layer sizes and activation functions to fit our data, which we one hot encoded. From figure 20, we see that the best neural network we tested had a single hidden layer of size 400 with the ReLU activation function.

Layer size	Train RMSE (relu)	Test RMSE (relu)	Train RMSE (logistic)	Test RMSE (logistic)	Train RMSE (tanh)	Test RMSE (tanh)
2	0.0883	0.0883	0.0881	0.0892	0.0881	0.1049
5	0.1029	0.1033	0.0935	0.0948	0.0879	0.0901
10	0.0289	0.0398	0.0882	0.0900	0.0641	0.1378
50	0.0181	0.0575	0.0888	0.0916	0.0414	0.0649
100	0.0181	0.0575	0.0890	0.0923	0.0395	0.0890
150	0.0150	0.0316	0.0890	0.0908	0.0320	0.0514
200	0.0147	0.0264	0.0897	0.0946	0.0257	0.0507
250	0.0136	0.0303	0.0890	0.0906	0.0350	0.0395
300	0.0132	0.0249	0.0899	0.0925	0.0326	0.0351
350	0.0123	0.0296	0.0931	0.0971	0.0384	0.0504
400	0.0122	0.0223	0.0928	0.0969	0.0387	0.0500
450	0.0125	0.0280	0.0906	0.0926	0.0359	0.0592
500	0.0118	0.0248	0.0915	0.0939	0.0262	0.0335
550	0.0117	0.0264	0.0897	0.0930	0.0413	0.0515
600	0.0114	0.0298	0.0933	0.0936	0.0388	0.0508

Table 2. RMSE values for 3 activation functions

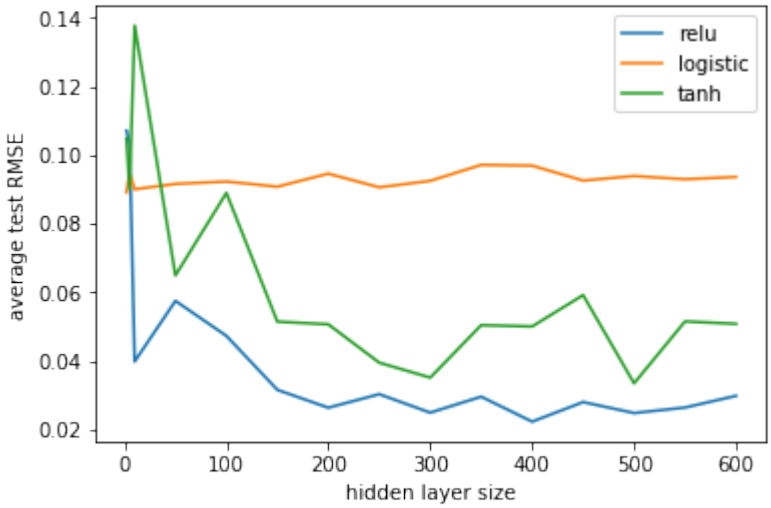

Fig. 20. RMSE vs. Size of hidden layer

(d) Now, we look at fitting each workflow individually. From Table 3, we see that all of the RMSE's are significantly lower than it was previously. Thus the

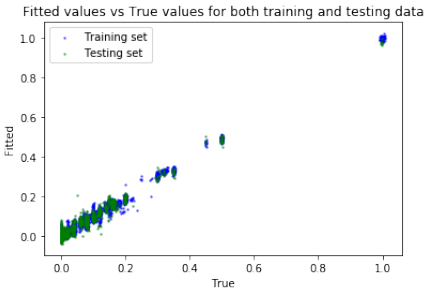

Fig. 21. Fitted vs. True, NN

Fig. 22. Residuals vs. Fitted, NN

fit is greatly improved when we fit per each workflow. For the scatter plots for each workflow, please see the Appendix.

Workflow	Train RMSE	Test RMSE
0	0.0358	0.0359
1	0.1488	0.1490
2	0.0429	0.0430
3	0.0072	0.0073
4	0.0859	0.0860

Table 3. RMSE's for each workflow, linear regression, scalar encoding

We also used polynomial feature encoding to analyze our fit. From the following figures and tables, we can see that for workflow 1, the RMSE stops decreasing for polynomial degrees greater than 8, while for the others the RMSE stops decreasing for degrees approximately greater than 6. Tables 4 and 5, along with Figures 23 and 24 describe our results.

Cross validation helps us control the complexity of our model by preventing over fitting. This is done by essentially optimizing over our hyperparameters, which in this case is the degree of the polynomial. This allows us to find the degree of our model that has the best error.

For our scatter plots for each workflow, please see the Appendix.

(e) Finally, we perform k-nearest neighbors regression. We found that the best number of neighbors is 4, as shown in Table 6 and Figure 25.

Question 3

We found that the neural network was able to classify the one-hot encoded data the best out of all of the networks. We tried using KNN regression and polynomial features along with one hot, but this did not yield promising results. We noticed

Degree	Workflow 0	Workflow 1	Workflow 2	Workflow 3	Workflow 4
2	0.0295	0.1298	0.0385	0.0064	0.0692
3	0.0263	0.1110	0.0345	0.0060	0.0609
4	0.0249	0.0899	0.0319	0.0055	0.0489
5	0.0182	0.0655	0.0259	0.0049	0.0360
6	0.0104	0.0428	0.0216	0.0046	0.0258
7	0.0091	0.0207	0.0210	0.0044	0.0208
8	0.0079	0.0081	0.0195	0.0043	0.0184
9	0.0076	0.0056	0.0184	0.0042	0.0160
10	0.0073	0.0052	0.0170	0.0040	0.0144

Table 4. Train RMSE's for polynomial fit

Degree	Workflow 0	Workflow 1	Workflow 2	Workflow 3	Workflow 4
2	0.0300	0.1309	0.0386	0.0064	0.0695
3	0.0267	0.1131	0.0348	0.0061	0.0614
4	0.0256	0.0928	0.0325	0.0055	0.0498
5	0.0190	0.0686	0.0267	0.0050	0.0374
6	0.0111	0.0457	0.0227	0.0048	0.0276
7	0.0100	0.0227	0.0227	0.0047	0.0231
8	0.0093	0.0092	0.0220	0.0048	0.0215
9	0.0097	0.0069	0.0219	0.0049	0.0205
10	0.0102	0.0071	0.0224	0.0052	0.0207

Table 5. Test RMSE's for polynomial fit

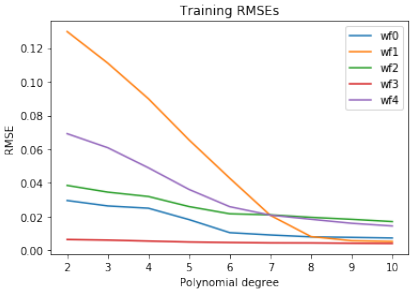

Fig. 23. Train RMSE for polynomial fit

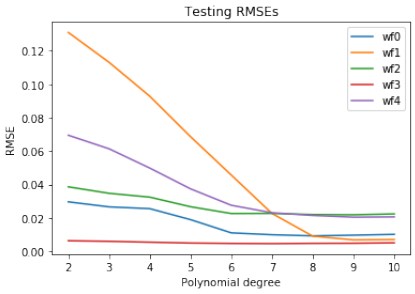

Fig. 24. Test RMSE for polynomial fit

# of neighbors	Train RMSE	Test RMSE
2	0.0298	0.0427
3	0.0293	0.0357
4	0.0280	0.0340
5	0.0274	0.0371
6	0.0297	0.0406
7	0.0336	0.0432
8	0.0370	0.0461
9	0.0398	0.0479
10	0.0419	0.0494

Table 6. RMSE's from k-neighbors regression

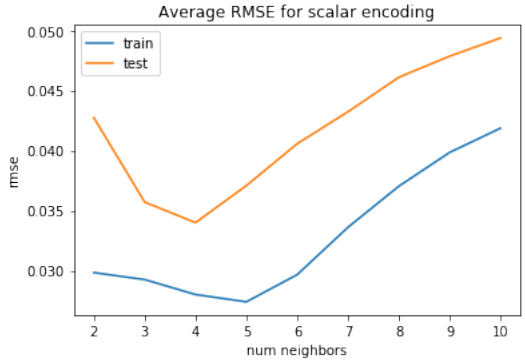

Fig. 25. RMSE for k-neighbors regression

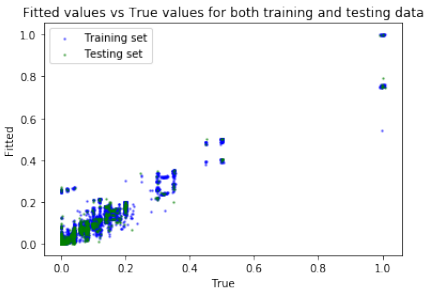

Fig. 26. Fitted vs True, KNN

Fig. 27. Residuals vs. Fitted, KNN

that performing one hot encoding along with polynomial features did not work well, especially with many unique features as the dimension of the problem grows immensely. For scalar encoding, we found that KNN regression outperformed the linear regression model significantly on the entire dataset. The random forest is very good at handling categorical features, since it is not necessary to have one hot or scalar encoding of the inputs. Overall, the random forest performed the best on the entire data set, as it achieved a testing RMSE of ∼0.013, which is significantly better than the other models.

Random Forest performs the best as it is very robust to overfitting and can handle categorical features without the need of scalar or one hot encoding.

Dataset 2

The Boston Housing dataset contains information about the suburbs of the greater Boston area. This information is taken by the Stat Library and maintained by Carnegie Mellon University. There are a little over 500 data points with features such as crime rate, age statistics, nearby employment centers, and price-related information. Our task is to predict the median value of owner-occupied homes (MEDV) given the other parameters of the area in question.

Question 2

(a) MEDV was set as the target variable, and ordinary least squares was used as the penalty function. The training set therefore consists of the data found in Fig. 28, and the desired output consists of the data found in Fig. 29

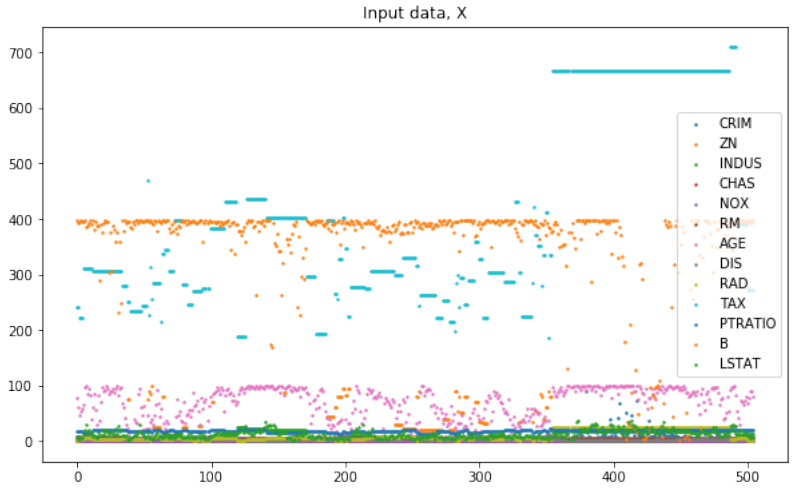

Fig. 28. Boston Housing dataset, training data

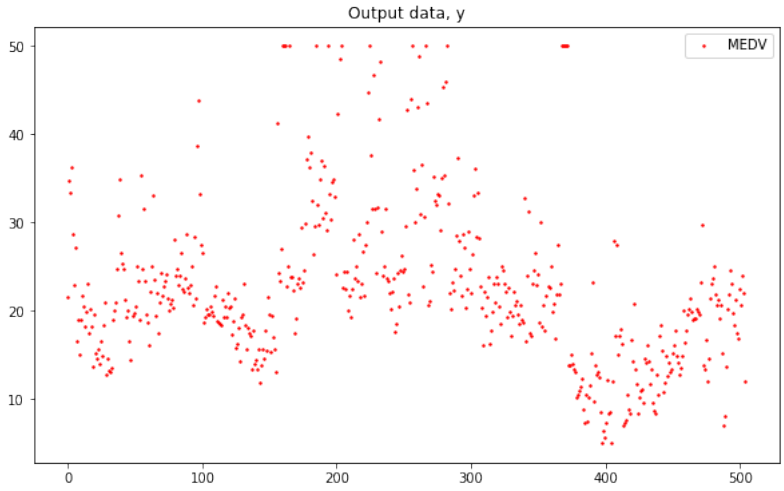

Fig. 29. Boston Housing dataset, desired output

(b) The significance of different variables for the linear regression model was analyzed using null-hypothesis testing and 10 fold cross validation. Table 7 shows the values from this testing. The significance of each variable was also determined by removing a variable from training, and measuring the new test RMSE. The higher the new RMSE, the more significant the variable was to the model during training. Table 8 shows these results. It is intuitive that the significance ranking of each variable from Table 7 match the ranking in Table 8. **LSTAT, RM, and DIS are the three most significant variables.** INDUS and AGE have very high p values, indicating that they are the least significant variables.

Scatter plots The scatter plot showing the fitted values vs true values for the Linear Regression model with no regularization is shown in Fig. 30. The scatter plot showing the residuals vs fitted values for the Linear Regression model with no regularization is shown in Fig. 31

Question 3

It is noticeable from the previous two figures that there are outliers in the test set which perform very poorly with respect to the training set. The model is therefore overfitting the dataset. To prevent this overfitting, regularization is performed on the Linear Regression model. **The train and test RMSE can be found in Table 9.**

(a.1) The Ridge Regularizer is used to regularize the dataset. The Ridge Regularizer has the following objective function:

Variables	Coefficients	Standard Errors	t values	p values
CRIM	-0.1074	0.033	-3.270	0.001
ZN	0.0461	0.014	3.361	0.001
INDUS	0.0143	0.062	0.231	0.817
CHAS	2.6711	0.861	3.102	0.002
NOX	-17.6336	3.819	-4.618	0.000
RM	3.7943	0.418	9.081	0.000
AGE	0.0011	0.013	0.081	0.935
DIS	-1.4792	0.199	-7.420	0.000
RAD	0.3015	0.066	4.541	0.000
TAX	-0.0121	0.004	-3.202	0.001
PTRATIO	-0.9589	0.131	-7.329	0.000
B	0.0093	0.003	3.467	0.001
LSTAT	-0.5276	0.051	-10.400	0.000

Table 7. Null hypothesis testing on the Boston Housing dataset

Variable removed	Test RMSE
CRIM	4.894
ZN	4.888
INDUS	4.851
CHAS	4.882
NOX	4.962
RM	5.227
AGE	4.839
DIS	5.108
RAD	4.939
TAX	4.885
PTRATIO	5.090
B	4.902
LSTAT	5.330

Table 8. Average Test RMSE when removing each variable

$$\min_{\beta} \|Y - X\beta\|^2 + \alpha\|\beta\|_2^2 \tag{1}$$

Fig. 32 shows the average RMSE on the test set for different α values. In order to prevent overfitting, the best alpha is one which reduces the gap between training and testing RMSE but also keeps the RMSE as low as possible. Based on this criterion, $\alpha = 10000$ was chosen. **The train and test RMSE can be found in Table 9.**

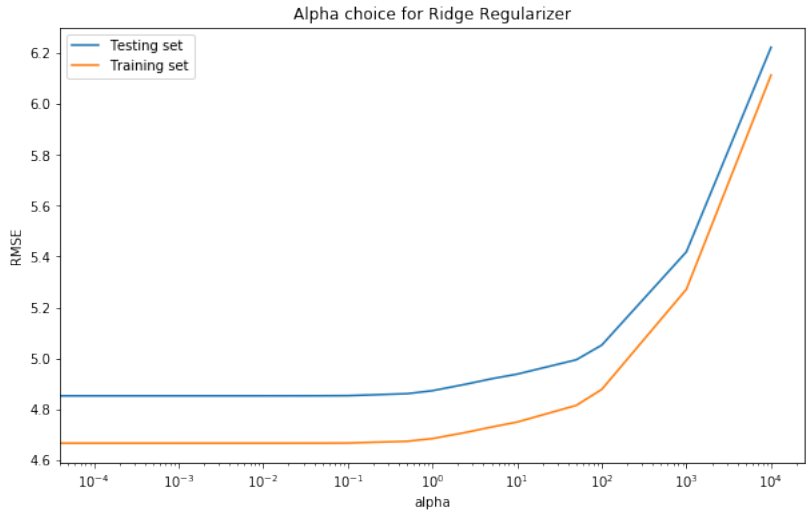

Fig. 32. RMSE vs α for Ridge Regularizer

Fig. 30. Fitted vs True scatter plot for vanilla Linear Regression model

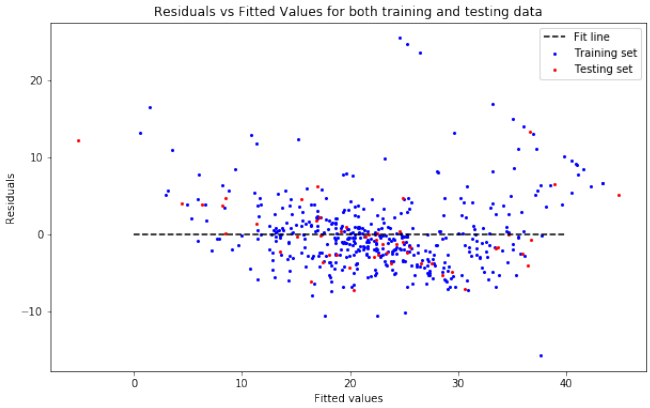

Fig. 31. Residuals vs Fitted scatter plot for vanilla Linear Regression model

(a.2) The Lasso Regularizer is now used to regularize the dataset. The Lasso Regularizer has the following objective function

$$\min_{\beta} \|Y - X\beta\|^2 + \alpha\|\beta\|_1 \tag{2}$$

Fig. 33 shows the average RMSE on the test set for different α values. In order to prevent overfitting, the best alpha is one which reduced the gap between training and testing RMSE but also keeps the RMSE as low as possible. Based on this criterion, $\alpha = 10$ was chosen. **The train and test RMSE can be found in Table 9.**

(a.3) The Elastic Net Regularizer is now used to regularize the dataset. The Elastic Net Regularizer has the following objective function. This objective function has two hyperparameters, λ_1 and λ_2:

$$\min_{\beta} \|Y - X\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2 \tag{3}$$

Fig. 34 shows the distance between the train and test RMSE when a model is fit using particular λ_1 (a) and λ_2 (b) values. In order to prevent overfitting, the best λ_1 and λ_2 are ones which minimize the distribution in Fig. 34. Based on this criterion, $\lambda_1 = 100$ and $\lambda_2 = 10$ were chosen. **The train and test RMSE can be found in Table 9.**

(a.4) Now that all of the regularizers have been implemented, the coefficients of β can be compared. Statistics for each β per linear regression model are shown in Table 9. It is clear that the initial minimum, maximum, and variance measurements for the parameter β are much larger than when regularization is

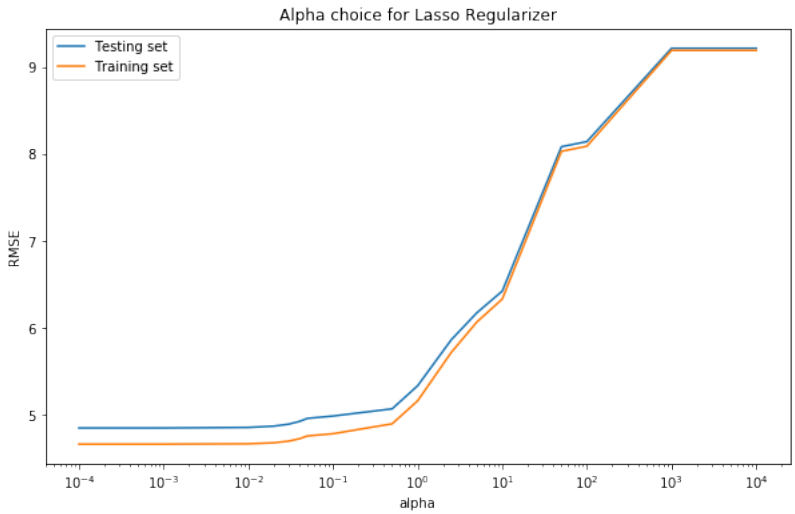

Fig. 33. RMSE vs α for Lasso Regularizer

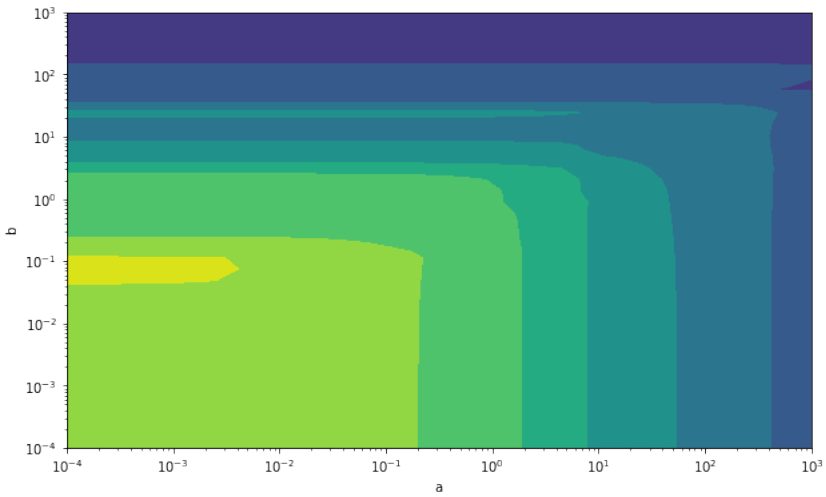

Fig. 34. Distance between train and test RMSE vs λ_1 (x-axis) and λ_2 (y-axis) for Elastic Net Regularizer

performed. This supports the idea that the regularization is keeping the magnitudes of the weights low, and preventing overfitting by making the network more resilient to small changes in X.

Regularization scheme	β min	β max	β variance	Train RMSE	Test RMSE
LR with no regularizer	-16.283	3.698	21.828	4.67	4.85
LR with Ridge Regularizer	-0.512	0.136	0.0242	6.13	6.22
LR with Lasso Regularizer	-0.583	0.020	0.0243	6.34	6.43
LR with Elastic Net Regularizer	-0.022	0.004	3.5e-05	8.09	8.15

Table 9. β values and RMSE per regularization scheme

The regularization hyperparameters that do not over-fit the dataset will naturally increase the train and test RMSE. In fact, if RMSE was being optimized instead, the optimal hyperparameter in each regularizer would be zero - suggesting that regularization does not help reduce the test or train RMSE by a significant amount.

This is why the hyperparameters were chosen based on how close the train and test RMSE were to one another. By using regularization to match the training and test RMSE, you will affect both scores simultaneously. This why for all three regularization cases, the RMSE is much higher than the original RMSE of Linear Regression without regularization.

To demonstrate, we compare the new scatter plots of fitted vs true values for each regularization scheme to the original. Fig. 35, Fig. 36, and Fig. 37 show the impact of regularization on the dataset. It is evident that the Ridge and Lasso Regularizers restrict the fitted values from matching their true counterpart if the true value is too high. In other words, the magnitude of the coefficients in β are too low to represent these high housing prices. The elastic regularization presents the extreme case, where the model is able to have very similar test and train RMSEs, but the RMSE is very high and the model no longer accurately fits the data. By regularizing, these models are focusing more on precision than the accuracy of their estimates.

Dataset 3

(a) Dataset 3 contains information regarding car insurance users' features, such as gender and geographic location, and the charges corresponding to each user. The dataset contains three numerical features (ft 1, ft 2, and ft3) and three categorical features (ft 4, ft 5, and ft6). Figure 38 is a plot of the complete car insurance dataset. Figure 39 is a plot of the inputs of the car insurance dataset.

1 Feature Pre-processing

(a) One-hot Feature Encoding One-hot encoding is used to convert the three categorical features (ft 4, ft 5, and ft6) to numerical values that can be used in

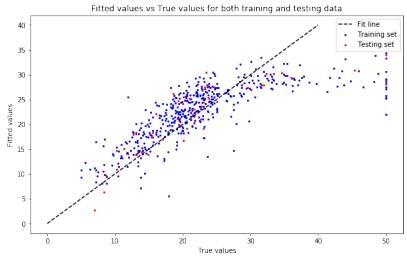

Fig. 35. Fitted vs True scatter plot for Ridge Regularization LR model

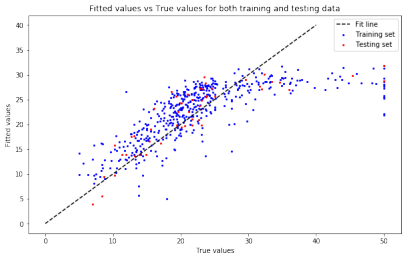

Fig. 36. Fitted vs True scatter plot for Lasso Regularization LR model

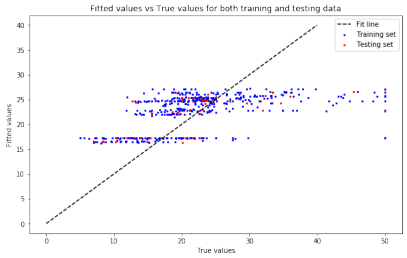

Fig. 37. Fitted vs True scatter plot for Elastic Regularization LR model

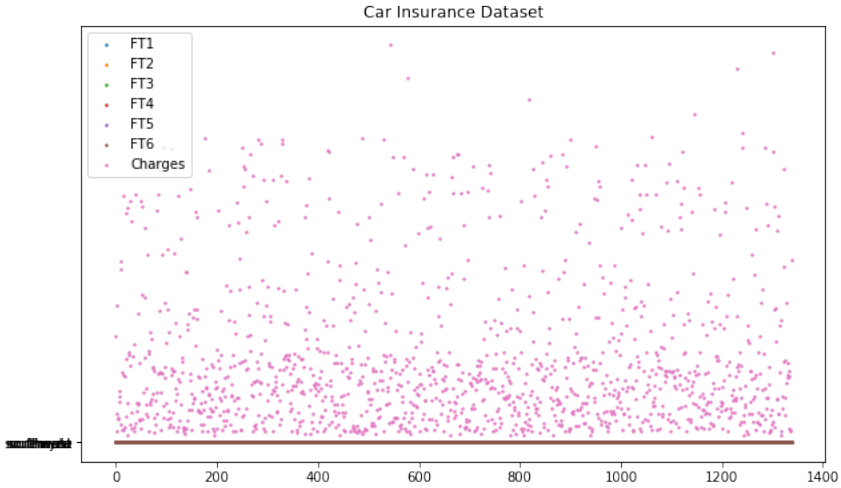

Fig. 38. Car Insurance Dataset

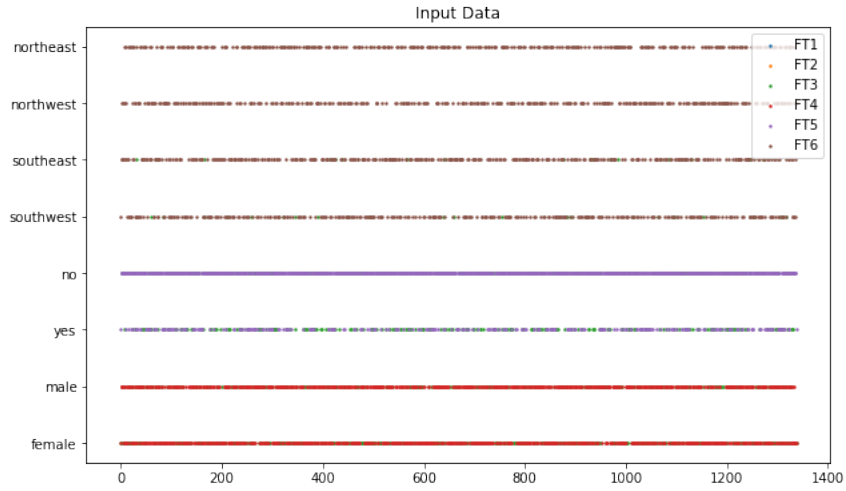

Fig. 39. Car Insurance Input data

machine learning algorithms. As the number of unique categorical entries increases, the number one-hot encoded numerical values proportionally increases. One-hot encoding eliminates the problem of assigning various numerical values to categorical features. For example scalar encoding might assign southwest 1, southeast to 2, and northeast to 3. Thus, if a model calculated the average of southwest and northeast ($\frac{1+3}{2} = 2$), it would return southeast, which doesn't make sense. Consequently, one-hot encoding performs a binary transformation of categorical features to avoid incorrect computational errors.

We use the one-hot encoded features and the unadulterated numerical features to fit a linear regression model to predict the users' insurance charges. Figure 40 represents the fitted values vs the true values for the whole data set. Figure 41 represents the residual values (true - fitted value) vs fitted values for the whole data set.

For one-hot Encoding:

- **Our average training RMSE was 6039.2962**
- **Our average test RMSE was 6085.6263**

(b) Standardization Feature Encoding Standardization is used to standardize all the numerical features (ft 1, ft2, and ft3). Standardization removes the mean of numerical data features and scales them to unit variance. Standardization makes numerical data normally distributed, which is a requirement for several machine learning algorithms.

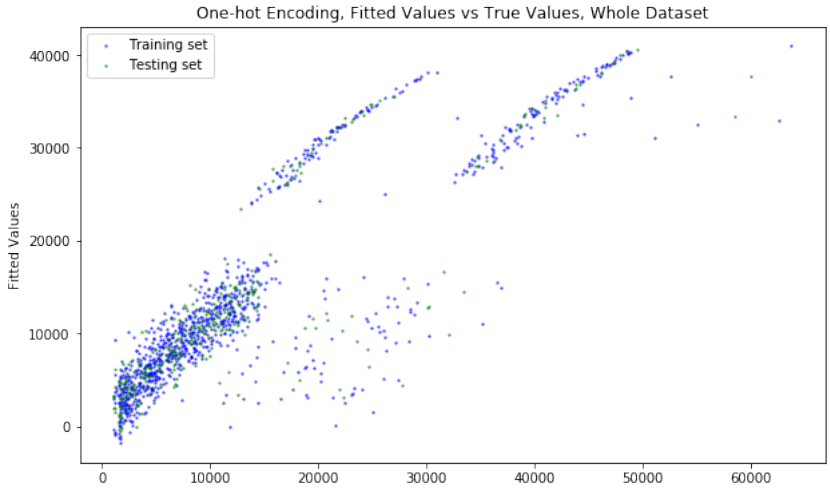

Fig. 40. One-hot Encoding, Fitted Values vs True Values

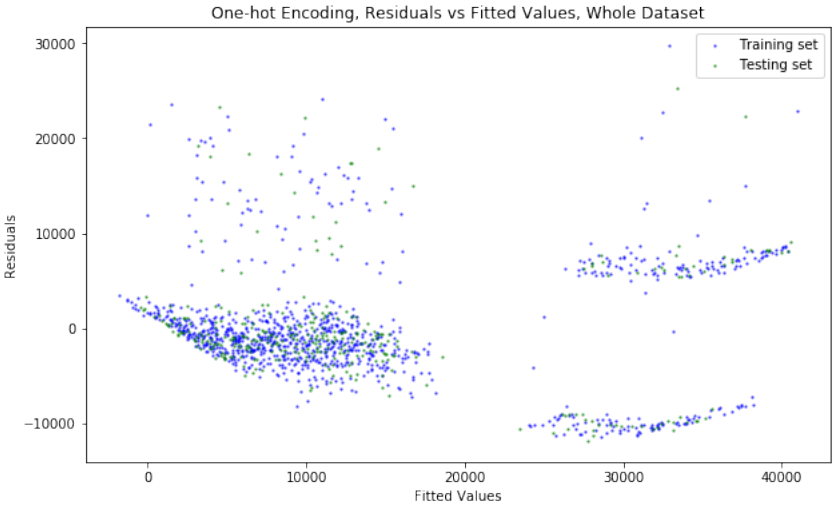

Fig. 41. One-hot Encoding, Residuals vs Fitted Values

We used the one-hot encoded features and the standardized numerical features to fit a linear regression model to predict the users' insurance charges. Figure 42 represents the fitted values vs the true values for the whole data set. Figure 43 represents the residual values (true - fitted value) vs fitted values for the whole data set.

For standardization:

- **Our average training RMSE was 6043.8651**
- **Our average test RMSE was 6093.1725**

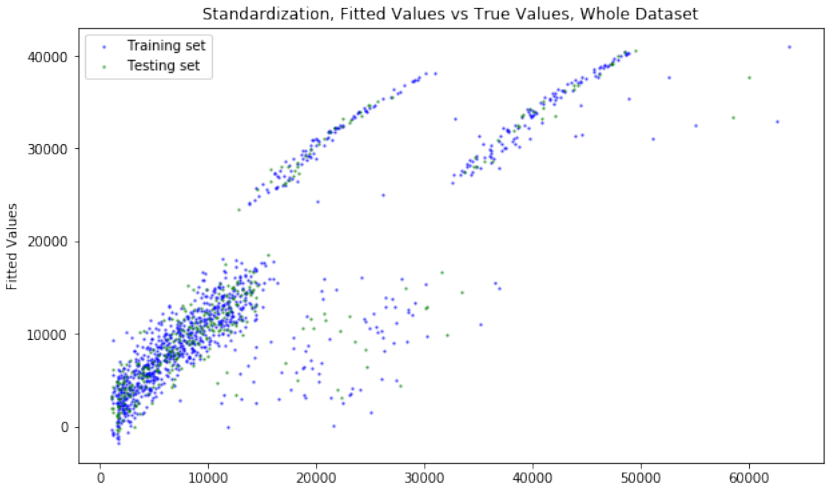

Fig. 42. Standardization, Fitted Values vs True Values

(c) Division Feature Encoding We divide ft 1 into three ranges: $< 30, [30, 50]$and $>$ 50 and set the new values to 1 for original values below 30, 2 for values between 30 and 50 and 3 for values above 50. We also standardize ft 2 and ft 3 and use one-hot encoding for the three categorical features.

Figure 44 represents the fitted values vs the true values for the whole data set. Figure 45 represents the residual values (true - fitted value) vs fitted values for the whole data set.

For Division:

- **Our average training RMSE was 6201.8464**
- **Our average test RMSE was 6239.2709**

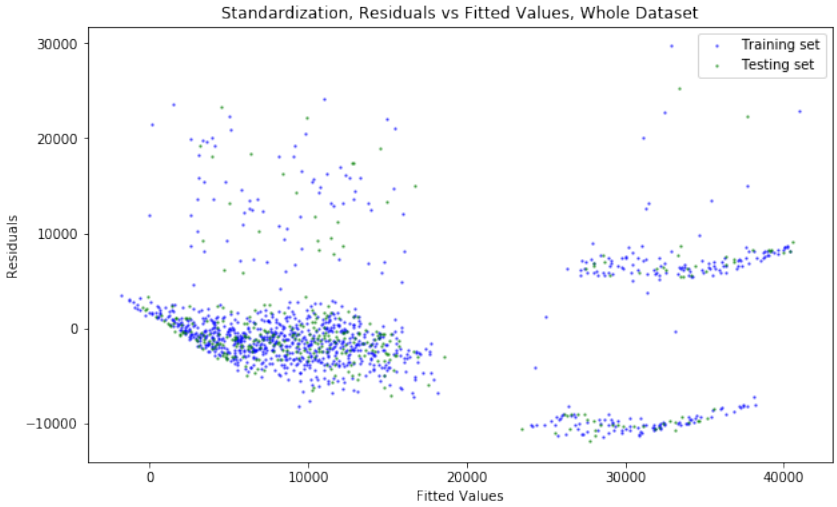

Fig. 43. Standardization, Residuals vs Fitted Values

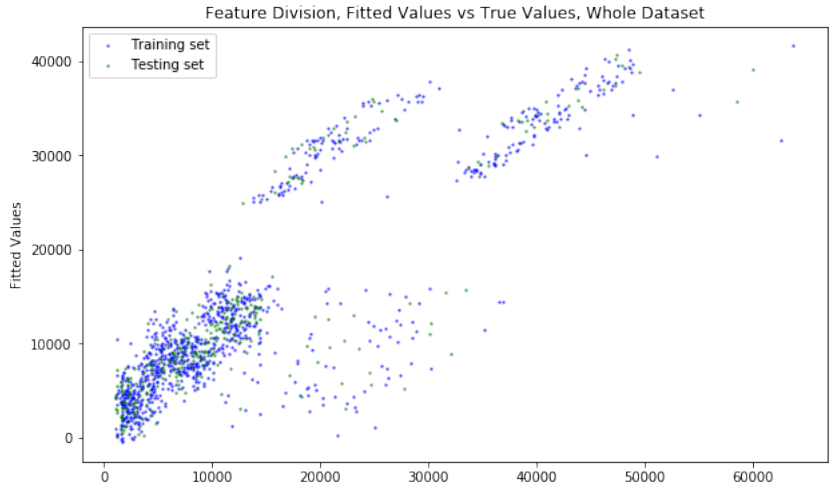

Fig. 44. Division, Fitted Values vs True Values

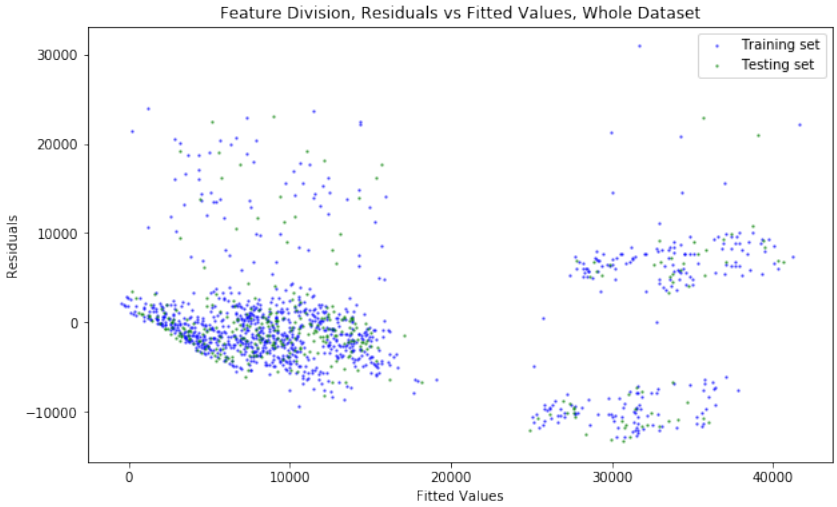

Fig. 45. Division, Residuals vs Fitted Values

2 Correlation Exploration

(a) One-Dimension Numerical Values: We convert each categorical feature into a one dimensional numerical value using the aforementioned techniques so that we have 6 numerical features. **ft 1 and ft 5 yielded the two highest f_regression scores of 131.174 and 2177.6149, respectively. ft 1 and ft 5 yielded the two highest mutual information scores of 1.5007 and 0.3692, respectively. We conclude that ft 1 and ft 5 are our two most important variables.**

Figure 46 is a scatter plot of users' insurance charges vs. ft 2. The points are color coded blue when ft 5 is yes and red when ft 5 is no. Figure 47 is a scatter plot of users' insurance charges vs. ft 1. The points are color coded blue when ft 5 is yes and red when ft 5 is no. We notice in both plots that charges are distinctly higher when ft 5 = yes, supporting our previous conclusion that ft 5 is important. Additionally, charges increase as ft 1 increases, supporting our conclusion that ft 1 is important.

3 Modifying the Target Variable

The insurance charges (y) have a wide range, indicating that transforming their scale might improve performance. Consequently, instead of fitting (y), we try fitting $\log(y)$.

(a) Training with $\log(y)$ We use the one-hot encoded features and the unadulterated numerical features to fit a linear regression model to predict the log of the users' insurance charges ($\log(y)$). Figure 48 represents the fitted values vs

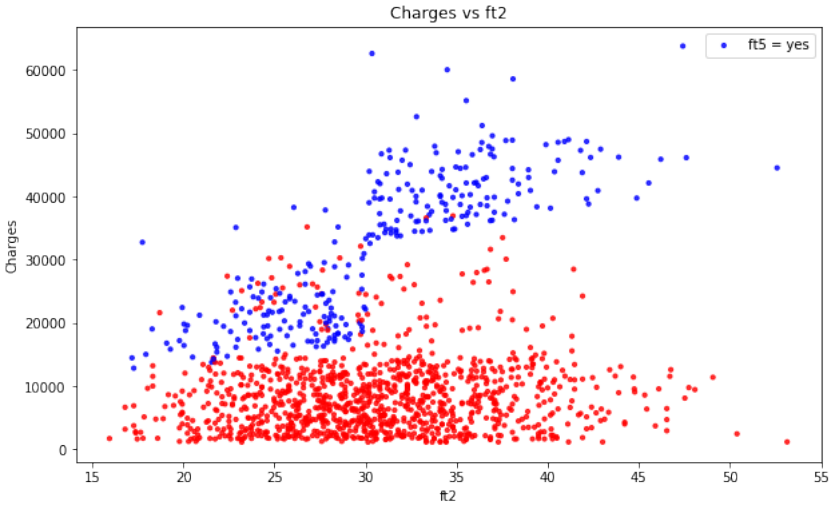

Fig. 46. Charges vs ft 2

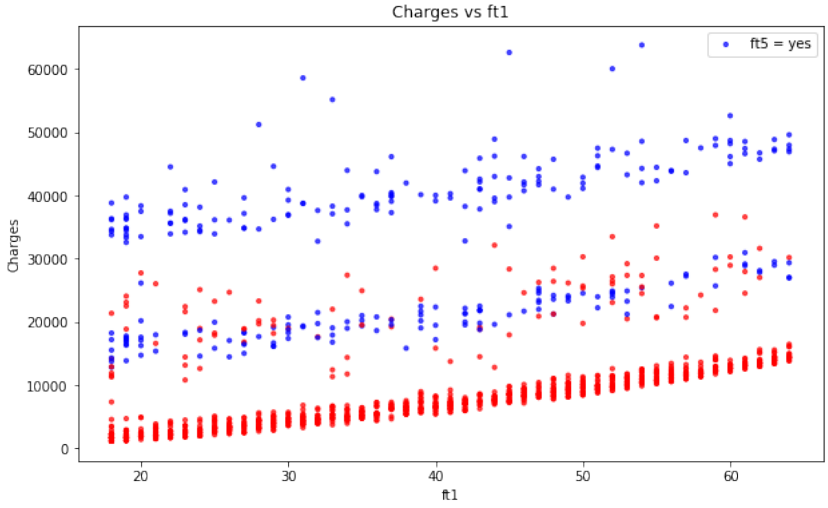

Fig. 47. Charges vs ft 1

the true values for the whole data set. Figure 49 represents the residual values (true - fitted value) vs fitted values for the whole data set.

For fitting to log(y):

- **Our average training RMSE was 8123.1295**
- **Our average test RMSE was 8228.7363**

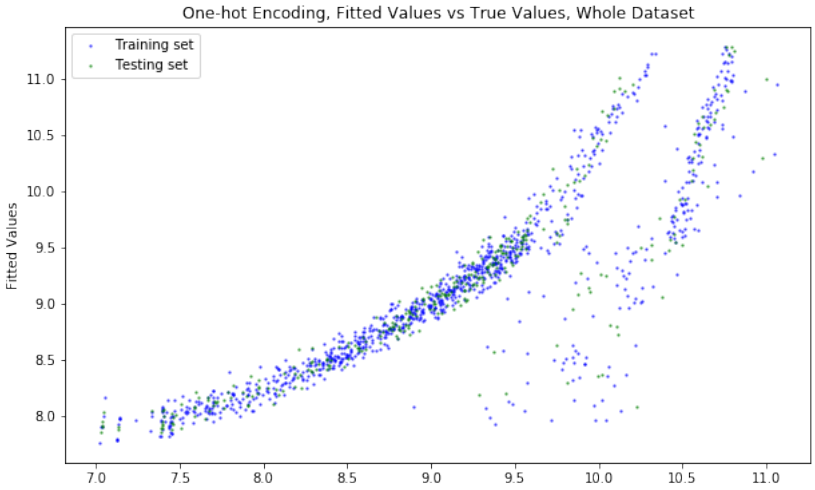

Fig. 48. Log, One-hot Encoding, Fitted Values vs True Values

(b) Correlation Exploration We convert each categorical feature into a one dimensional numerical value using the aforementioned techniques so that we have 6 numerical features. **ft 1 and ft 5 yielded the two highest f_regression scores of 515.9771 and 1062.1239, respectively. ft 1 and ft 5 yielded the two highest mutual information scores of 1.5003 and 0.3694, respectively. We conclude that ft 1 and ft 5 are our two most important variables.**

Figure 50 is a scatter plot of users' insurance log charges vs. ft 2. The points are color coded blue when ft 5 is yes and red when ft 5 is no. Figure 51 is a scatter plot of users' insurance log charges vs. ft 1. The points are color coded blue when ft 5 is yes and red when ft 5 is no. We notice in both plots that charges are distinctly higher when ft 5 = yes, supporting our previous conclusion that ft 5 is important. Additionally, charges increase as ft 1 increases, supporting our conclusion that ft 1 is important.

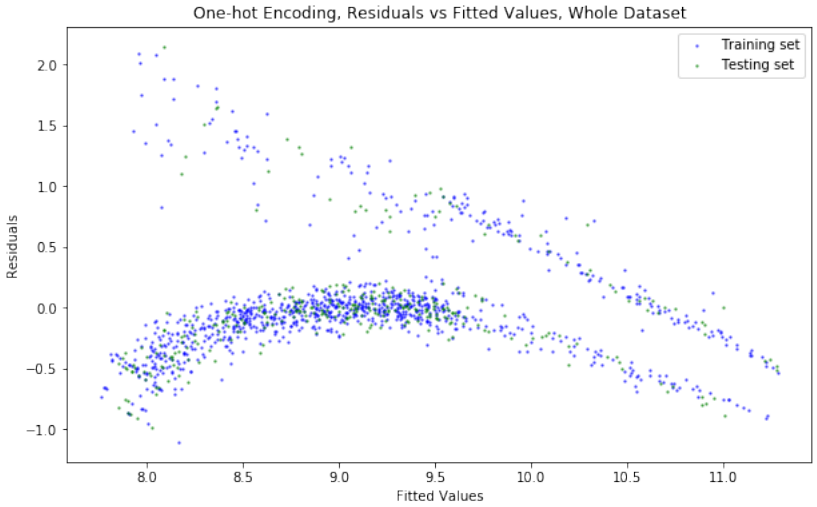

Fig. 49. Log, One-hot Encoding, Residuals vs Fitted Values

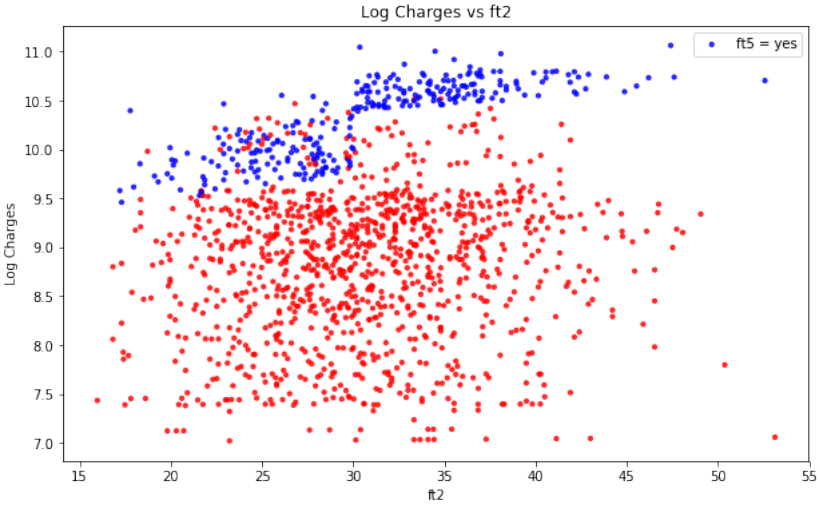

Fig. 50. Log Charges vs ft 2

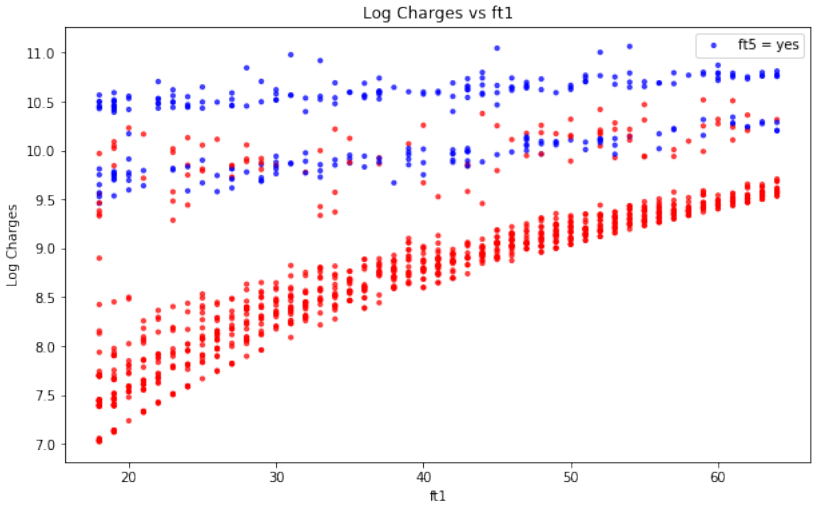

Fig. 51. Log Charges vs ft 1

4 Bonus Questions

(a) Feature Encoding: In this section we implemented polynomial feature encoding twice, once using one-hot encoded features and once using standardized features.

Figure 52 is a plot of the test and training RMSE vs the polynomial degree for one-hot encoded features. Figure 53 is a plot of the test and training RMSE vs the polynomial degree for standardized features. In both polynomial encodings, using polynomial degree 2 produced the lowest test RMSE (5907.5305 and 4882.7132, respectively). Additionally, although the training RMSE decreases as the polynomial degree increases, the test RMSE significantly increases. Thus, it appears that implementing polynomial encoding leads to overfitting as the polynomial degree increases. However, **using low degree polynomial encoding does improve the model performance.**

(b) Different Models We used three different models to try to improve our fitting performance: a **Deep Neural Network, KNN-regression, and Linear Regression with Lasso Regularization.**

Deep Neural Network: The Deep Neural Network following the structure found in Fig. 54 is constructed using TensorFlow. The network uses 3 Fully-connected layers with 100 neurons each. It was found that by making the network any deeper or wider, the loss would flatline and no learning would occur.

The best combination of encoding scheme, optimizers, batch size, epochs, and weight regularization are presented in Table 10. Combinations which produced

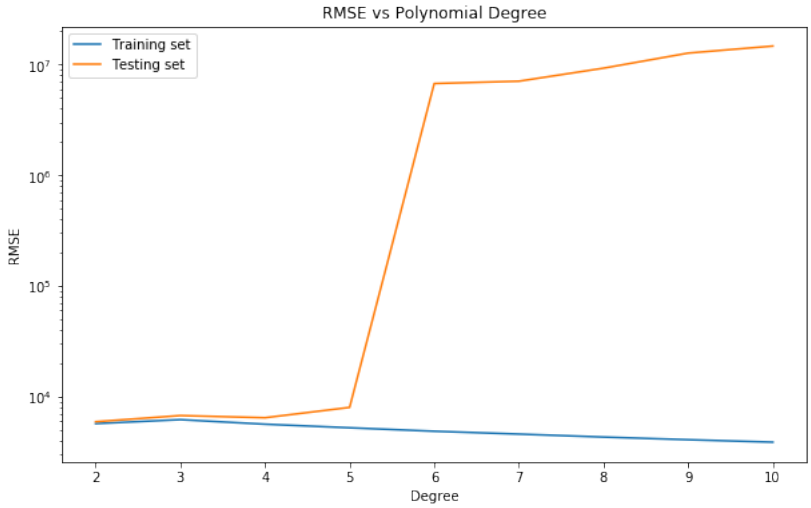

Fig. 52. RMSE vs Degree, Polynomial with one-hot encoding

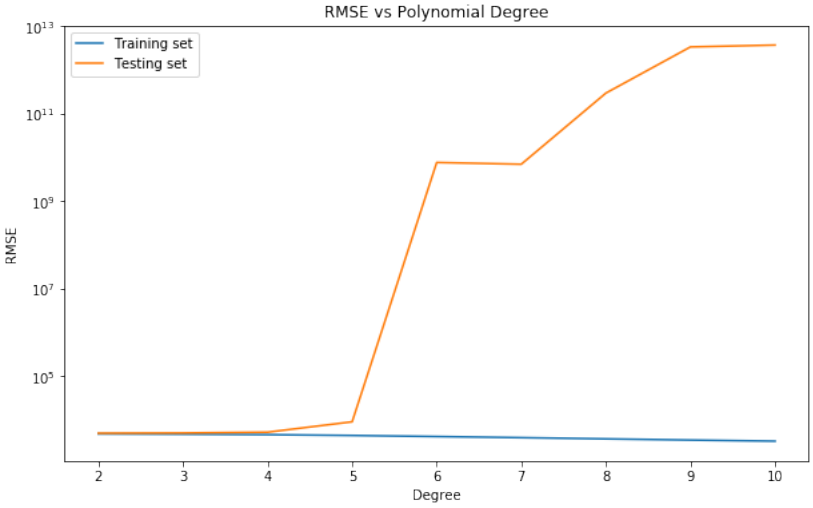

Fig. 53. RMSE vs Degree, Polynomial with standardization

extremely high RMSE values were not included in the table. The Deep Neural Network designed with the best combination of hyperparameters outperforms the other models that we tried with a testing RMSE of 4490.70.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 11)                132
_________________________________________________________________
dense_1 (Dense)              (None, 100)               1200
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 101
=================================================================
Total params: 21,633
Trainable params: 21,633
Non-trainable params: 0
```

Fig. 54. Deep Neural Network structure

Encoding	Optimizer	Batch size	Epochs	Dropout	Batch Norm	Test RMSE
One-hot, standardized, scalar	Adam	100	2000	0.1	No	4490.70
One-hot, standardized, scalar	Adam	100	2000	0.0	No	4670.14
One-hot, standardized	Adam	100	2000	0.0	No	4682.26
One-hot	Adam	100	2000	0.0	No	4780.84
One-hot, standardized, scalar	RMSProp	100	2000	0.0	Yes	4813.15
One-hot, standardized, scalar	Adam	100	2000	0.1	No	4934.49
One-hot, standardized, scalar	Adam	100	2000	0.0	Yes	6772.45

Table 10. Best combinations for the Deep Neural Network

KNN Regression: We implemented KNN Regression using the one-hot encoded features. We spanned the number of neighbors from 1 to 10. Figure 55 represents the fitted values vs the true values for the whole data set. Figure 56 represents the residual values vs fitted values for the whole data set. For each number of neighbors, the test RMSE was 10,000, while the training RMSE grew logarithmically from 500 to 9,500. The KNN regression model does not improve our results, since the test RMSE is significantly higher than other models.

Linear Regression with Lasso Regularization: We implemented linear regression with lasso regularization for a logarithmic range of α, ranging from

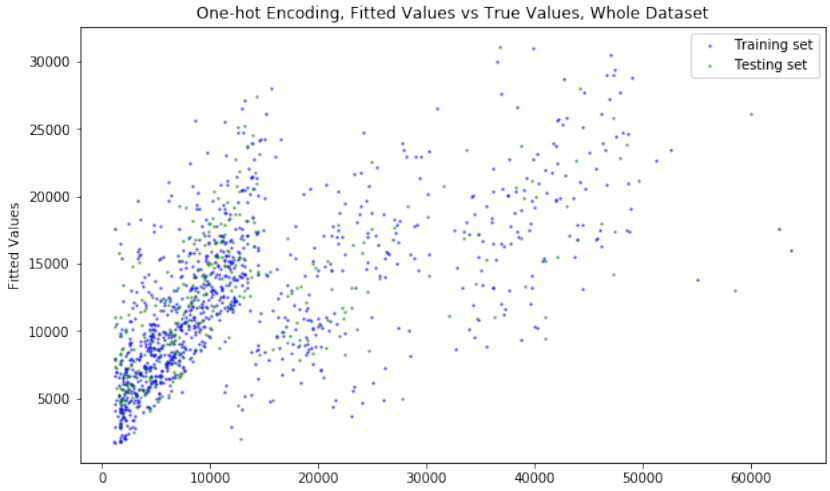

Fig. 55. KNN Regression, One-hot Encoding, Fitted Values vs True Values

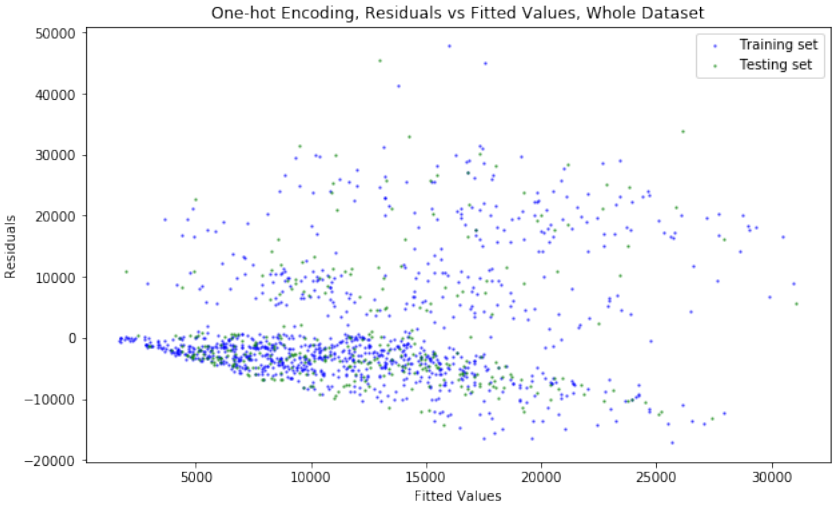

Fig. 56. KNN Regression, One-hot Encoding, Residuals vs Fitted Values

0.0001 to 10,000 on the one-hot encoded features. Figure 57 plots the test and training RMSE vs the range of alphas we used. The testing RMSE was lowest with a value of 6081.3150 when $\alpha = 10$. Although adding lasso regularization slightly improves the performance of the linear regression model, the improvement is essentially negligible, suggesting that un-regularized linear regression is just as effective. As is evidenced in the plot, the training and testing RMSE are very close for all alpha values. Figure 58 represents the fitted values vs the true values for the whole data set. Figure 59 represents the residual values vs fitted values for the whole data set.

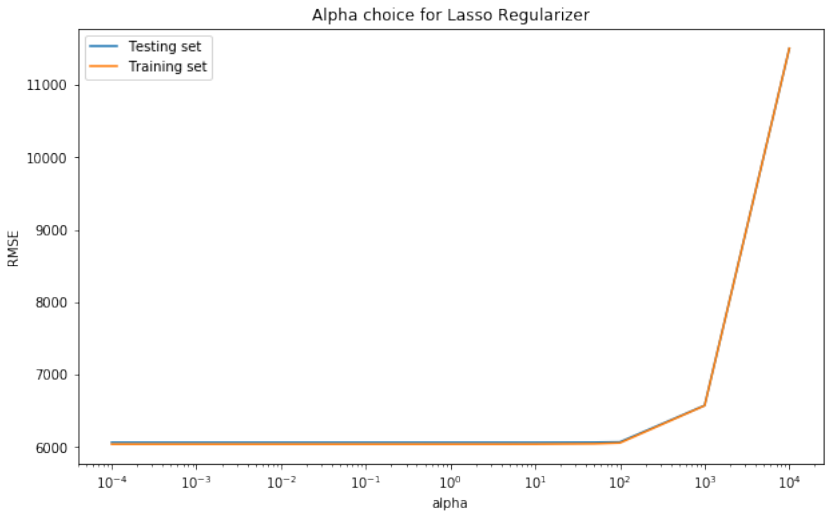

Fig. 57. RMSE vs Alpha

Conclusion

A total of three datasets were used as bases for regression problems.

From Dataset 1 we notice that the best-performing models were the random forest and single layer neural network, while trying to perform linear regression with various encoding on the entire dataset did not yield yield good results, comparatively. However, when we further analyzed each individual workflow, we found that our linear models performed very well. This can be attributed to the fact that each workflow is highly correlated, and thus a simple linear model will perform well. Thus we can say that simple linear models can be used for correlated portions of a dataset, while more complex and nonlinear sets should be fitted with robust models such as random forests or neural networks.

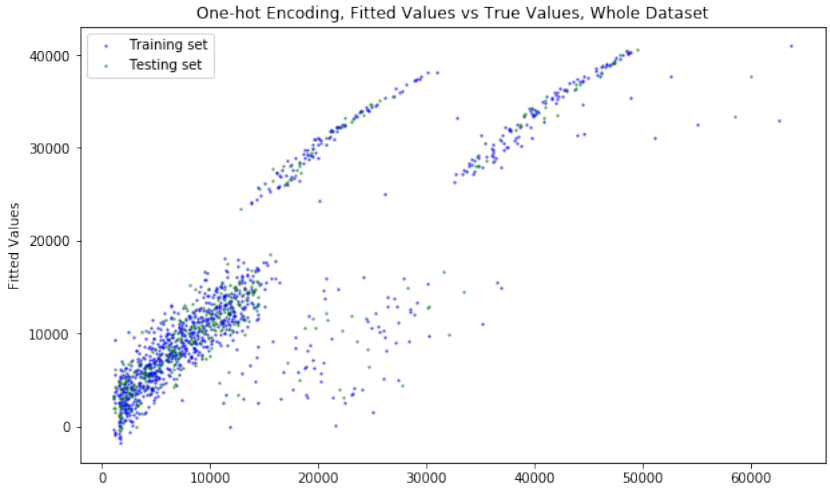

Fig. 58. LR with Lasso, One-hot Encoding, Fitted Values vs True Values

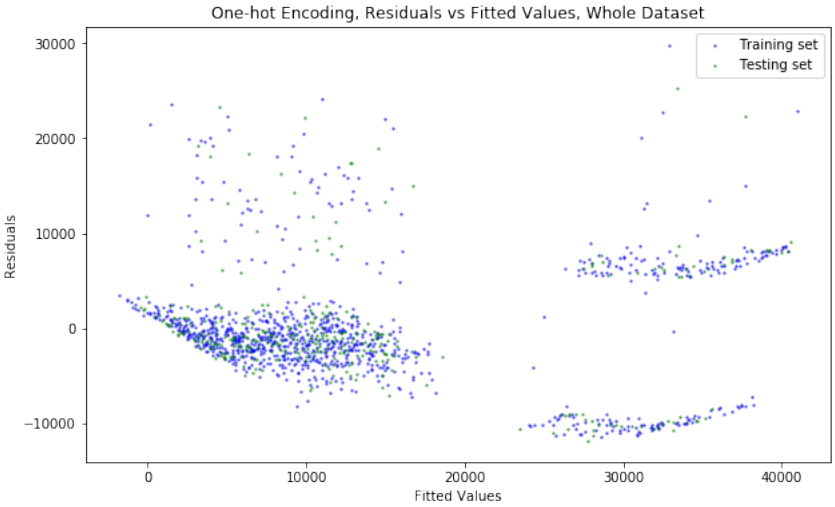

Fig. 59. LR with Lasso, One-hot Encoding, Residuals vs Fitted Values

From Dataset 2 we notice that although regularization reduces the amount the model will overfit on the data, we sacrifice RMSE in the process. Furthermore, specific regularizers may have a large impact on the dataset, as was demonstrated by the Elastic Net Regularizer. We must be careful to create a generalized model that accepts many unforseen test cases, while at the same time accurately represents the dataset. Regularization may help when the analyzer can strike a balance between the precision and accuracy of the model's output.

In general, the RMSE values are very high for the car insurance dataset 3, regardless of the feature pre-processing that is implemented or the models that are used. (However, it should be noted that our Deep Neural Network model did reduce the RMSE by 1,500, which could be attributed to Deep Neural Network's ability to use non-linear models.) There are several potential explanations for this. The high error could indicate that the insurance dataset does not contain the features that are the most important in determining users' car insurance charges. Alternatively, the dataset might provide the most important features, but there could be several other features that impact users' car insurance charges. It is also possible that that dataset might contain incorrect information: for example, smokers might not report that they smoke, which would corrupt the data. It should be noted that a whole profession, actuary science, exists to predict users' insurance charges.

For datasets that have very complex or poorly defined patterns, including insurance or economic data, simple regression models often perform poorly, so feature encoding offers little improvement. In these cases, using more elaborate models, such as neural networks, can provide better results.

Appendix

Scatter plots for Dataset 1, individual workflows, scalar encoding:

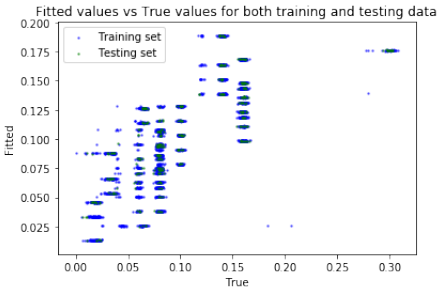

Fig. 60. Fitted vs. True, Workflow 0

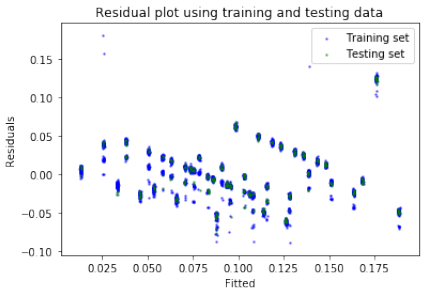

Fig. 61. Residuals vs. Fitted, Workflow 0

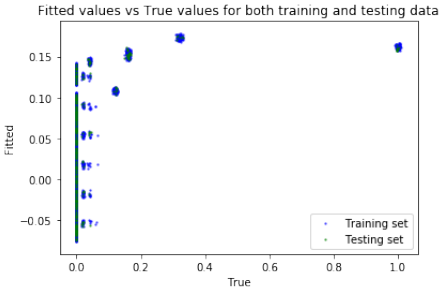

Fig. 62. Fitted vs. True, Workflow 1

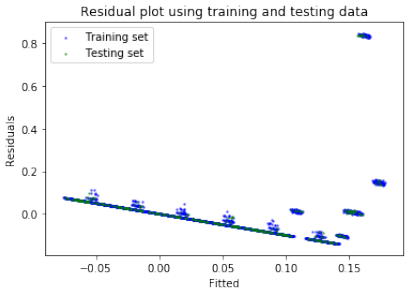

Fig. 63. Residuals vs. Fitted, Workflow 1

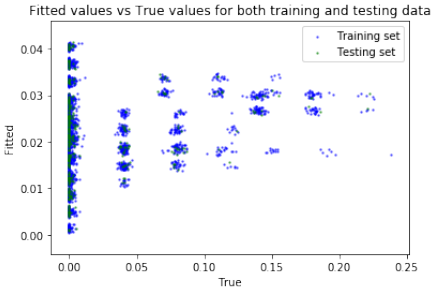

Fig. 64. Fitted vs. True, Workflow 2

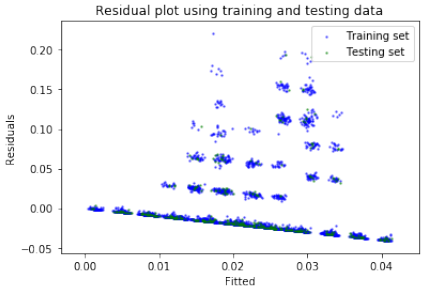

Fig. 65. Residuals vs. Fitted, Workflow 2

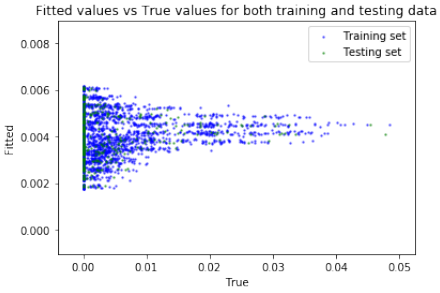

Fig. 66. Fitted vs. True, Workflow 3

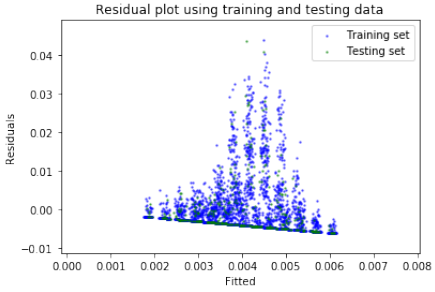

Fig. 67. Residuals vs. Fitted, Workflow 3

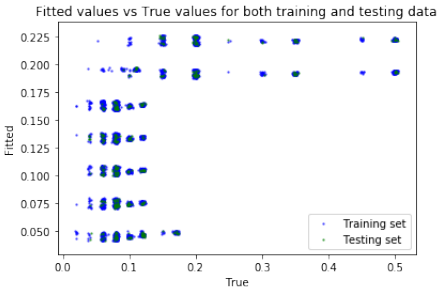

Fig. 68. Fitted vs. True, Workflow 4

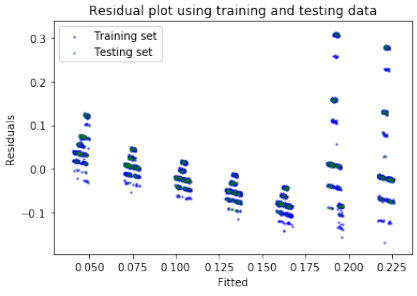

Fig. 69. Residuals vs. Fitted, Workflow 4

Scatter plots for Dataset 1, individual workflows, polynomial features:

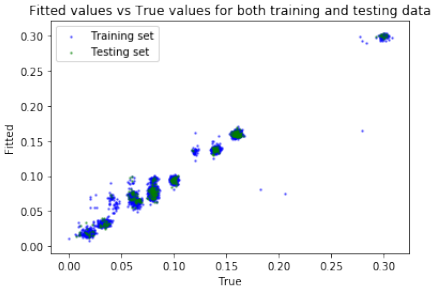

Fig. 70. Fitted vs. True, Workflow 0

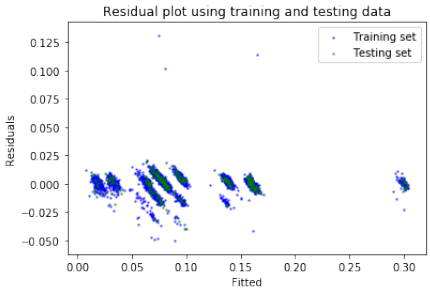

Fig. 71. Residuals vs. Fitted, Workflow 0

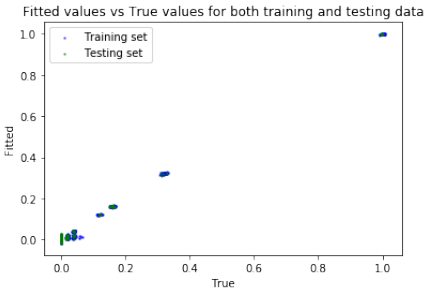

Fig. 72. Fitted vs. True, Workflow 1

Fig. 73. Residuals vs. Fitted, Workflow 1

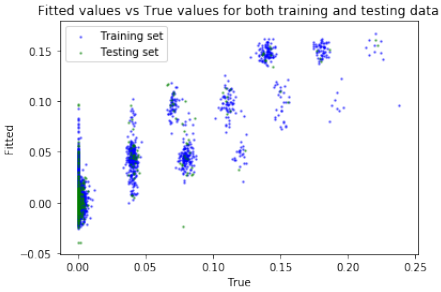

Fig. 74. Fitted vs. True, Workflow 2

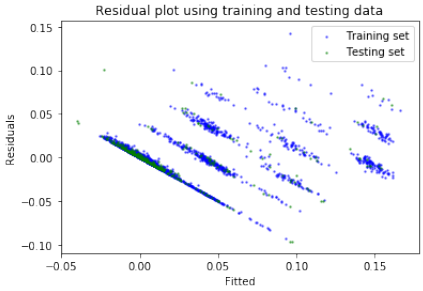

Fig. 75. Residuals vs. Fitted, Workflow 2

Fig. 76. Fitted vs. True, Workflow 3

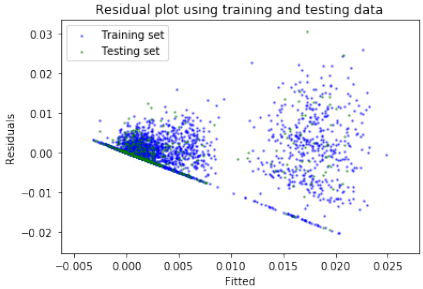

Fig. 77. Residuals vs. Fitted, Workflow 3

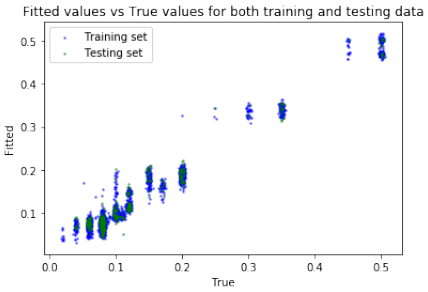

Fig. 78. Fitted vs. True, Workflow 4

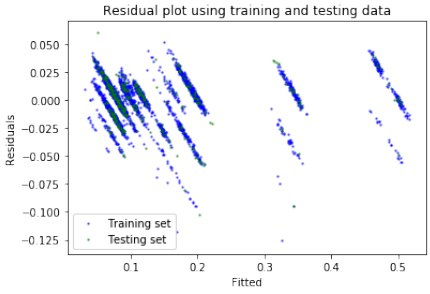

Fig. 79. Residuals vs. Fitted, Workflow 4

Bibliography

[Roy19] Roychowdhury, Vwani: Project 4: Regression Analysis, University of California, Los Angeles, 2019