Machine Learning Engineer Nanodegree

Capstone Project - Zillow House Price Predictions

Ann Chong

16 August 2017

1. Project Overview

The aim of this project is to build and compare the performance/predictive power of a linear model versus a neural network. In particular, this project aims to replicate the `hedonic pricing approach`[1] to house price predictions and to compare this to the predictive power of a neural network using the Zillow dataset.

This project is inspired by the NZARES[2] Paper and Kaggle's Zillow[3] (https://www.kaggle.com/c/zillow-prize-1) competition.

> [1]*The hedonic pricing approach is one which identifies price factors according to the premise that price is determined both by the internal characteristics of the good being sold and external factors affecting it.* Source: http://www.investopedia.com/terms/h/hedonicpricing.asp *It breaks down the item being valued into its constituent characteristics, and obtains estimates of the contributory value of each characteristic.*
>
> *The approach measures the relative importance – through use of regression analyses – of independent 'explanatory' variables on property prices. If, for example, through regression analyses increased distance from an open cast mining site is found to be correlated with increased house prices, it can be ascertained that the open cast site has a negative impact on house prices. The regression analysis can also be used to provide a value for the size of the relative impact. It may be found that a 1km movement away from the open cast site equates to an increase of £5,000 on a house price.* Source: http://www.cbabuilder.co.uk/Quant5.html
>
> *Under this approach, the change in a house price resulting from the marginal change in one of these characteristics is called the hedonic price (sometimes referred to as the implicit price or rent differential) and can be interpreted as the additional cost of purchasing a house that is marginally 'better' in terms of a particular characteristic.*

[2]http://ageconsearch.umn.edu/bitstream/97781/2/2004-9-house%20price%20prediction.pdf (http://ageconsearch.umn.edu/bitstream/97781/2/2004-9-house%20price%20prediction.pdf)

[3]https://www.kaggle.com/c/zillow-prize-1 (https://www.kaggle.com/c/zillow-prize-1)

1.1 Domain Background

Why estimate house prices?

Individuals, corporations and governments alike are interested in house prices.

For many individuals, buying a house constitutes the largest financial transaction they will ever make. This, together with the obvious financial gain that it is possible to achieve from accurately predicting house prices, makes house price predictions important to many stakeholders, including the following:

- prospective homeowners
- property developers
- investors
- government agencies
- other real estate market participants such as mortgage lenders and insurers.

Whilst individuals and corporations may be interested in house prices as a direct input into their investment decisions and financial planning, government agencies have an interest as the housing market is often thought to be an important indicator of social sentiment and how the economy of a country is doing.

Whether implicitly or explicitly, these stakeholders employ predictive models in their investment/decision making processes, albeit with varying degrees of sophistication, conscious realisation and conscious application. Nonetheless, it is difficult to imagine that any of these stakeholders would argue against more accurate predictive models.

The interest in accurate house price predictions can be seen in the number of `competitions` on the subject, including the following:

- Kaggle's Zillow competition - predicting the log error between Zillow's house price estimate and the actual sale price
- Kaggle's Sberbank competition - predicting the sale price of property in Russia

---

Common method for estimating house prices

Property prices are commonly estimated using the hedonic pricing method: the price of a property is determined by the characteristics of the property (size, appearance, features, condition) as well as the characteristics of the surrounding neighborhood (accessibility to schools and shopping, level of water and air pollution, value of other homes, etc.) The hedonic pricing model is used to estimate the extent to which each factor affects the price.[Source: http://www.investopedia.com/terms/h/hedonicpricing.asp#ixzz4kj51Z87i]

The price of a property is then the sum of the base price plus the contribution from each property characteristic.

Advantages of the hedonic pricing approach

The hedonic pricing approach uses statistically familiar multi-variate methods which makes it possible to test whether a proposed explanatory variable is statistically significant and to estimate the covariance between these variables. It also allows for the calculation of confidence intervals for the marginal contribution of each

explanatory variable to property prices.

This results in some intuitively appealing output such as people's marginal willingness to pay for a specific characteristic (e.g. extra bathroom) which can also be expressed as the changes in property values for a unit change in each characteristic.

An alternative to the hedonic pricing approach?

The literature to date suggests the hedonic pricing approach to be the most popular in the valuation of property prices.

The results from the NZARES Paper, however, suggests an alternative in the form of a neural network. More importantly, the paper suggests that hedonic pricing models do not outperform neural network models, whilst also acknowledging that other papers have reached a different conclusion and commented on the black box nature of neural networks.

It is the apparent ambiguity as to the performance of these two types of models on house price predictions that is the area of interest and further investigation of this project.

1.2 Approach

The variable we are trying to predict is the price of a house at a point in time. The potential solution to this is an algorithm that can make predictions about the sale price of a home at a point in time.

The proposed solution is the following:

1. build the best hedonic i.e. regression model to use to predict future house price.
2. build the best neural network model to predict future house price
3. compare the predictions from the two models on test dataset to determine which model type has better predictive capability on the test data

1.2.1 Evaluation Metrics

As previously proposed, in assessing the predictive quality of the two sets of algorithms, the R^2 metric and ***residual plots*** has been used. Additionally, I have also introduced the ***mse*** metric as a complement to the R^2 metric. The rationale for these metrics is further explained below.

The accuracy of a prediction is classified as: `absolute_errror=|predicted price - actual price|` i.e. the absolute difference between predicted and actual sale price. The smaller this number is, the more accurate the prediction is. A measure which relates to this is the mean squared error ('mse') i.e.

> mse=$\sum_{k=1}^{N} \left(prediction_k - actual_k\right)^2 / N$.

The intuitiveness and ease of interpretation of this measure also contributed to its choice.

The R^2 and **residual plots** were proposed for the following reasons:

> R^2 = explained variance/total variance. This is the proportion of the variation in the target variable that is explained by the linear model, and takes values between 0% and 100%. The closer the R^2 value is to 100%, the better the model explains the variability of the target variable around its mean.
>
> Mathematically, R^2 is the square of the correlation coefficient, ***r***, and can be calculated as follows:
>
> R^2=r^2 where r=[n($\sum(xy)$-($\sum x$)($\sum y$)]/$\sqrt{[n(\sum(x^2) - (\sum x)^2][n(\sum(y^2) - (\sum y)^2]}$
>
> This measure has an advantage over ***mse*** in being scale invariant i.e. its interpretation does not change with the scale of inputs and its interpretation will not change from model to model.
>
> Residual plots are used as the R^2 variable on its own is not able to determine if a model is biased.Ref:http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

1.2.2.Benchmark Model

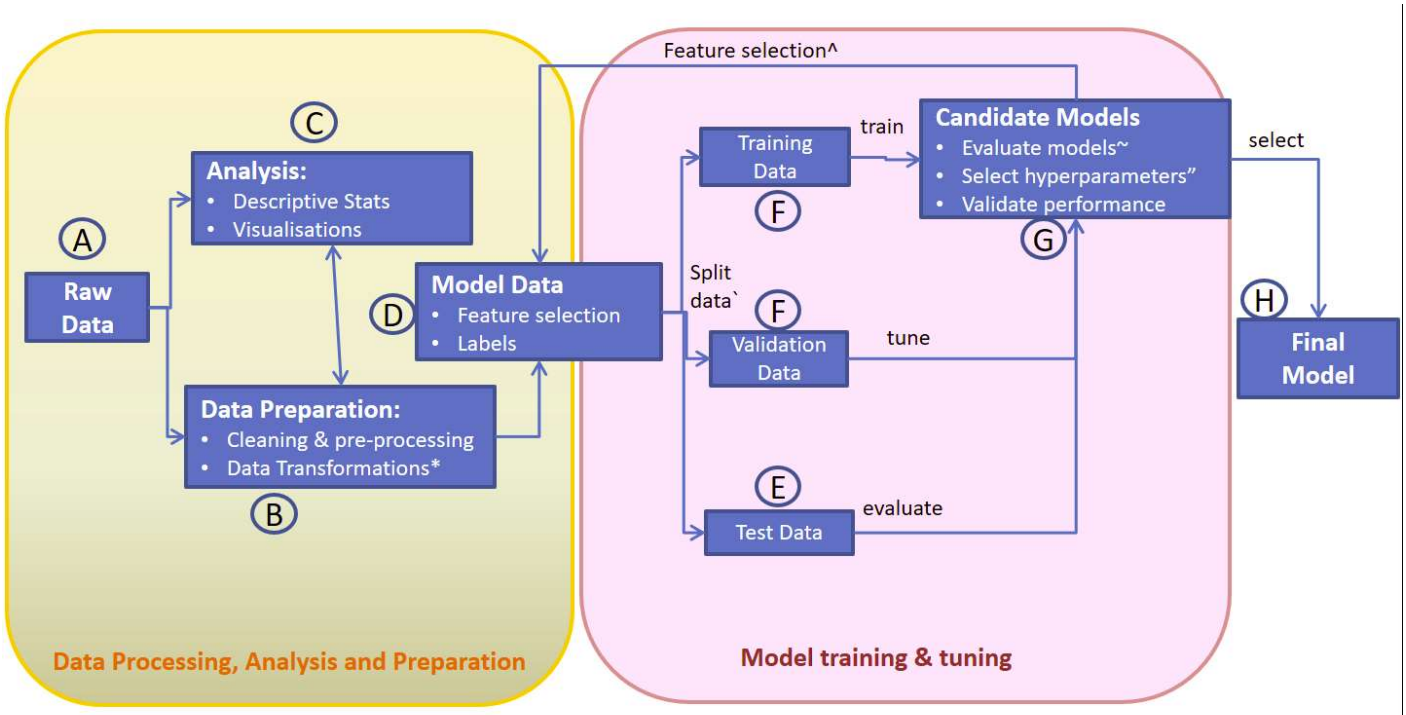
A benchmark model to use as reference is one which always predicts the average value in the dataset. In this case, given the dataset's split into training and testing datasets, this would be a model that always predicts the average logerror of the training dataset.

The following are the metrics for the benchmark model:

Item	Benchmark
R^2	-0.00003415
mse	0.02463887

2. Project Workflow

The previously proposed project workflow is replicated below:



*:transformations to better align the structure of prediction problems to modelling algorithms as different algorithms make different assumptions about data.
`:data can be split a variety of ways, including k-fold cross validation and repeated random train-test split
~:evaluation metrics include mean absolute error, mean squared error and R-squared.
^:features can be selected using Recursive Feature Elimination or PCA (a data reduction technique that can aid in feature selection)
":hyperparameters may be selected using grid search approaches or random search approach

The following is a discussion of the different parts of the process.

2.A Raw Data

The source of the data for this project is the dataset from Kaggle's Zillow competition. This is comprised of the following files:

File	Description
train_2016.csv	A list of 2016 property transactions, predominantly pre 15 October, 2016. This file comprised of 90811 rows and 3 columns. The 3 columns are made up of the ids of properties transacted in 2016, the transaction date and the logerror of each transaction.
properties_2016.csv	A database of characteristics for all 2016 listed properties in 3 counties in California: Los Angeles, Orange and Ventura. This file is comprised of 2,985,217 rows and 58 columns.
zillow_data_dictionary.xls	This file contains a description of the variables in the properties file and was a big input into **part B** of the process.

2.B Data Preparation

The aim of this part of the process is to ensure the data provided is fit for purpose, and encompasses both of the following two steps: ***Data Preprocessing*** and ***Data Transformation***

2.B.1 Data Preprocessing

Ultimately, this step ensures that the data is in the format and form required by the learning algorithms. For example, the regression models and neural network models require that their inputs be numeric. This has implications for variables which are categorical or variables which have numeric variables but where the variables do not have a numeric relationship with each other. This is further explained in item 5 below.

Additionally, the following other preprocessing steps were taken:

1. replacing the ***`transactiondate`*** variable into 2 corresponding variables: ***transaction_day*** and ***transaction_mth***.
2. renaming variables so they are more intuitve and understandable.
3. identifying and removing variables which are not expected to have any information content. An example of this is the identifier variable ***parcelid***.
4. restating variables so that they would have a more straightforward interpretations. Examples of this are:
    - restating the variable ***year_built*** into ***bldg_age*** i.e. building age as at 2016.
    - restating the variable ***tax_delinqncy_year*** to ***tax_delinqncy_age*** as at 2016.
5. Identifying categorical and non-categorical variables. Variables were classified as either categorical or non-categorical with reference to the zillow_data_dictionary, general knowledge and personal interpretation of the variables. This is expanded further below. I ackowledge that better subject matter experts may have made different choices in this regard.

2.B.1.1 Categorical vs Non Categorical Variables

Ultimately, since the linear models and neural networks in Python require numeric variables as inputs, categorical variables had to be transformed into a numeric equivalent. This was done via ***one hot encoding***.

The original dataset had 56 useful features (i.e.after removing identifier type variables such as parcelid). Of these, 22 were identified as categorical variables and 34 were considered numeric variables. Categorical variables were defined as either:

- features which do not have a pre-defined numeric relationship with each other such as architecture_style and air_cond_type. This involved personal judgment and may have an element of subjectivity; or
- features which only had 1 unique values or missing values. Whilst these may be legitimate numeric variables, these were classified as categorical so that they can be one-hot-encoded as the alternative treatment of imputing values of the empty cells would lead to loss of any useful information. Additional discussion on dealing with missing values is provided below.

The remaining non-categorical variables were sanity checked by looking at their data types and ensuring only numeric data types (e.g. floats or ints) were returned.

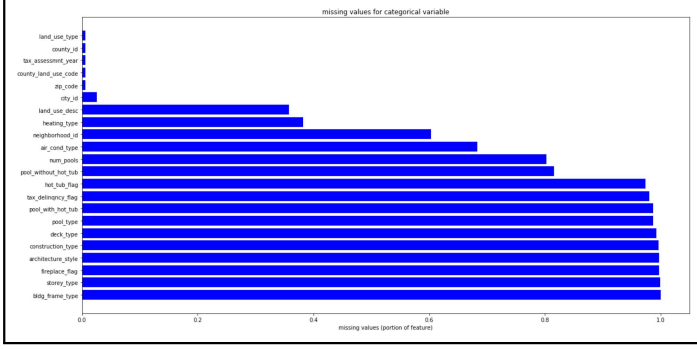
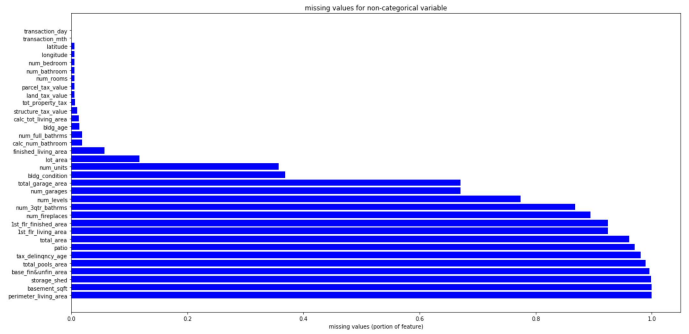
2.B.1.2 Missing Values

The following are the methods used to deal with missing values:

non categorical variables: missing values were imputed using the mean value from the feature set. Other methods which could have been used: imputing the features with the median or mode of the feature set.

categorical variables: missing values were first demarcated by an 'NA' value. These variables were then one-hot encoded, with the features containing missing values receiving its own 'feature_NA' column.

2.D.1.1 Missing Values

Missing Values - Categorical Variables	Missing Values - Non Categorical Variables
	

Observations:

- A large number of property characteristics have a significant proportion of NULL values.
- A few characteristics are mostly empty.

Given this, a new variable, ***non_sparse_features*** was created which identifies the features with less than 80% missing values, as a precursor to creating an alternative feature space for the analysis. This is used by the PCA technique in the Feature Selection stage of the process.

2.B.2 Data Transformations

As mentioned above, given that different machine learning algorithms make different assumptions above data, data transformations are sometimes required to better align the prediction problem to the proposed solution. In this project, the following transformations were made to the input data:

- **standardisation**: this involves rescaling each feature in the dataset such that is has a mean of zero and a standard deviation of 1, and is generally suitable for learning algorithms that assume input variables have a Gaussian distribution, including linear regression.
- **normalisation**: This is feature scaling to standardize the range of features such that each feature contributes approximately equally to the outcome. This is especially useful for sparse datasets or those with attributes of varying scales.

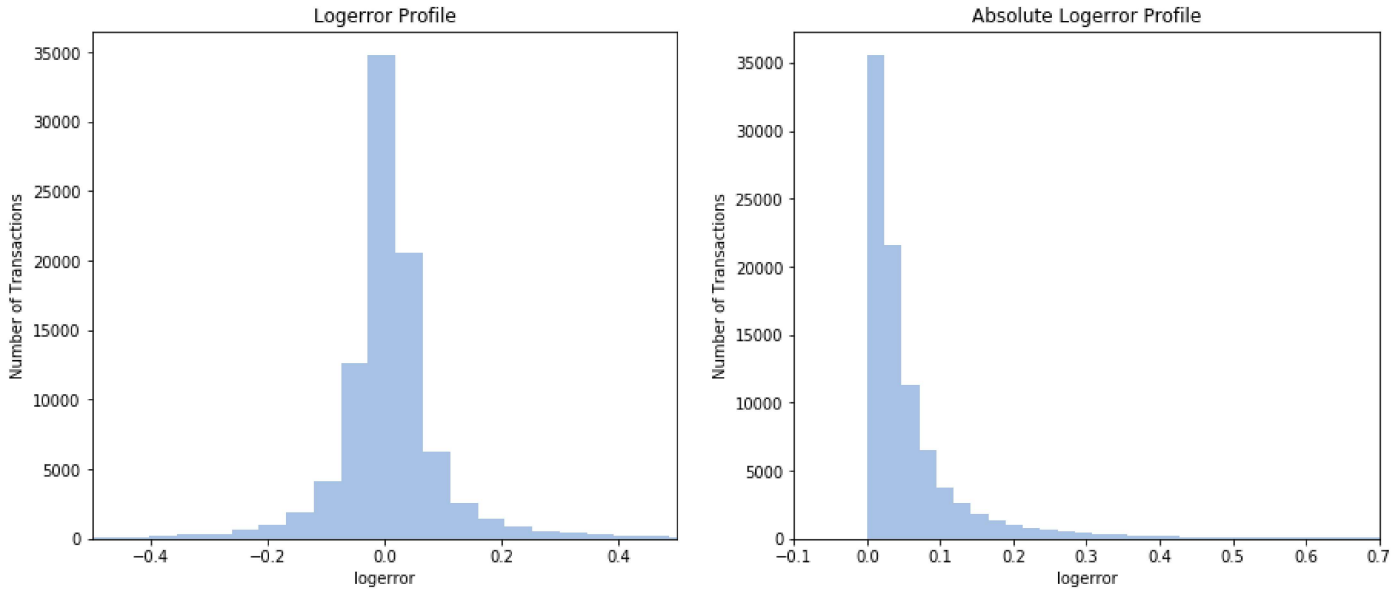
Since there are no generally agreed rules on data transformations and no any guarantees on a betterment in performance from performing data transformations, I have also maintained a dataset with no standardisation nor normalisation performed on it.

2.C Analysis

This stage of the process is important in providing greater understanding and context for the problem at hand and to highlight any instances which require further investigation or action. Ultimately, the hope is that this stage provides more insight and intuition into the problem at hand and the tools (i.e. data inputs) available to address it.

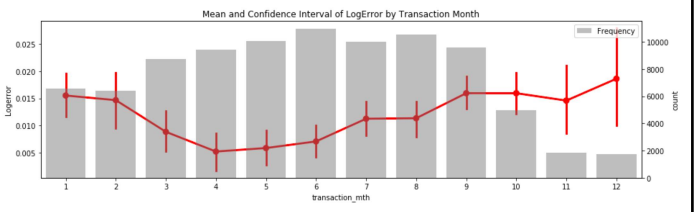
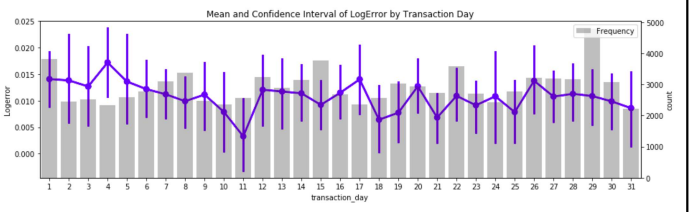
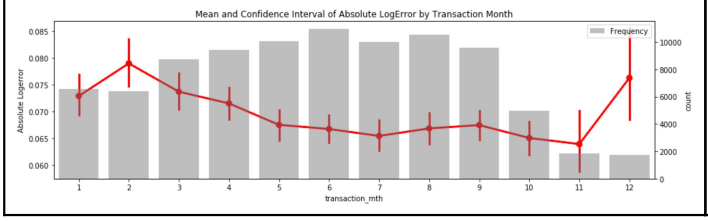
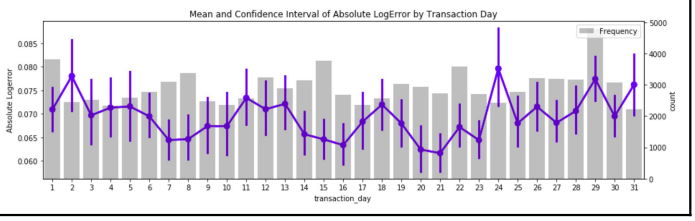
Some interesting observations which have resulted from this process include the following:

2.C.1 Logerror Profiles



Comment: logerror profile seems like it could be normally distributed and does not look skewed.

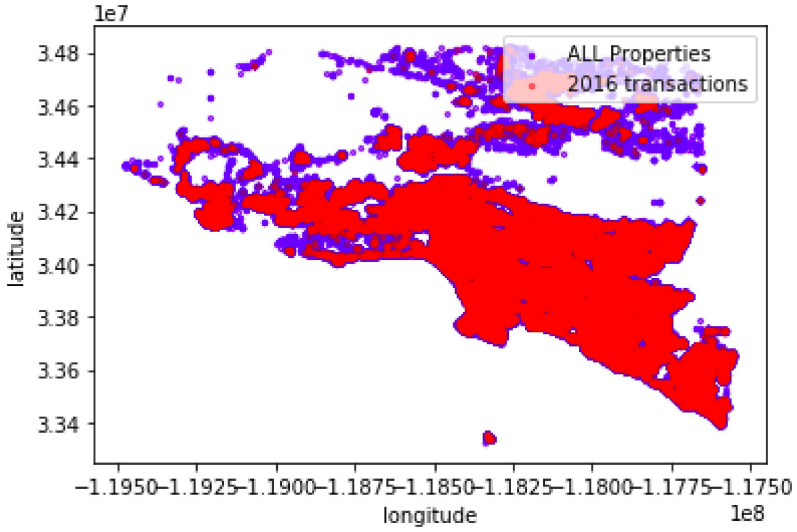
2.C.2 Logerror and <u>Absolute</u> Logerror Distributions by Transaction Day and Month

Logerror vs Absolute Logerror By Transaction Month	Logerror vs Absolute Logerror By Transaction Day
	
	

Comment:

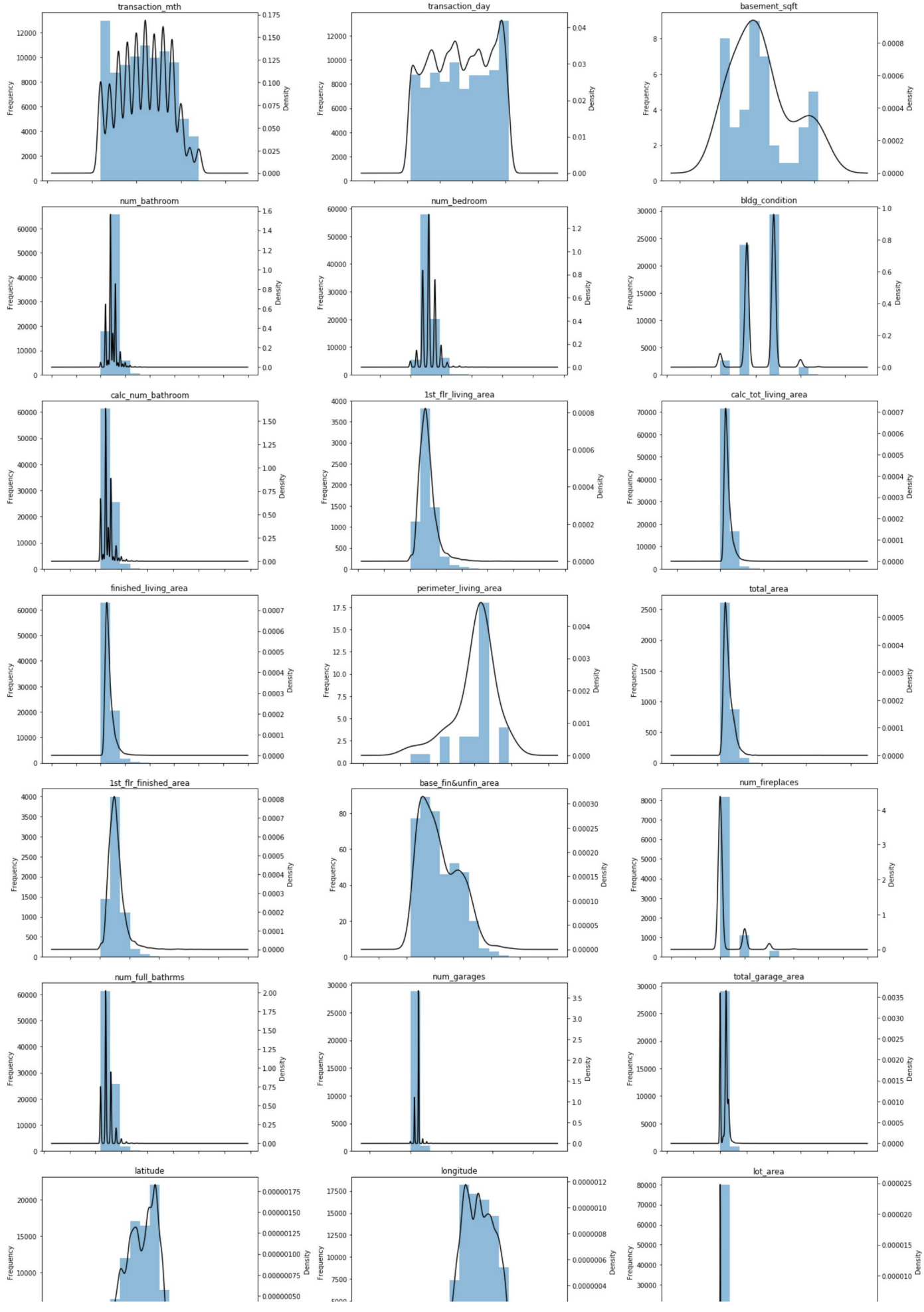
- Only some of the transactions after 25 Oct 2016 have been included in the training set. This probably accounts for the abnormally low numbers for months 10, 11 and 12.
- Visually, there appears to be an approximately inverse relationship between logerror and exposure (i.e. number of transactions) when viewed on a transaction month basis. This relationship is less clear when viewed on a transaction day basis. ***Transaction month could be a useful predictor.***
- The approximately inverse relationship also appears to hold true between ***absolute logerror*** and exposure (i.e. number of transactions)
- The logerror profile seems to show some variation between using the logerror vs abs_logerror, although the previous observation about the inverse relationships between the logerrors and number of transactions seems to be generally true. This project focuses on predicting the logerror, although an argument might be made for predicting the absolute logerror (abs_logerror) instead.

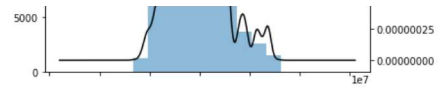
2.C.2 Geographic View of Properties

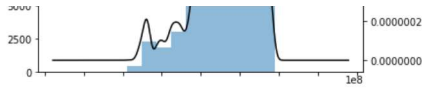


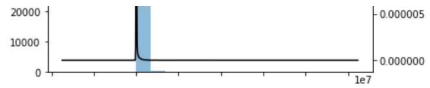
2.C.3 Histogram and density plots of Numeric Variables

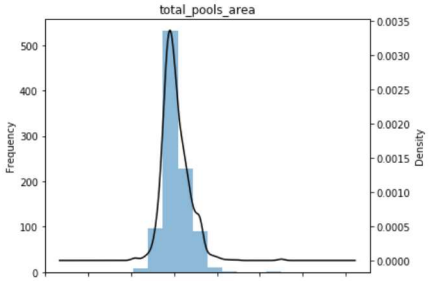
Visually, some of the non-categorical variables appear skewed, non continuous or non normally distributed. This makes it especially important that data transformations and feature selection takes place.

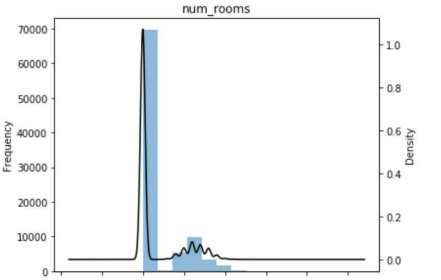

transaction_mth
transaction_day
basement_sqft
num_bathroom
num_bedroom
bldg_condition
calc_num_bathroom
1st_flr_living_area
calc_tot_living_area
finished_living_area
perimeter_living_area
total_area
1st_flr_finished_area
base_fin&unfin_area
num_fireplaces
num_full_bathrms
num_garages
total_garage_area
latitude
longitude
lot_area
Frequency
Density

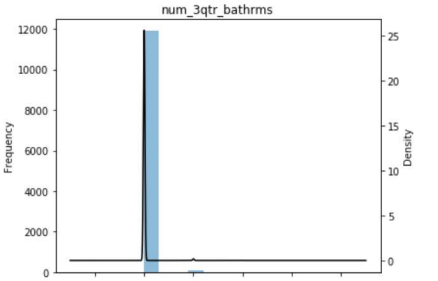


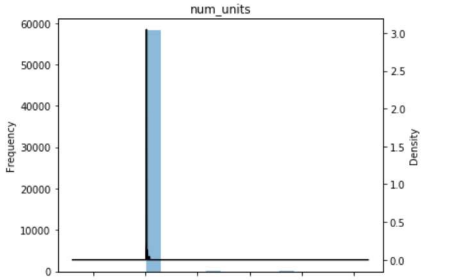


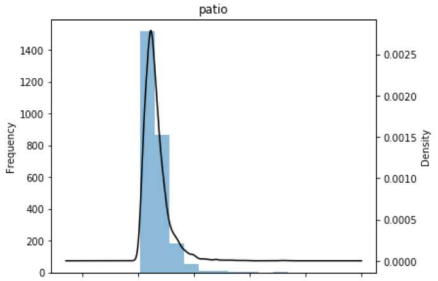


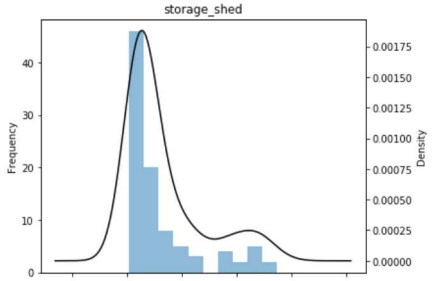

total_pools_area
Frequency
Density

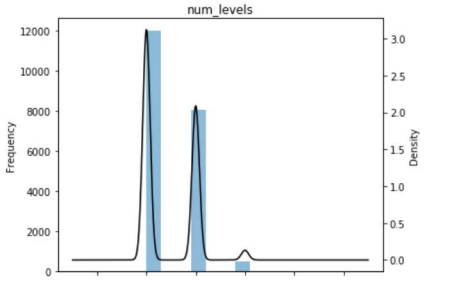

num_rooms
Frequency
Density

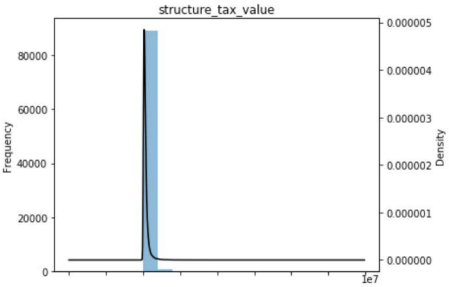

num_3qtr_bathrms
Frequency
Density

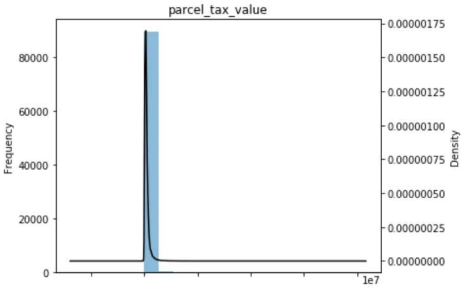

num_units
Frequency
Density

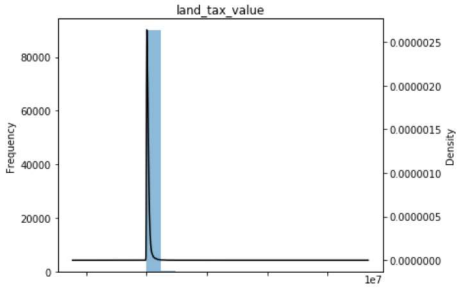

patio
Frequency
Density

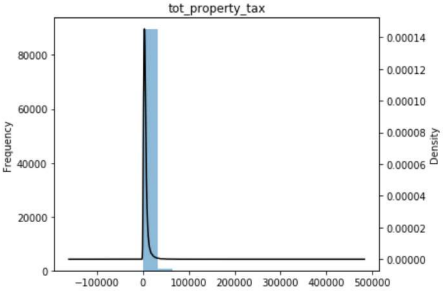

storage_shed
Frequency
Density

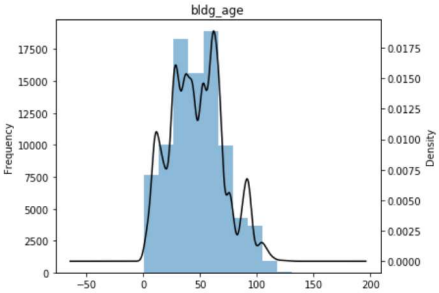

num_levels
Frequency
Density

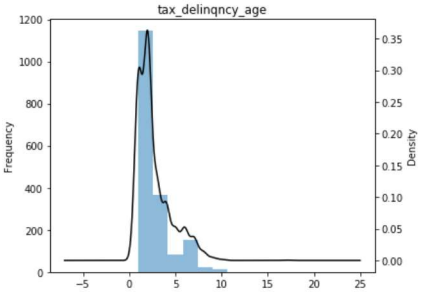

structure_tax_value
Frequency
Density


parcel_tax_value
Frequency
Density

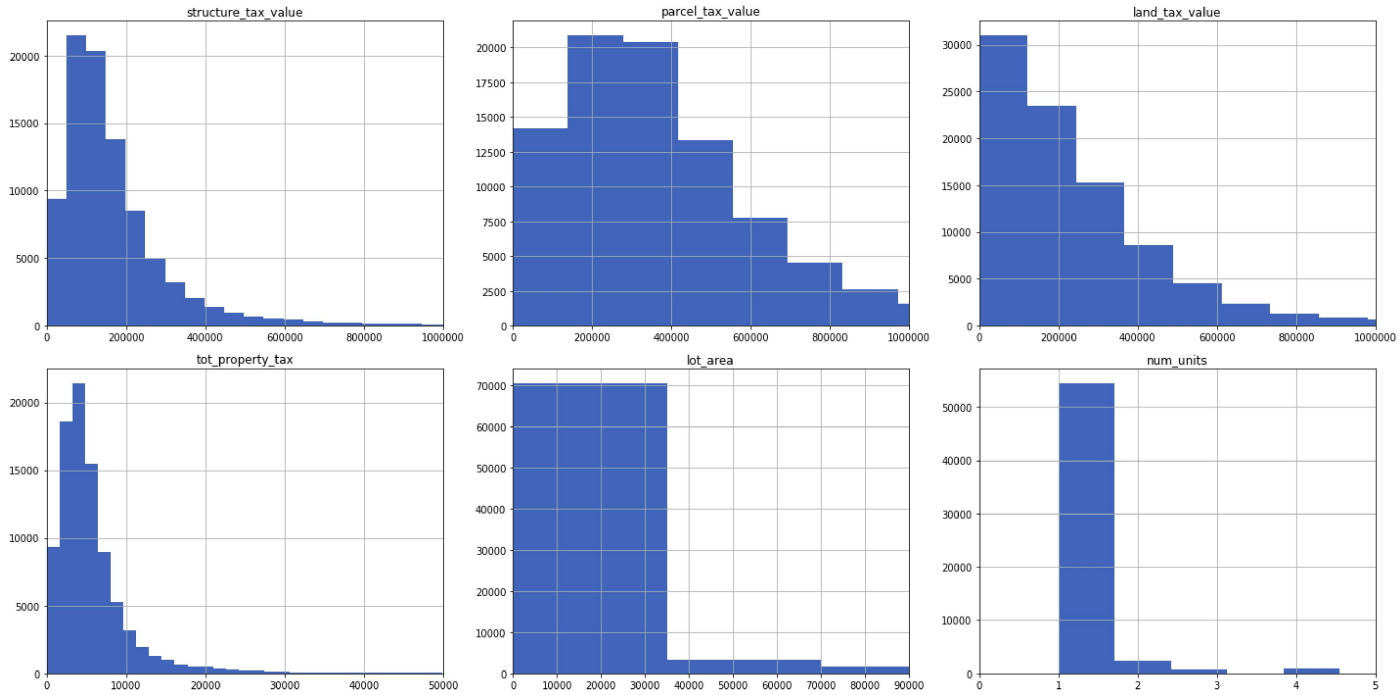

land_tax_value
Frequency
Density


tot_property_tax
Frequency
Density


bldg_age
Frequency
Density


tax_delinqncy_age
Frequency
Density

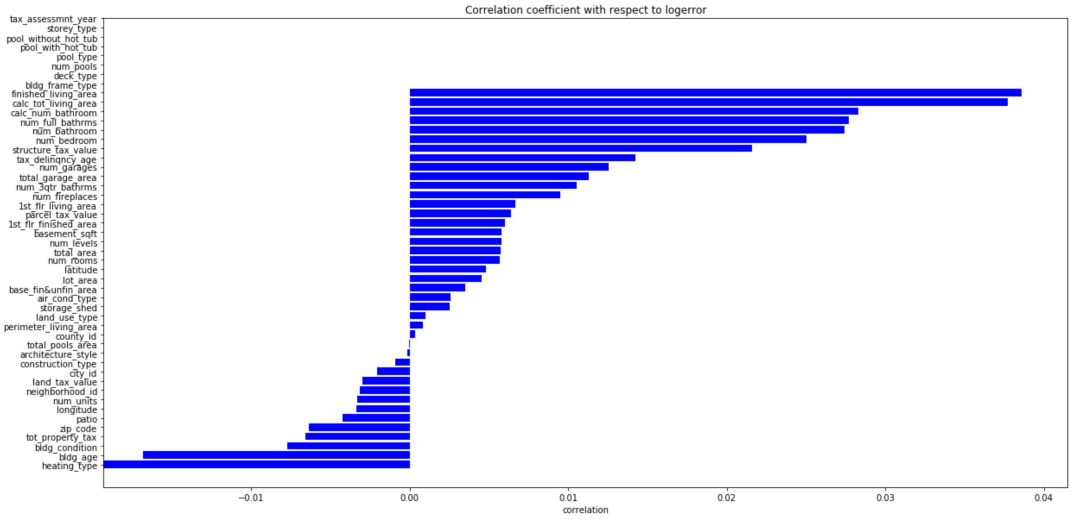
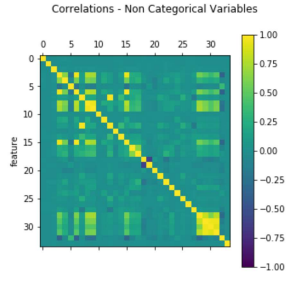
Zooming in on some variables:



2.C.4 Correlation Profiles

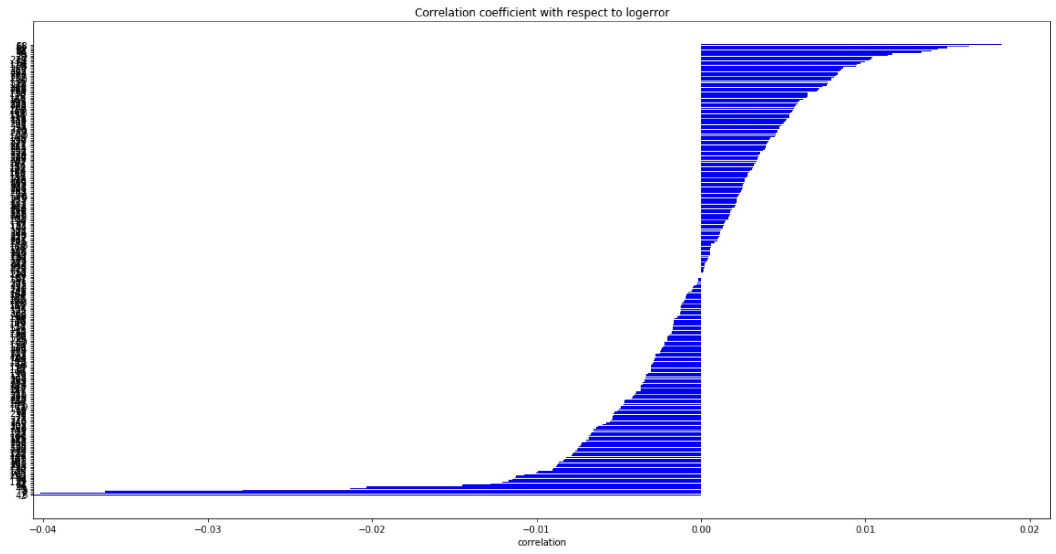
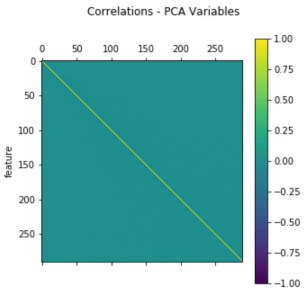
2.C.4.1 Input Features (no data transformations)

The inter feature correlation plots definitely reveal some strong correlations between certain features.

Logerror Correlations	Inter Feature Correlations
	

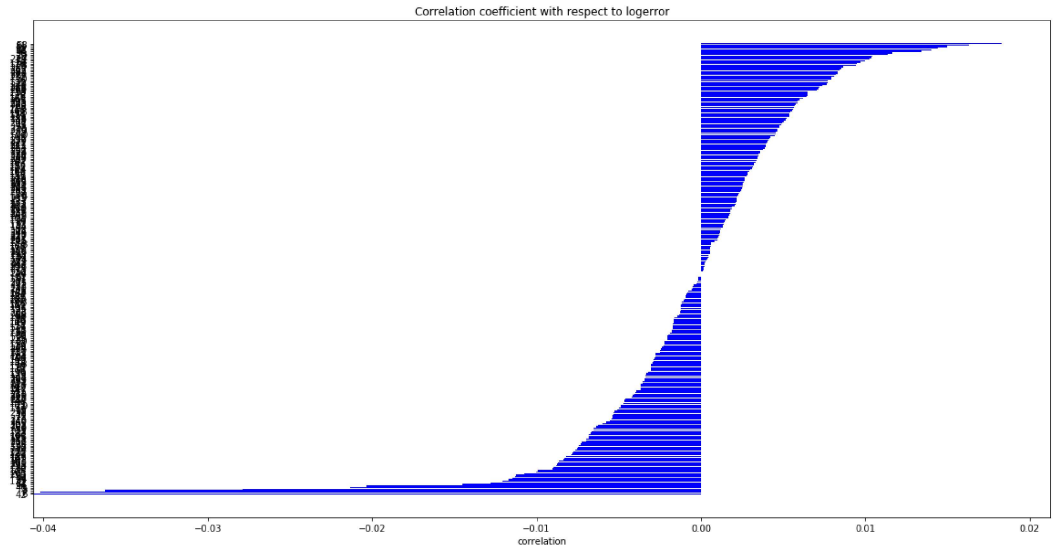
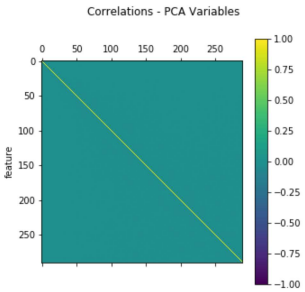
2.C.4.2 PCA transformed features, No Standardisation, No Normalization

The inter feature correlations look ideal after performing the PCA transformation of the feature space.

Logerror Correlations	Inter Feature Correlations
	

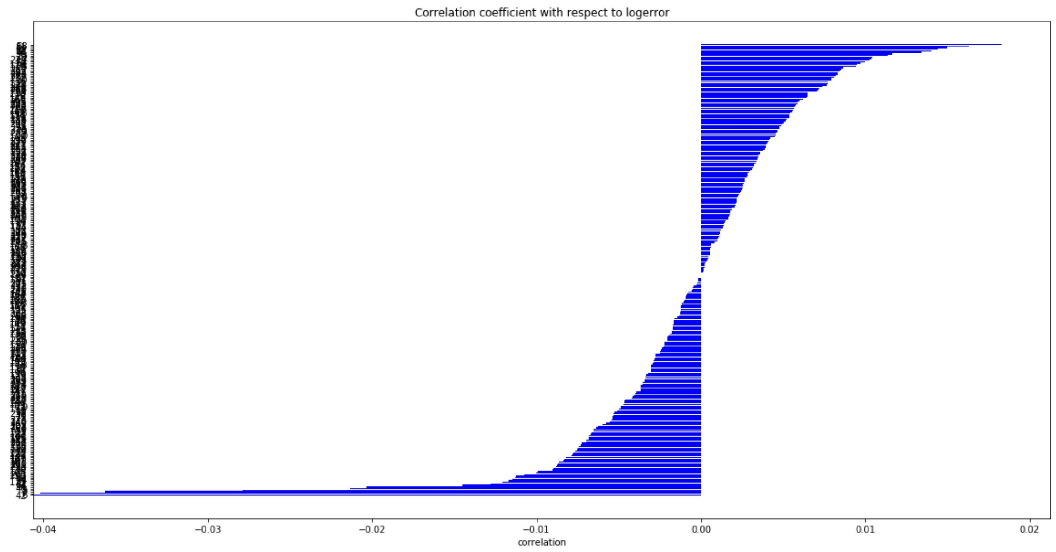
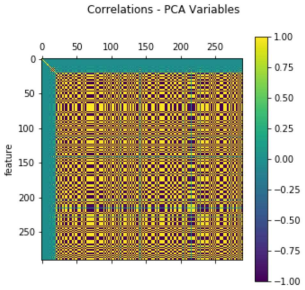
2.C.4.3 PCA transformed features, With Standardisation

Standardising a PCA-transformed feature set does not seem to have had any impact on its correlation structures.

Logerror Correlations	Inter Feature Correlations
	

2.C.4.2 PCA transformed features with Normalisation

The results suggest normalising a PCA-transformed feature set has the impact of correlating the variables. It suggests a model based on PCA transposed ***then*** normalised features might perform poorly.

Logerror Correlations	**Inter Feature Correlations**
	

2.D Model Data

2.D.1 Feature Selection

The features that go into a learning model can greatly influence its performance. Hence, there is an inherent tension in including features that are irrelevant and excluding features with relevant information content, both of which can degrade the performance of the algorithm.

Whilst 56 features were provided in the data, the one hot encoding of categorical variables transformed the feature space into 3243 variables. Given the sparseness of the original feature set and the fact that some variables had large numbers of unique values (an example being ***land_use_desc*** which had 1997 unique values), there was every possibility of irrelevant features in the new feature space.

To manage the size of the feature space and to select only the most relevant features, I considered the following two approaches: **Feature Importance** and **PCA**.

2.D.1.1 Feature Importance

Two tree based models were fitted to the training model to provide an indication of the importance of each feature. The two models were an **xgboost** model and a **random forest** model.

Generally, an importance score provides an indication of how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The performance measure may be the purity (Gini index) used to select the split points or another more specific error function.

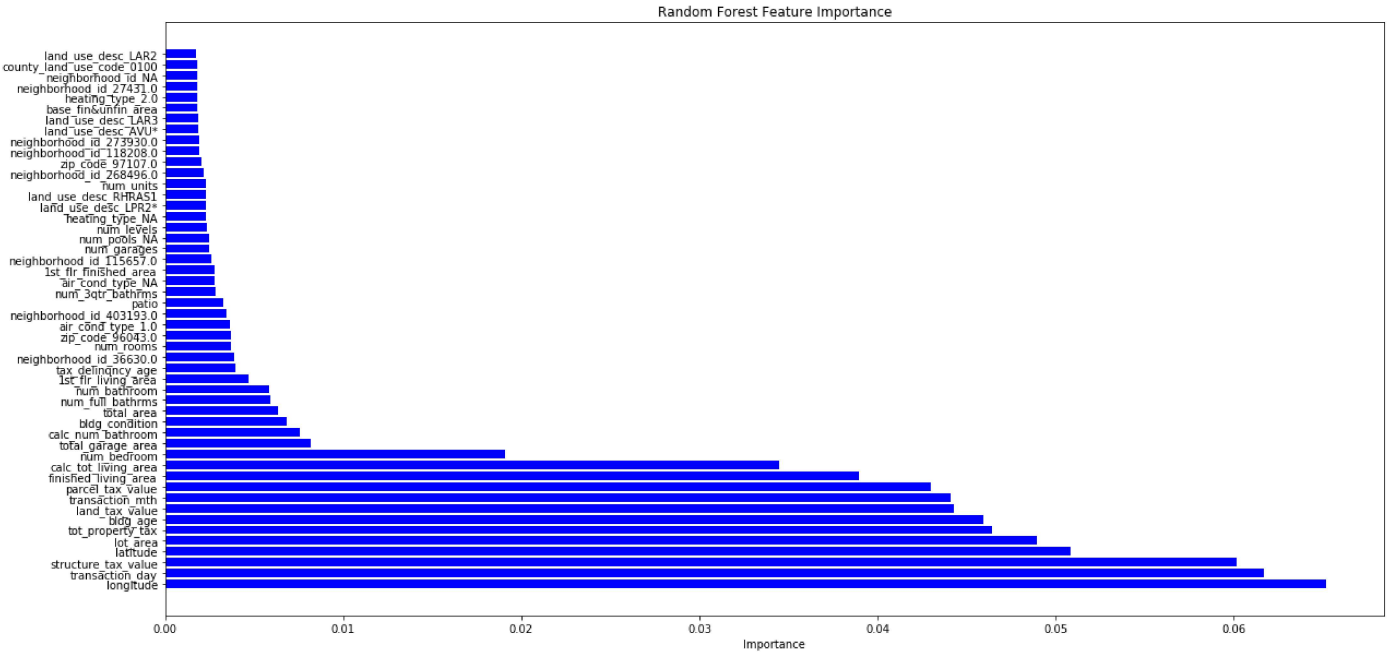
The feature importances are then averaged across all of the the decision trees within the model.

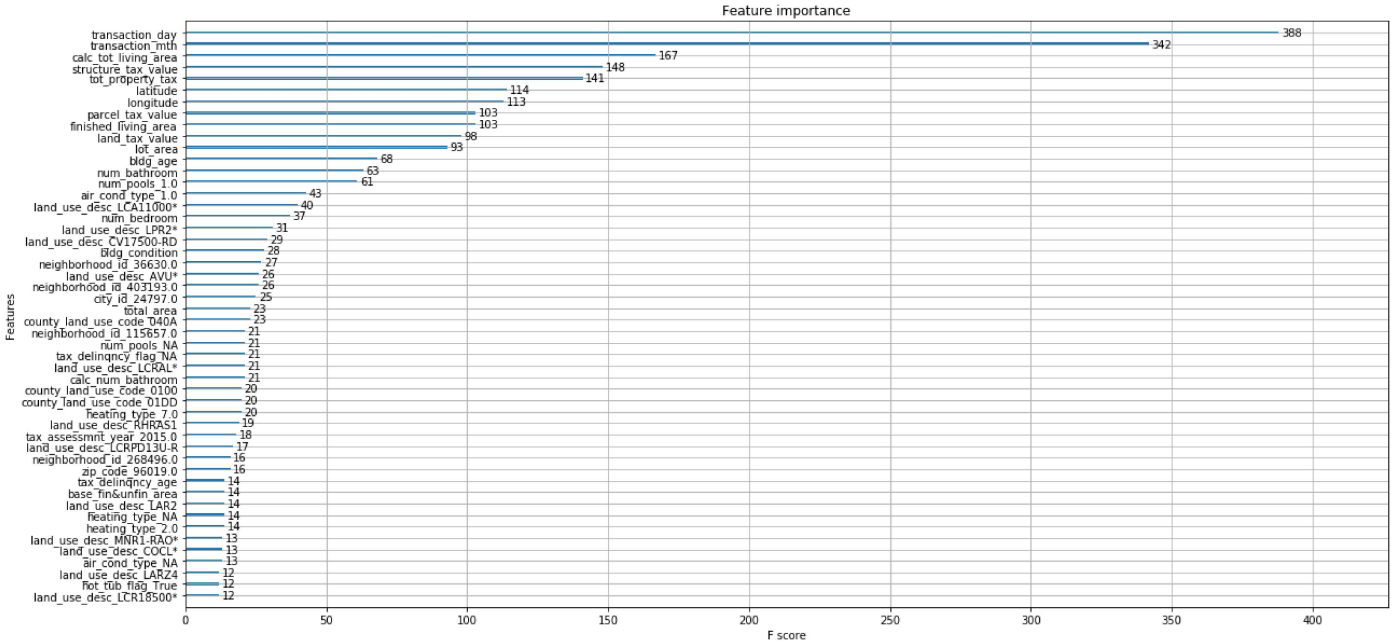
The xgboost model was set with the following parameters:

eta	max_depth	subsample	colsample_bytree	objective	eval_metric	silent
0.05	8	0.7	0.7	reg:linear	'rmse'	1

The RandomForestRegressor model was set with sklearn's default parameters.

The top 50 results from this process is illustrated below:





Interestingly, there is overlap in the importance of features between the two models. For example, transaction_day, structure_tax_value and total_property_tax are both within the top 10 features of importance according to these models.

However, the two models had the following performance on test data:

Models Result - XGBoost	**Models Result - Random Forest**
Number of features in training data: 3243 Results from Booster: r2 score: 0.00898876 MSE (model vs baseline): 0.02292748 vs 0.02313985 Number of features in training data: 3243	Results from RandomForestRegressor: r2 score: -0.15417506 MSE (model vs baseline): 0.02670235 vs 0.02313985

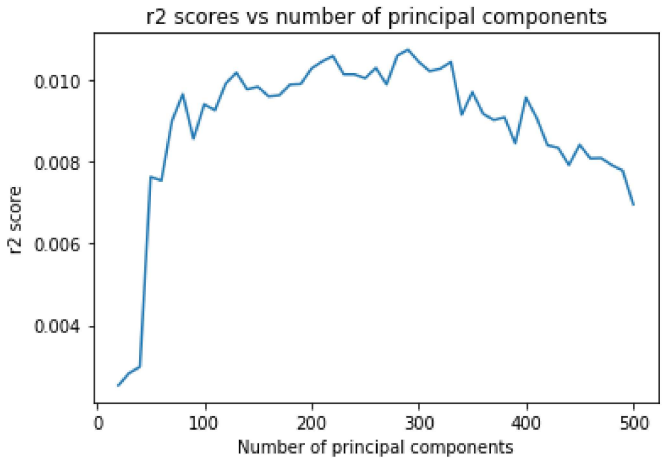
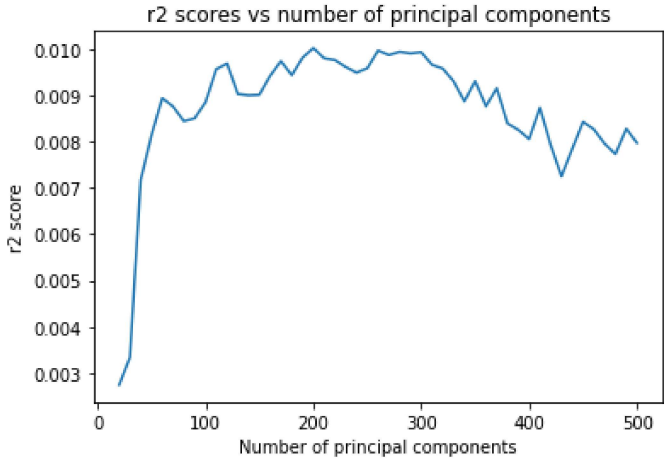
As the random forest had poor results as evidenced by the negative R^2 score indicating it is not predictive at all, I have decided to only use the results from the XGBoost model in determining the model's importance factors.

2.D.1.2 PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that seeks to compress the dataset into a user specified number of dimensions/features.

Given the trade off between number of dimensions and loss of information, I analysed the performance of a Linear Regression model on different feature spaces/dimensions using both X_all (i.e. the entire original feature space) and X_non_sparse (the feature space with sparse variables i.e. less than 80% NULLS removed) using the PCA approach.

The results are shown in the following graphic:

Linear Regression R2 scores - all data	**Linear Regression R2 scores - non sparse data (<80% NULL)**
	
Optimal number of dimensions by r2 score: 290	Optimal number of dimensions by r2 score: 200

As the r2 scores are marginally higher using 290 principal components on X_all compared to X_non_sparse, I have selected the best feature space according to the PCA method to be 290 components on X_all.

2.D.1.3 PCA vs Feature Importance

At this point in the feature selection process, I have the following two data subsets:

1. X_all_pca: PCA transformed data with 290 components/features
2. X_select: a subset of the original dataset, X_all, containing only the 350 features selected as important by the xgboost algorithm.

To decide on which dataset seems to have the most information content, I then a linear regression model using these two data sets.

The results of these analyses is summarized below:

Linear Results Regression - X_all_pca	**Linear Regression Results - X_select**

`Results from LinearRegression:`
`r2 score: 0.01016620`
`MSE (predicted vs baseline): 0.02290024 vs 0.02313985`
	`Results from LinearRegression:`
	`r2 score: 0.00465941`
	`MSE (predicted vs baseline): 0.02302765 vs 0.02313985`

As the results from using the PCA transformed data was better than that using the subset of data as suggested by the xgboost importance factors, I have applied the PCA transformed data for the rest of the process.

2.E Test Data

The test data was split using the **train_test_split** function from the **sklearn.cross_validation** module. 20% of the original dataset was split into the test dataset.

3.F Training and Validation Data

The training data was split using the **train_test_split** function from the **sklearn.cross_validation** module. 80% of the original dataset was split into the test dataset.

Within each training data set, validation data was implicitly generated using a KFold cross validation method which was imposed using the **cross_val_score** function from the **sklearn.model_selection** module.

3.G Candidate Models

3.G.1 Linear Models

The following table sets out the linear models considered:

Regression Model	Comment	Regularization Term	Advantages	Disadvantages
Linear	Assumes input variables are independent of each other and have a linear relationship to the output variable. Minimises the sum of squared error function. Run using the **LinearRegression()** function from **sklearn.linear_model** .	-	Simple to understand. Works well when the features are independent and the relationship between them and the target is linear are	Susceptible to overfitting. Also assumes that the dependent variables are continuous and that the errors are independent and identically distributed.
Ridge	Modification to Linear Regression by imposing a penalty term that is the $L2^*$ norm. Run using the **Ridge()** function from **sklearn.linear_model**.	alpha	Useful when data suffers from multicollinearity (ie independent variables are highly correlated). Improves prediction accuracy by reducing large regression coefficients in order to reduce overfitting.	Cannot zero out coefficients.
Lasso	Modification to Linear Regression by imposing a penalty term that is the $L1^\wedge$ norm. Hence, it is easier for coefficients to be zero. Run using the Lasso() function from **sklearn.linear_model**	alpha	Performs parameter shrinkage and variable selection automatically. Preferred over ridge when the solution is believed to have sparse features.	If two predictors are highly correlated, lasso can end up dropping one arbitrarily i.e. inconsistent feature selection and possible loss of independent variables along the way.

Regression Model	Comment	Regularization Term	Advantages	Disadvantages
Elastic Net	Combines properties of Ridge and Lasso Regressions. It imposes both the L1 and L2 norm penalty terms on a regular linear regression model. Run using the **ElasticNet()** function from **sklearn.linear_model**	alpha, l1_ratio	Preferred over lasso in situations where the number of features is greater than the number of samples and with correlated features, where the lasso behaves erratically.	

^L1 norm: $\sum_{k=1}^{N} |coeff_k|$

*L2 norm: $\sum_{k=1}^{N} coeff_k{}^2$

3.G.2 Neural Network

The neural network model considered in this project is a multi layer perceptron model which was implemented using the **Keras** wrapper library in Python.

The network topology is comprised of the following parameters:

Layer	Item	Comment
Input	Number of Inputs	number of features in the dataset
Input	Number of neurons	number of features in the dataset
Input	Kernel Initialization	small random numbers sampled from a normal distribution
Input	Activation function	rectifier activation function
Hidden	Number of neurons	number of features in the dataset
Hidden	Activation function	rectifier activation function
Hidden	Kerner initialization	small random numbers sampled from a normal distribution
Output	Number of neurons	1

The only change in the network topography that is allowed in the search for an ***optimal*** model is in the number of hidden layers, with each hidden layer having the same characteristics as seen above.

The following is the code used to create the network topology:

```
def NN_baseModel(num_inputs,num_hidden_layers=1):
    numpy.random.seed(123)

    model=Sequential()
    model.add(Dense(num_inputs,input_dim=num_inputs,kernel_initializer='normal',activation='relu'))

    hidden_layer_count=0

    while hidden_layer_count<num_hidden_layers:
        model.add(Dense(num_inputs,init='normal',activation='relu'))
        hidden_layer_count+=1

    model.add(Dense(1,init='normal'))

    model.compile(loss='mean_squared_error',optimizer='adam',metrics=[])

    return model
```

Once the network structure is specified, additional compiler parameters need to be specified. This provides more instruction on how to train the model. The following were the compiler variables used::

Compiler Inputs	Value	Comment
Loss Function	mean_squared_loss*	the loss function is used to evaluate the performance of a set of network weights
Optimizer	adam	this is the gradient descent algorithm. this is how the algorithm will search through the different weights for the network

Compiler Inputs	Value	Comment
metrics	-	as mentioned below, due to software errors and an inability to resolve them, no specific metrics were specified.

The following default training parameters for the neural networks were used:

- batch size: 100
- number of epochs: 5

* whilst the mean_squared_loss is a legitimate variable for assessing prediction accuracy, the preference at the time of modelling was for the r2_score to be returned instead. However, I was unable to resolve an error with Keras' interface to the underlying backend which meant I could not specify other metrics to use in the calculations.

3.G.3 Model Selection Process

The model selection process is performed over the following dimensions:

- dataset:
  - regular data (i.e. non-standardized, non normalized)
  - standardized data
  - normalized data
- model type
  - linear:
    - Linear Regression
    - Ridge
    - Lasso
    - Elastic Net
  - neural network. The neural network type is a multilayer perceptron with variations only in the number of layers of each model.

The model selection process is comprised of the following steps:

1. choose best linear model for each dataset. See section 3.H.1 for results from this process.
   - observe results from baseline linear models. Additional details on baseline models are provided in section 3.G.3.1 below.
   - vary the regularization parameters and observe the results from each resulting model. Additional details on this step of the process is provided in section 3.G.3.2 below.
   - select the model with the best performance. The result from this step is summarized in 3.G.3.3 below.
2. choose best neural network model for each dataset. See section 3.H.2 for results from this process.
   - This is performed by observing the model results from varying the number of hidden layers in the multilayer perceptron model. The number of hidden layers range from 1 to 5.
3. choose best model from the output of steps 1. and 2. See section 3.H.3 for results from this process

3.G.3.1 Baseline Linear Models

For each of the linear model types, a baseline model is determined. The baseline models are set using the following default parameters:

- alpha=1.0
- l1_ratio=0.5 (for the Elastic Net model)

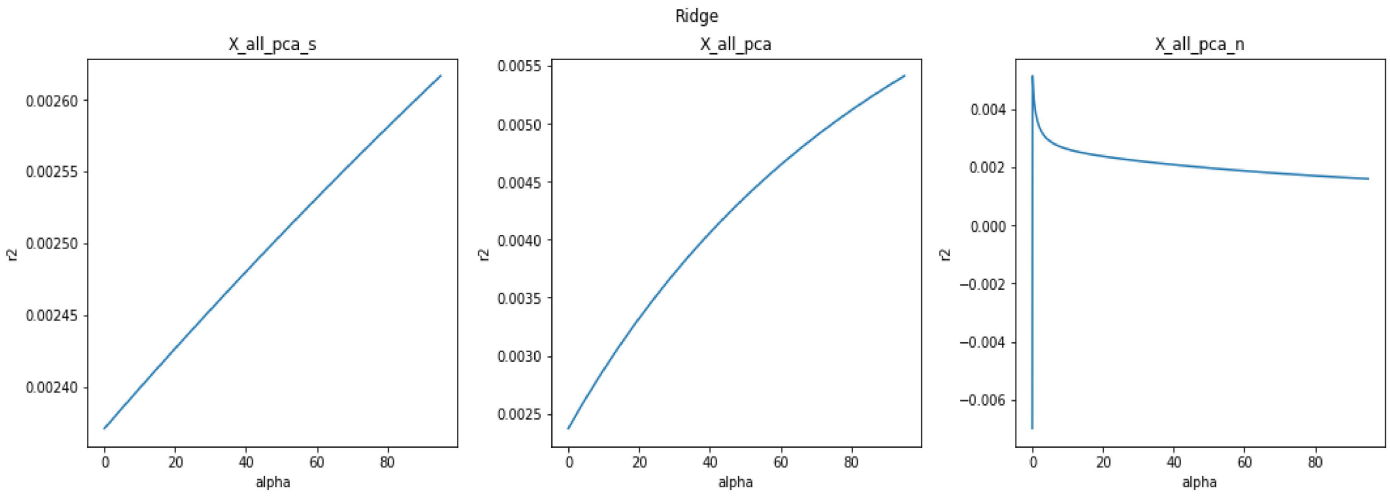
These baseline model parameters are consistent with the default parameters as specified in the **sklearn.linear_model** module.

3.G.3.2 Tuning Linear Models

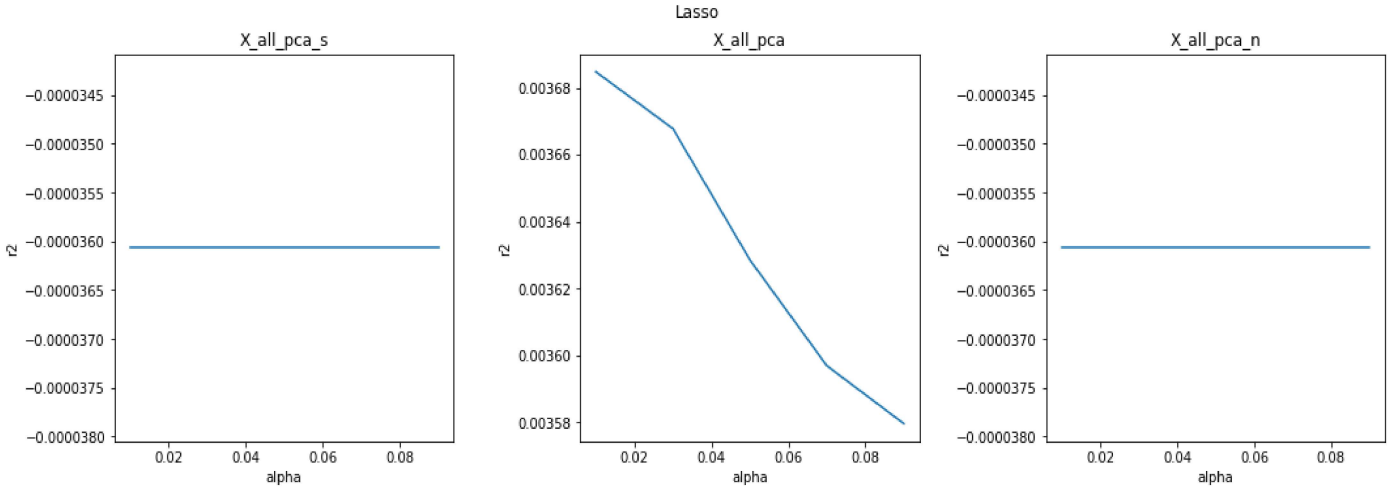
Parameter tuning for each model was performed by using the **GridSearchCV** function from the **sklearn.model_selection** module. Each set of grid parameters was first set by generating parameter values on either side of the default parameter and considering the performance of each model at each parameter value. Additional parameters are generated on a case by case basis with a view towards obtaining convergence toward the optimal parameter value.

The following graphics illustrate the parameter performance profile for each of the linear models

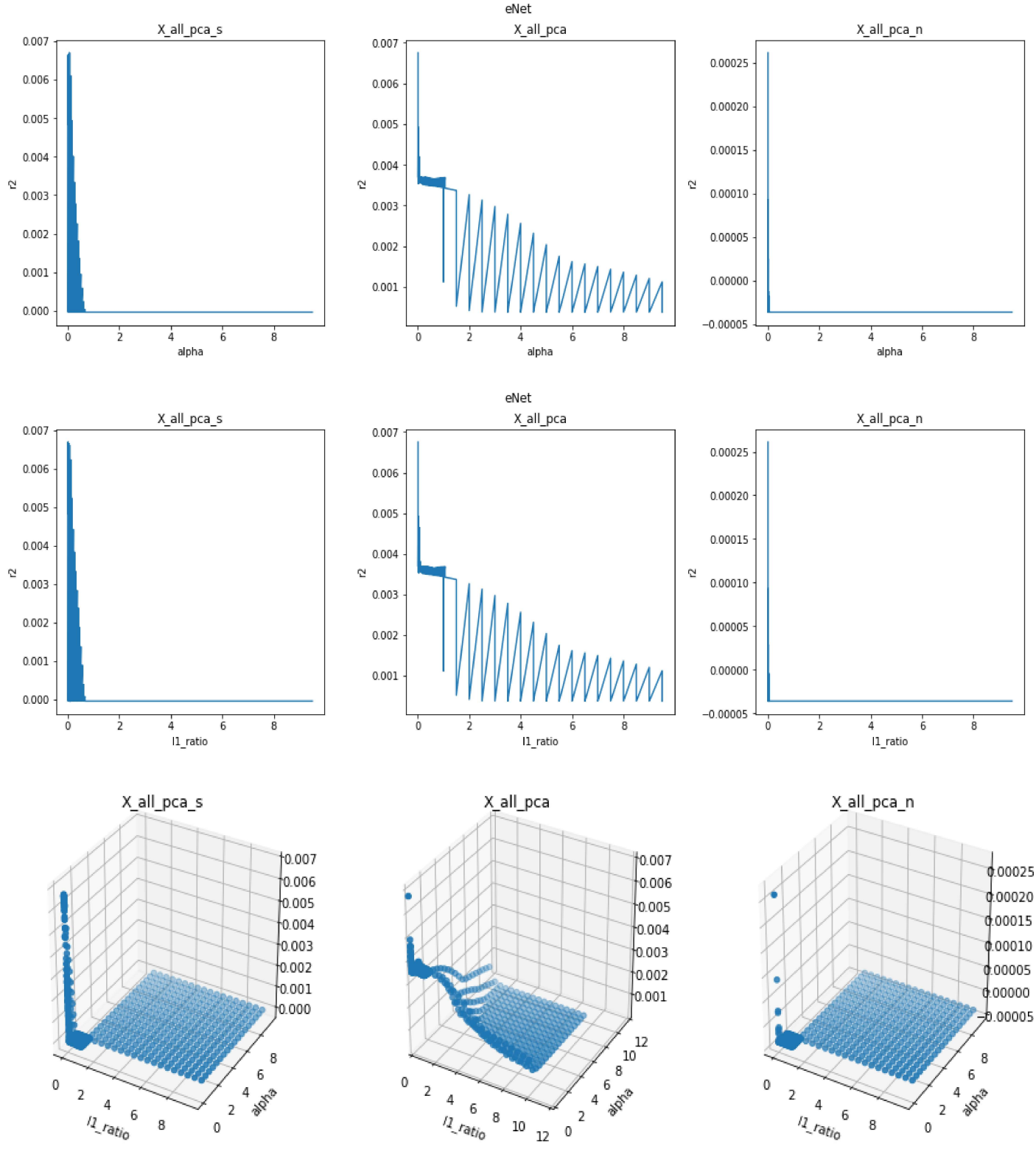
3.G.3.2.1 Ridge Regression



3.G.3.2.2 Lasso Regression



3.G.3.2.3 Elastic Net Regression



2.G.3.3 Best Parameters

The following table summarizes the results from the tuning process.

Regression Model	Parameter	Default	X_all_pca_s	X_all_pca	X_all_pca_s	Comment
Linear	-	-	-	-	-	No regularization term
Ridge	alpha	1.0	95.0	95.0	0.10	
Lasso	alpha	1.0	0.01	0.01	0.01	
Elastic Net	alpha	1.0	0.06	0.01	0.01	
Elastic Net	l1_ratio	0.5	0.01	0.01	0.01	

3.H Final Model

3.H.1 Linear Model Performance

3.H.1.1 Baseline vs Optimized Results on Test Data

Most of the tuned models performed better than their untuned counterparts on the test data, with the exception of the ridge model on the non normalized, non standardized dataset.

The best linear model over all the datasets was the **tuned Elastic Net model estimated using standardized data.**

Interestingly, the R^2 value that was achieved by the PCA model of 0.0101662 during the feature selection process is higher than that of both the best linear model and the best neural network model.

Normalized

```
==================== X_all_pca_n ======================
           Data     basis model type       mse        r2
16  X_all_pca_n  baseline     linear  0.024457  0.007358
17  X_all_pca_n     tuned     linear  0.024457  0.007358
18  X_all_pca_n  baseline      ridge  0.024472  0.006745
19  X_all_pca_n     tuned      ridge  0.024445  0.007838
20  X_all_pca_n  baseline      lasso  0.024639 -0.000034
21  X_all_pca_n     tuned      lasso  0.024639 -0.000034
22  X_all_pca_n  baseline       eNet  0.024639 -0.000034
23  X_all_pca_n     tuned       eNet  0.024612  0.001075

Best r2 score: 0.00783839
Best model: ridge
Best basis: tuned
```

Non Standardized, Non Normalized

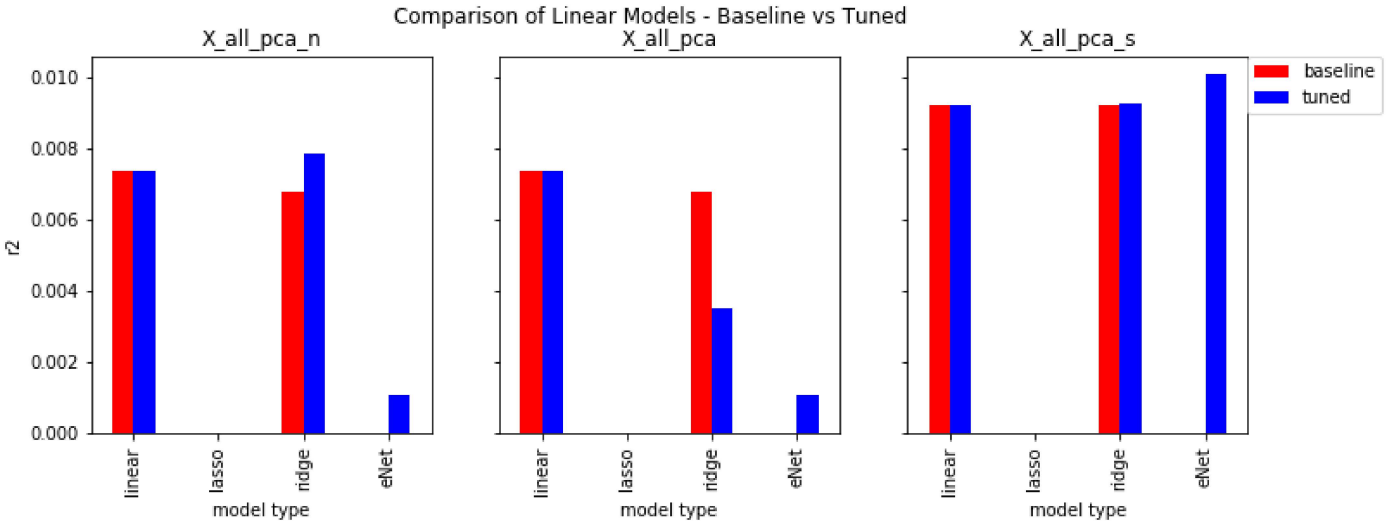
```
==================== X_all_pca ======================
          Data     basis model type       mse        r2
8   X_all_pca  baseline     linear  0.024457  0.007358
9   X_all_pca     tuned     linear  0.024457  0.007358
10  X_all_pca  baseline      ridge  0.024472  0.006745
11  X_all_pca     tuned      ridge  0.024552  0.003511
12  X_all_pca  baseline      lasso  0.024639 -0.000034
13  X_all_pca     tuned      lasso  0.024639 -0.000034
14  X_all_pca  baseline       eNet  0.024639 -0.000034
15  X_all_pca     tuned       eNet  0.024612  0.001075

Best r2 score: 0.00735769
Best model: linear
Best basis: baseline
```

Standardized

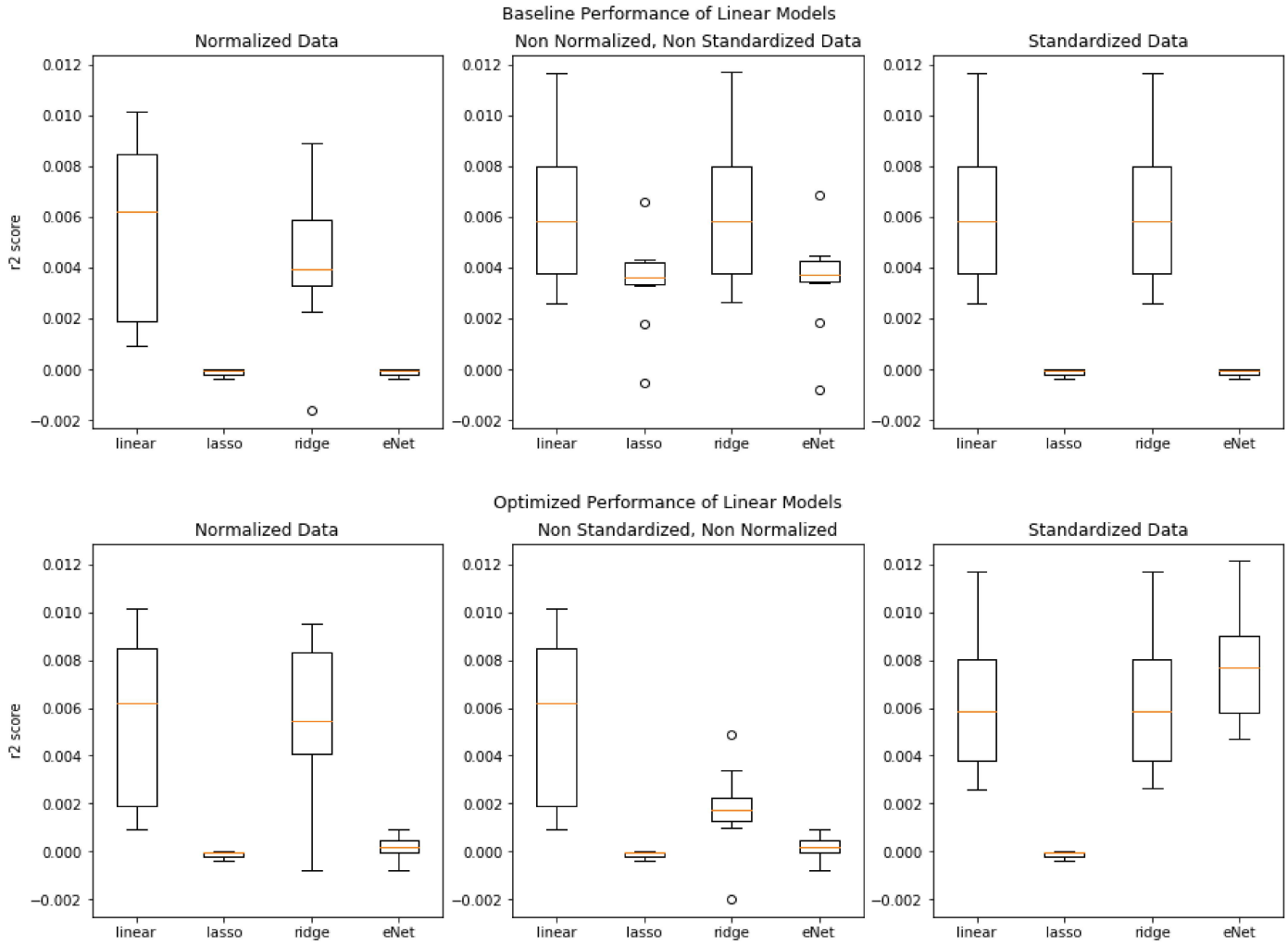
```
==================== X_all_pca_s ======================
          Data     basis model type       mse        r2
0  X_all_pca_s  baseline     linear  0.024411  0.009232
1  X_all_pca_s     tuned     linear  0.024411  0.009232
2  X_all_pca_s  baseline      ridge  0.024411  0.009232
3  X_all_pca_s     tuned      ridge  0.024410  0.009243
4  X_all_pca_s  baseline      lasso  0.024639 -0.000034
5  X_all_pca_s     tuned      lasso  0.024639 -0.000034
6  X_all_pca_s  baseline       eNet  0.024639 -0.000034
7  X_all_pca_s     tuned       eNet  0.024390  0.010076

Best r2 score: 0.01007631
Best model: eNet
Best basis: tuned
```



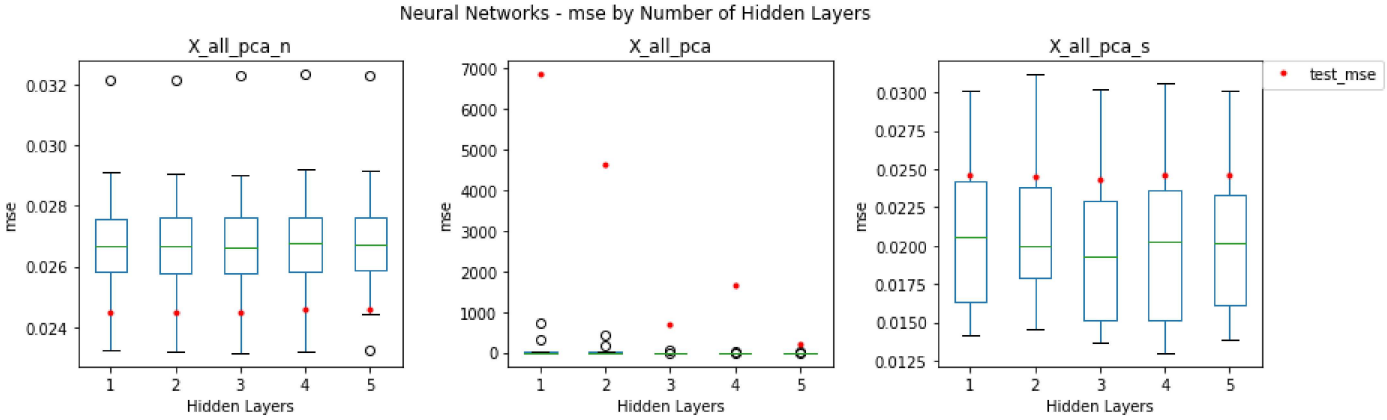
3.H.1.2 Linear Models - training results

The following graphic illustrates the performance of the linear models on training data. Overall, the baseline vs optimized models similar trends in performance with the exception of the Elastic Net model on standardized data whichis akin to its performance on the test data.



3.H.2 Neural Network Performance

The best linear model over all the datasets was the **3-layer model estimated using standardized data.**

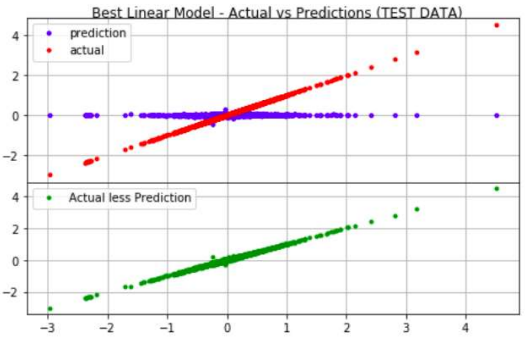
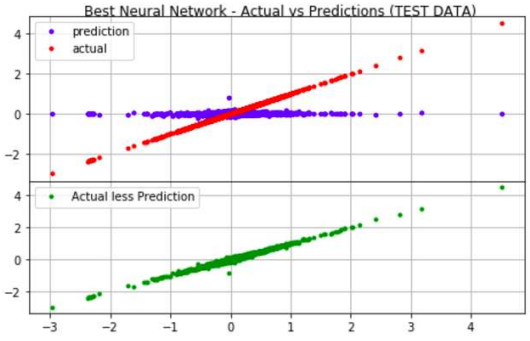


Normalized	Non Standardized, Non Normalized	Standardized
================== X_all_pca_n ============ Hidden Layers test_mse 1 0.024485 2 0.024499 3 0.024492 4 0.024598 5 0.024586	================== X_all_pca ================== Hidden Layers test_mse 1 6866.218601 2 4617.136059 3 683.419925 4 1652.845014 5 211.843375	================== X_all_pca_s =========== Hidden Layers test_mse 1 0.024649 2 0.024476 3 0.024342 4 0.024627 5 0.024613

The best neural network performance is summarized below:

Best Model by Dataset	Overall Best
==================== X_all_pca_s ==================== Best test_mse score: 0.02434236 Best model (number of layers): 3 Best dataset: X_all_pca_s ==================== X_all_pca ==================== Best test_mse score: 211.84337533 Best model (number of layers): 5 Best dataset: X_all_pca ==================== X_all_pca_n ==================== Best test_mse score: 0.02448455 Best model (number of layers): 1 Best dataset: X_all_pca_n	==================== Best Neural Network Model ==================== Best test_mse score: 0.02434236 Best model (number of layers): 3 Best dataset: X_all_pca_s

3.H.3 Neural Network vs Linear Model

Linear Model	Neural Network
================ BEST LINEAR MODEL PERFORMANCE ON TEST SET: ================ r2_score: 0.01007631 Best mse: 0.02438977 	================ BEST NEURAL NETWORK PERFORMANCE ON TEST SET: ================ r2_score: 0.00894968 Best mse: 0.02441753 

Given the higher R^2 score and lower **mse**, the results above indicate the best linear model has performed better than the best neural network model.

However, it must be noted that the R^2 scores are very poor for both models. This is reflected in the Actual vs Predictions graph, where predictions seem to get poorer and poorer the further away from zero the actual values are.

4. Conclusion

4.1 Absolute Performance

The best linear model (i.e. Elastic Net) has performed better than the best neural network model (number of hidden layers=3) when assessed againt the R^2 and mse metrics. Both performed best using standardized data. However, the following items are noteworthy:

- Overall, the R^2 scores from all the models considered have been poor i.e. none are more than 1.0%.
- The R^2 value achieved by the PCA approach that was used for feature selection is actually highest than the best score achieved by the linear models and the neural networks.
- The margin of difference between the R^2 scores for the best linear model and neural network is small. Greater levels of discrepancy has been observed when training the linear models using different volumes of data.

Overall, these factors indicate that using the approach described in this proposal has not resulted in any useful predictive model for the problem at hand.

4.3 Performance Against Benchmark

Item	Benchmark	Linear	Neural Network
R^2	-0.00003415	0.01007631	0.00894968
mse	0.02463887	0.02438977	0.02441753

Overall, the best linear and neural network models outperform the benchmark model, as seen in the higher R^2 values and lower mse values. However, as previously opined, the R^2 figures for both models are very small (no more than 1%) and are hence not indicative of predictive performance.

4.2 Model Robustness

Apart from poor performance on the problem at hand, there is also doubt on the linear model's robustness. The linear models have been found to produce significantly different results when trained on different volumes of the same dataset. This suggests the model is possibly overfitting to the data or that there is insufficient data for the number of features for the model to reach a stable state. This is set out in detail in section 6.

The following table summarizes insights that are specific to this project:

Item	Description
Data Transformation	Standardisation produces the best results
Missing values	Even sparse features have information content as the PCA approach preferred a complete dataset rather than one with less sparse features
Feature selection	PCA provided better results than Feature Importance via a standard XGBoost model or a Random Forest model
Neural Network	A deeper network architecture could lead to poorer results possibly due to overfitting

5. Limitations and Improvements

Student adequately summarizes the end-to-end problem solution and discusses one or two particular aspects of the project they found interesting or difficult.

The following are limitations and improvements that can be made to the workflow/process:

- **implementation issues:** Complications occurring during the coding process: inability to use certain built in functions such as sklearn's cross_val_score with Keras' KerasRegressor wrapper. This necessitated the need to write cross validation code from first principles instead. An example of this is the **`cross_val`** function written for the neural network section of the project. Similarly, unable to utilise the Keras wrapper's interface with the sklearn library to utilise the make_scorer function, hence I was unable to perform the neural network analysis training analysis on R^2 data.
- **`neural network evaluation metrics`** Only a limited number of evaluation metrics are available using keras, with seemingly none of them being either the mse or r2 variables.
- **`transaction date`** might have been utilised and a model which more adequately takes into account the temporal properties of that data used. <such as?>
- **`normality assumptions`** The linear models used make assumptions such as the normality of the target and error variables which might be violated by this project's dataset. Whilst a cursory visual check of the distribution of the target variable (ie the logerror variable) was undertaken, deeper analysis on the estimated distribution of this variable could be undertaken to see if other distributions better describe this variable. Similarly, no attempt was made in this project to investigate the distribution of residuals for each model. An area of development would be to investigate the distribution of residuals for each model and to formulate individual minimization functions for each model. This is a challenging area. Investigations into these two areas could also prompt the use of more generalized linear models rather than the ones selected in this project.
- current approach may be limiting in that it assumes a feature space and trains the neural networks and the linear models on this space. This biases the linear models as the features were ultimately selected based on their performance on a baseline linear model. An alternative approach would be to train each set of models on the entire feature space, as it is possible the neural networks would be able to make better use of the other features that have been thrown out. This would be a feasible approach if more and better computing grunt and time was available for this project.
- **feature selection**: other approaches to feature selection could include grouping categorical features into more aggregated subgroups prior to performing one-hot encoding. An example where this might have been appropriate is the ***land_use_desc*** variable which had 1997 unique values. This feature may have been further aggregated into higher level categories such as **farming uses** or **residential uses**. This approach was not utilized due to limitations in time and subject matter expertise.
- **feature selection**: a recursive feature selection method may be applied a method the method of reducing the opposite method of
- **neural network architectures**: different topologies could have been experimented with by changing not just the depth, but also the width of the model i.e. changing the number of neurons in each layer. If processing resources were not limited, it might have been possible to try work out more optimal combinations of number of epochs as well as batch size.

- Similar analysis but aiming to predict the abs_logerror as the target variable.
- more study into different ways of performing feature selection, apart from the approach used in this project of:
  - PCA.
    - PCA results were run on a Linear Regression model. It would be more comprehensive if the same investigations were run on the other linear models with different parameter values. I note that the
    - The PCA approach could also have been run on neural network models with different configurations and architectures. However, that is considered out of scope of this project as the author has no access to more computing resources.
  - importance variable from running the xgboost and random forest algorithms.
    - These algorithms were run with default variables without too much analysis into the inputs into the models and their optimality for the task at hand. An avenue for further investigation is to build and refine different tree based models to see which would have been better suited for this prediction problem.
- **Missing Values**: Different methods used to deal with missing values:
  - ***non categorical variables***: missing values were imputed using the mean value from the feature set. Other investigations which could have been done: other ways of imputing the features, such as using the median or mode if the feature set.

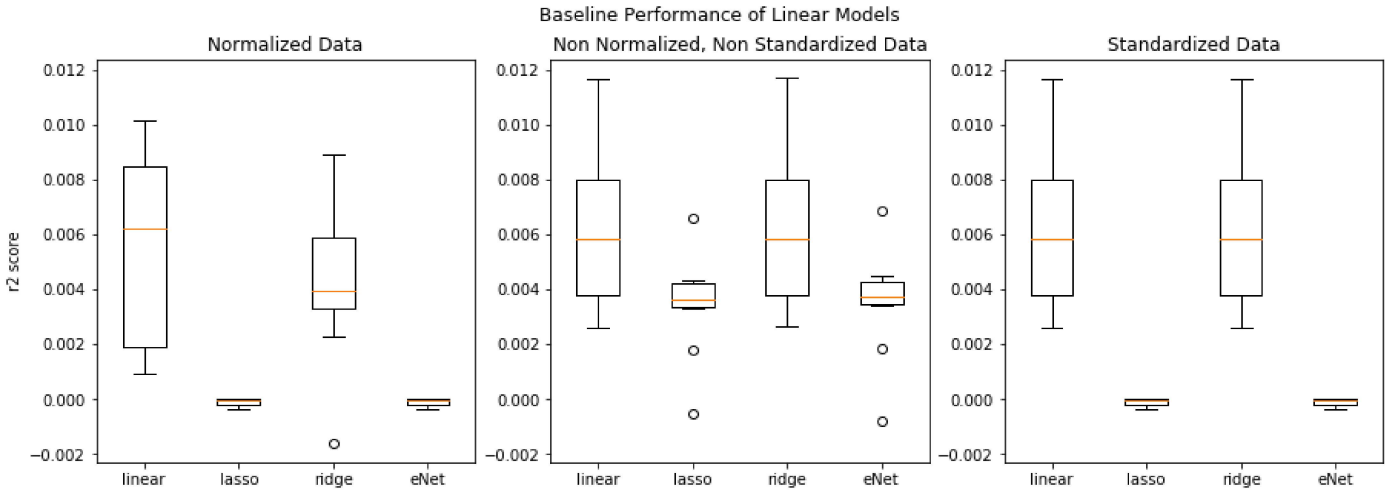
6. Further Investigation

Linear Model Cross Validation Results on different datasets

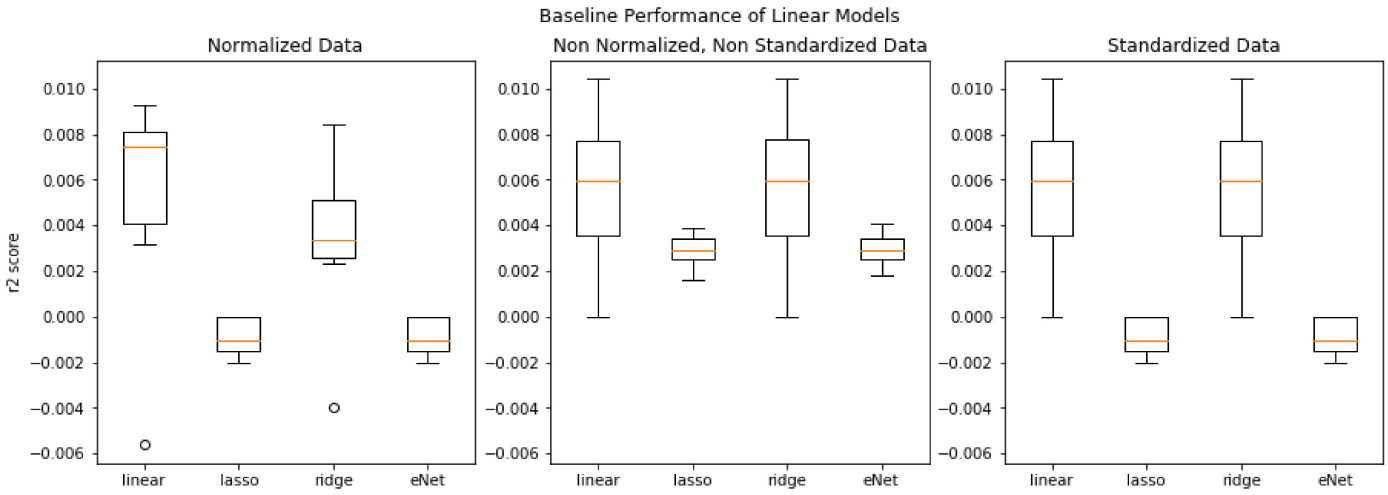
Ranges have been generated from the results of a 10-Fold cross validation approach. The results from using two different datasets (***training_data*** is a subset of ***ALL_data*** i.e. it is 80% of All_data). Given that training_data is a subset and not an entirely different dataset from ALL_data, the results suggest that the model is sensitive to the volume of data used and that one of the following might be happening:

1. There is insufficient data for the number of features
2. The models are overfitting to data.

training_data



ALL_data



References:

The following documents/links have been referenced in the course of this project:

[1] https://www.diva-portal.org/smash/get/diva2:131529/FULLTEXT01.pdf (https://www.diva-portal.org/smash/get/diva2:131529/FULLTEXT01.pdf)
[2] http://ageconsearch.umn.edu/bitstream/97781/2/2004-9-house%20price%20prediction.pdf (http://ageconsearch.umn.edu/bitstream/97781/2/2004-9-house%20price%20prediction.pdf)
[3] http://www.doc.ic.ac.uk/~mpd37/theses/2015_beng_aaron-ng.pdf (http://www.doc.ic.ac.uk/~mpd37/theses/2015_beng_aaron-ng.pdf)
[4] http://www.ecosystemvaluation.org/hedonic_pricing.htm (http://www.ecosystemvaluation.org/hedonic_pricing.htm)
[5] http://oppla.eu/sites/default/files/uploads/methodfactsheethedonic-property-pricing-method_0.pdf (http://oppla.eu/sites/default/files/uploads/methodfactsheethedonic-property-pricing-method_0.pdf)
[6] https://en.wikipedia.org/wiki/Hedonic_regression (https://en.wikipedia.org/wiki/Hedonic_regression)
[7] http://www.cbabuilder.co.uk/Quant5.html (http://www.cbabuilder.co.uk/Quant5.html)
[8] http://www.investopedia.com/terms/h/hedonicpricing.asp (http://www.investopedia.com/terms/h/hedonicpricing.asp)
[9] http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit (http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit)
[10] Machine Learning Mastery in Python by Jason Brownlee
[11] Deep Learning with Python by Jason Brownlee
[12] http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10/Lab%2010%20-%20Ridge%20Regression%20and%20the%20Lasso%20in%20Python.pdf (http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10/Lab%2010%20-%20Ridge%20Regression%20and%20the%20Lasso%20in%20Python.pdf)