Contents

Goal 1

Introduction

Linear regression has many applications within the realm of predictive analytics, including the prediction of housing prices. At the request of Sberbank – Russia's oldest and largest bank – our team was tasked with performing this type of analysis using data provided by the bank. Our team's first goal in the introductory project for the Applied Statistics: Inference & Modeling course in the Southern Methodist University Master of Science in Data Science (MSDS) program was to build a model employing linear regression techniques we've learned in this course - and the course prior - to date. Our team elected to use SAS as the preferred analysis platform for its analytical power and statistical reliability, as it has been proven stable for decades within the industry. We elected to use R for data wrangling and performing intermediary analysis as R performs quickly and scales easily when transformations are required across large dimensions of data.

The object of our first goal was to apply four distinct predictive models to assess the suitability of our parameter selections for determining the realty prices across a region of Russia relative in distance to the Kremlin – roughly 70 kilometers, or 43 miles. The measures of accuracy were applied in terms of Akaike Information Criterion, Standard Block Metric, k-fold internal Cross Validation measured in terms of the CV PRESS statistic, and external cross validation between competing models. Our approach outlined in this first objective is limited in that we were not permitted to use more advanced algorithms we will be exposed to later in the MSDS program; rather, in conjunction with the aforementioned linear regression techniques.

Description of the data with a table or a reference to a table in an Appendix.

The Sberbank real estate data set describes the sale of both investment and owner occupier properties from 2011-2015 in the Moscow, Russia metropolitan region. The data provided was comprised of 70 quantitative variables and three qualitative variables. Altogether, our data was split conservatively, with 75% for used for training the models and 25% used for testing. Altogether, these splits were applied to 19,835 observations. While data originally included 25,471 observations, 5,636 were excluded in the analysis and transformation process after being identified as either outliers or possibly the result of data entry or survey sampling error. Within the first objective of the project, we determined sale prices (price_doc) as the response of 69 selected linear ("main effects") and interactive variables, featuring proportional imputation of missing values and logarithmic and square root transformations of the numeric variables, as well as one-hot encoding for the categorical variables. For reference, an example of the proportional transformation is explicated in the appendix.

A complete list of the variables provided is in the data dictionary section of the appendix. From this list, we excluded from our model the variables timestamp, full_sq, material, build_year, raion_popul, children_preschool, preschool_quota, hospital_beds_raion, office_raion, metro_min_avto, metro_min_walk, metro_km_walk, railroad_station_walk_km, railroad_station_walk_min, railroad_station_avto_min, public_transport_station_min_walk, public_trans_station_time_walk and basketball_km.

Data Cleaning / Wrangling (any renaming of variables or standardizing of values.)

The variables with missing observations for which we provided imputations were floor of building (floor), number of floors in any given building (max_floor), total living area, excluding balconies and other non-residential areas (life_sq), kitchen area (kitch_sq), number of living rooms (num_room), number of seats in pre-school organizations (preschool_quota), walking time to the nearest railroad station (railroad_station_walk_min) and the ID of that railroad station (ID_railroad_station_walk), walking distance to the nearest railroad (railroad_station_walk_km), and shares of local buildings constructed within particular time periods - build_count_before_1920, build_count_1921-1945, build_count_1946-1970, build_count_1971-1995, and build_count_after_1995 - (metro_min_walk), and (metro_km_walk). A table of our missing values and counts can be seen in the appendix.

Next, we removed variables for wall material (material), number of hospital beds within the district (hospital_beds_raion, and the year build of buildings (build_year) as these contained too much missing information to provide imputation – we considered imputing these variables, but determined the risk of imputing outweighed the benefits to modeling. We also dropped count of preschools (preschool_quota) and walk time to nearest public transportation station (public_trans_station_time_walk) as these were not in the projectionData.csv file.

We furthermore removed observations for imputed values of life_sq and kitch_sq where these exceeded the total area (again, designated by full_sq). These imputations – originally calculated as a mean response of their total non-null values proportional to corresponding total square meters – exceeded total square meters and were thus determined to be impractical imputations.

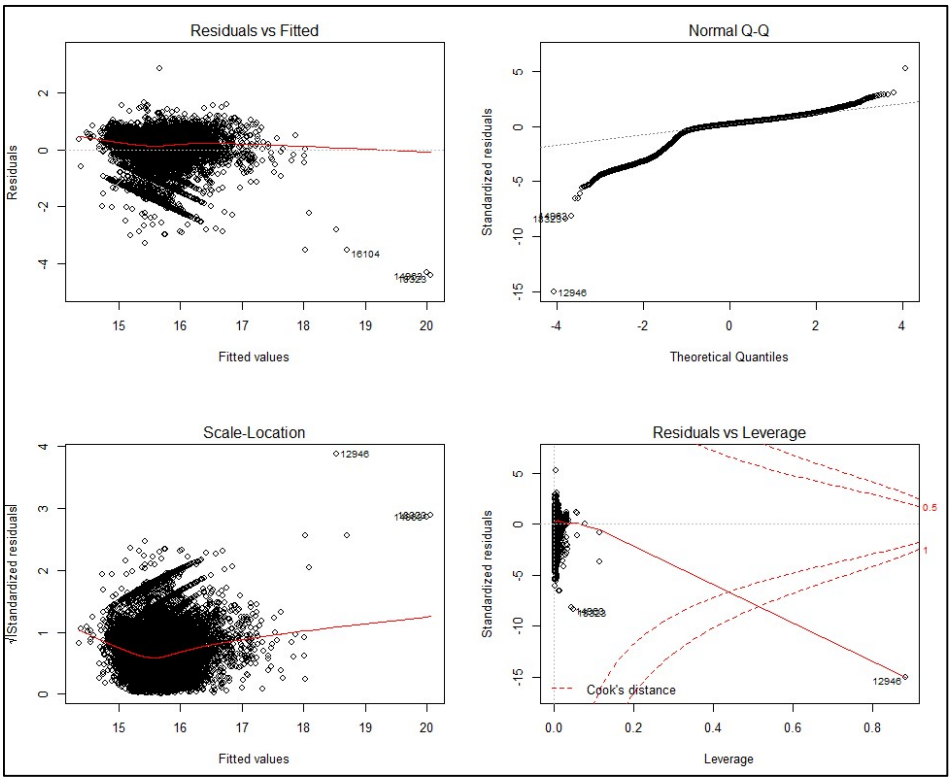
Finally, we provided logarithmic (of varying bases) transformations to floor and price_doc and square root transformations of values raised to various exponentials within the predictor features kitch_sq, and office_raion. We considered further transformations, but determined this would increase the risk of over-fitting once our chosen model was applied to new data, as done for the predictions provided using our file projectionData.csv. This panned to likely be a reasonable decision as evidenced by the very close adjusted R-squared value in our projected data (44%, roughly) compared to our modeling data (46%, roughly). Our approach was to make as much use of the available data as possible, but not to the extent that we would have over-fit modeling in our project. While this is not a guarantee in any statistical project, we feel confident in our efforts herein to mitigate this risk. Sample histograms for our variables' transformations can be seen in the appendix.

Individual Variable Outlier Identification and Handling

Because of the high volume of imputations and transformations to non-imputed variables, we determined, with the assistance of analyzing histograms of the individual predictors, that the best approach would be to run the models without first removing individual observations. Instead, we removed for key variables – based on their significance in predictive power as indicated by the correlation matrix – where the values were extreme. For example, where the kitchen was equal to less than two square meters of the total square meters or where living space was greater than the full square meters available, we removed these observations (2314 observations for life_sq and 68 observations for kitch_sq).

Checking Assumptions & Model Outlier Identification and Handling

Following the development of our models (more on this later), we analyzed diagnostics plots to determine what further outlying observations should be removed. We performed this analysis in R due to the support R provided; we encountered obstacles where the large dimensions of our data and model prevented us from visualizing these diagnostics in SAS. Although our missing data was handled en masse earlier on in our initial individual predictor variable exploratory data analysis (EDA), we determined the most appropriate method would be to assess the diagnostics following development of our final model in SAS and remove additional outliers following this analysis, as needed. We then reassessed our model after removing outliers to determine if one of our other models was better suited for the task assigned by Sberbank in objective one. The residuals diagnostics plots are output as follows:



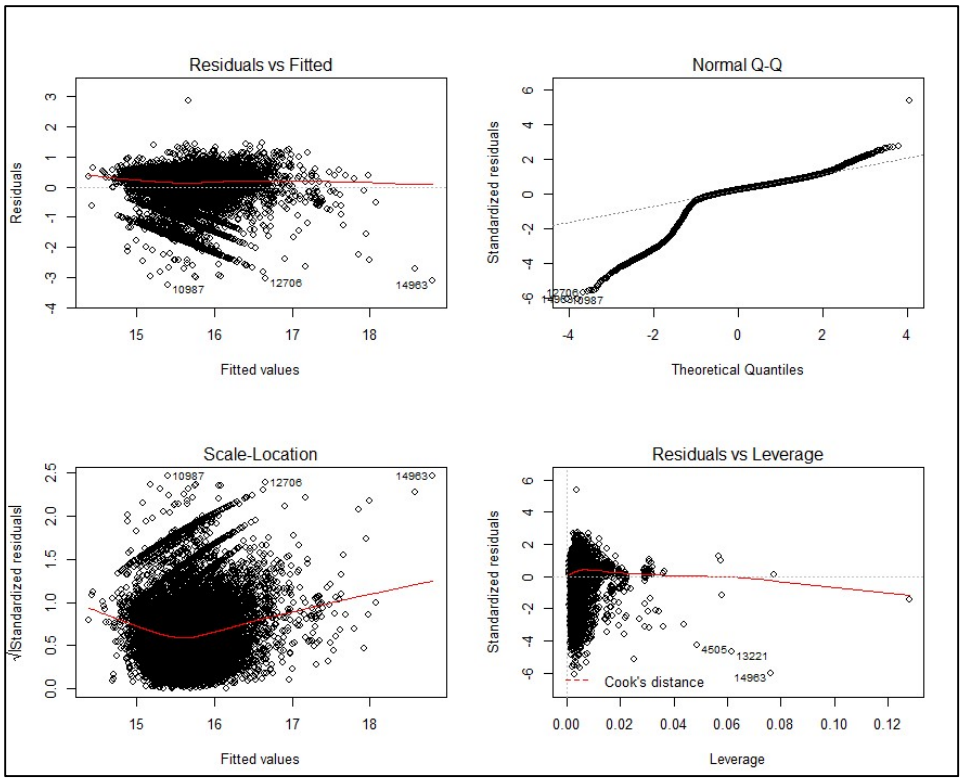
Linear regression diagnostics residuals plots for custom OLS model, prior to outlier removal

As indicated above, it appeared possible that some of our imputations – while necessary to capture a sufficient volume of data relevant to our entire analysis and predictions – resulted in a minor violation of the required non-constant variance assumptions for the regression model, as indicated in our **Residuals** (and Studentized Residuals, which has a higher consideration of standard error) **vs. Fitted** plot above. This could also be the result of sampling error or other phenomena that occurred concurrently with the timeframe relevant to our project's scope. Even so, there remained a healthy enough scatter among the residuals to assume constant variance in the model.

In analyzing the **QQ plot**, it is notably left-skewed. However, based on the relative adherence to normality of the remaining residuals in the distribution, we feel confident that the central limit theorem – especially with more data gathered over time – would resolve this kurtosis. With that in consideration,

it is important to note that this data was gathered over time and we therefore cannot safely assume a high likelihood of independence among the observations.

Finally, in analyzing the **Residuals vs. Leverage**, we identified several outlying observations that required removal from the dataset. Upon removing these, our leverage measurement – especially with respect to Cook's distance, improved significantly (below plots), which resulted in a 0.3% overall improvement on our final adjusted R-squared metric. While the leverage was improved, the removal of these observations had very little impact on the skewedness and variance within our model.



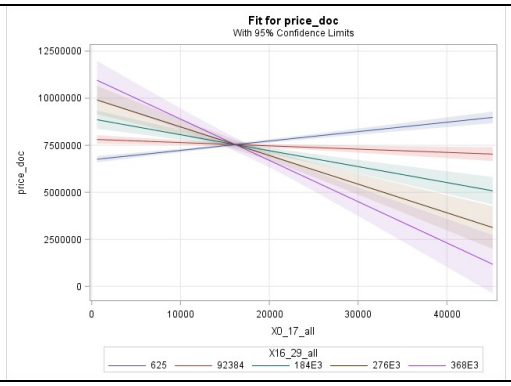
Linear regression diagnostics residuals plots for custom OLS model, following outlier removal

Variable Selection

Variable selection was initially managed by the Ordinary Least Squares (OLS) and LASSO regression techniques. Using all variables, excluding wall materials, hospital bed counts by district, and build year variables, which, as previously mentioned, were dropped for excessively high volumes of missing observations that were too risky for imputation. After processing all variables through these methods, Backward Elimination provided the highest adjusted R-squared and lowest AIC and SBC metrics. Consequently, we chose the variables suggested by this method. However, we provided further variable elimination based on direct analysis of the correlation matrix and Variable Inflation Factor (VIF) table's thresholds. We then considered interactions between linear terms based on our reasonable domain knowledge and reapplied the updated variable set through our OLS and LASSO selection processes again, removing those that were not significant. Following this, we removed interactions based on outputs from the interaction plots (dropping interaction terms where lines were parallel and their 95% confidence intervals did not overlap) and applied OLS and LASSO one more time. A complete list of

interaction plots can be viewed in the appendix, with a sample of significant interaction immediately below:

Interaction Plots



Interaction Parameter Estimates and Statistics

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6722496.745	89736.31432	74.91	<.0001
X0_17_all	50.317	5.12768	9.81	<.0001
X16_29_all	11.858	1.59684	7.43	<.0001
X0_17_all*X16_29_all	-0.001	0.00008	-8.74	<.0001

Potential Interactions

Given the variables present in the data set there may be combinations of variables that would make sense to paired together. Interactions such as school and distance to a school may be of interest to buyers with children; the build material and the year the home was built may be informative as different time periods may have had preferred building materials or availability. By including these types of interactions, a model may have better fit statistics suggesting that it can better describe and predict home pricing values.

Modeling

In expansion of the modeling methodology, this section will provide statistical output and discussion of results from our modeling. We tested four models, opting to make predictions based on our custom model. The models we developed were OLS Forward Selection, OLS Backward Elimination, OLS Stepwise Regression, and LASSO Regression. Ultimately, the aforementioned combination of OLS Backward Elimination, VIF analysis, and a correlation matrix with visual inspection of interaction plots proved the best. SAS code for these models can be found in the appendix.

Overall, the custom model outperformed all other models in terms of AIC, SBC, Internal k-fold Cross Validation, and external cross validation. Because these results changed slightly as testing was repeated across the folds, we considered this and a combination of other metrics, including Mean Square Errors produced from the models as well as the adjusted R-Square values across both internal and external cross validation as this data source diversity would increase chances of representing natural phenomena encountered under practical scenarios. Output tables are in the appendix for this section.

	LASSO Estimates using LASSO Variables Appendix table	**OLS (Backward) Estimates using LASSO Variables** Appendix table	**OLS (Backward) Estimates using OLS Variables** Appendix table	**OLS (Backward) Estimates using Custom OLS Variables** Appendix table
AIC (Internal)	621818	621818	620432	620750
AIC (External)	470167	470167	468947	469383
SBC (Internal)	602100	602100	601094	601467

SBC (External)	455277	455277	454424	454912
Internal Cross Validation (5 folds)	$3.0667x10^{17}$	$3.0698X10^{17}$	$5.1845x10^{17}$	$3.089X10^{17}$
External Cross Validation (5 folds)	$2.3004x10^{17}$	$2.2927x10^{17}$	$5.2081x10^{17}$	$2.3074x10^{17}$

Prediction

After considering information provided by internal and external cross validation, it was found that the custom model had the highest performance. Compared to the other models, the Custom model had the lowest RMSE, indicating that the amount of variation in predicted and observed values are smaller. Although the AIC and SBC values – which suggest better goodness of fit while not being over-fit – were between 0.05% and 0.1% better for the Backward OLS model, the backward OLS model produced the worst of the four models in terms of CV PRESS, suggesting the amount of error from the test and training sets was the worst for this model. However, since the Custom model performed consistently well, and better than all four on the CV PRESS metrics – here, indicating the amount of error from test and training sets was the least, we used the Custom model for predictions. Cross Validation and Model performance statistics can be found in the appendix, along with the Extra Sums of Squares outputs.

Data from the test set was cleaned in a similar manner to the training set and appended to the training set. After cleaning the test data there were 3,912 observations. Many of these observations were removed due to absence of values and to maintain consistency within the structures of the training and test sets. The Custom model was then re-run with the training/test set in SAS to allow for predictions with 95% confidence intervals. Please see the attached "SampleSubmission.csv" for the Custom model predictions.

Finally, our final RMSLE for this objective was 0.5980.

Goal 2

Introduction

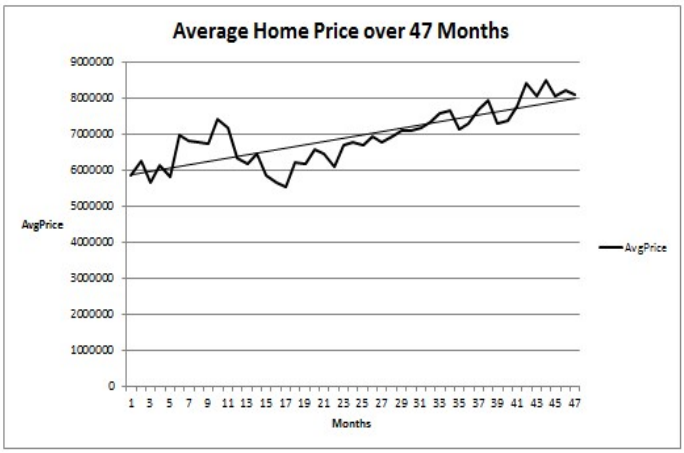
Many factors play a role in housing price and demand. Sberbank, a large bank in Russia provide historical housing data from August 2011 to June 2015 so we could predict the average property price for July 2015 through July 2016. The housing market appears to be stable during this time – although the country's economy was not, which can make forecasting difficult.

Data Wrangling

Using the data provided we extracted the timestamp and price values from the data provided by the bank. The timestamp required formatting into a proper "date/time" standard. Our goal was to recognize the average price by month-year and the variation over time. The date field was separated into three additional columns; month, day year. Then we merged the year and month columns together to capture the average price by month-year then this was numbered by in ascending order, example below.

	A	B	C
1	MonthNumber	monthYear	AvgPrice
2	1	201108	5850000
3	2	201109	6255310.057
4	3	201110	5667466.103
5	4	201111	6140269.116
6	5	201112	5812806.357

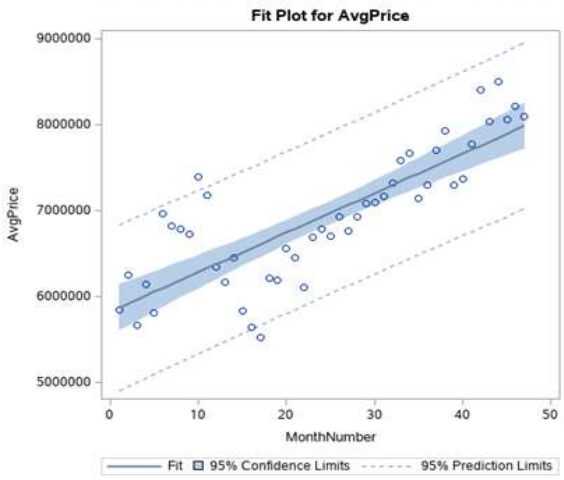
Looking at the data graphically we can see an upward trend of the average home price in the 47 months of data collected, however looking at the average price by year the trend is upward but has a sharp decline in 2015. To explain the trends we analyzed the 71 variables the bank provided and we need to consider Russia's volatile economy.

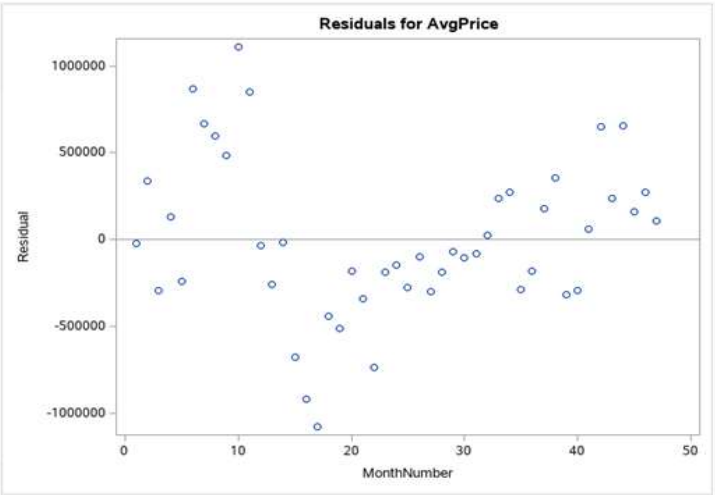


Looking at a simple linear regression model: $\widehat{AveragePrice} = \hat{\beta}_0 + \hat{\beta}_1 MonthNumber$. Can we assume the average home price will trend above or below the regression line for an extended amount of time, suggesting in serially correlated residuals? Upon visual inspection of the residual plot, there evidence the series will run above and below the trend line.

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	5824662	136857	42.56	<.0001
MonthNumber	MonthNumber	1	45969	4964.32907	9.26	<.0001





Serial Correlation Verification

This model has 1 variable, with 47 observations; given this information and the use of a Durban-Watson (DW) Table the lower and upper bounds used for identifying serial correlation are approximately: $DW_{LB} \approx 1.50$ and $DW_{UB} \approx 1.59$. For a set of data to exhibit serial correlation, a model needs that have a DW statistic that is less than the DW lower bound. If a model's DW statistic is between the upper and lower bound than the model is inconclusive for serial correlations; if a model's DW statistic is larger than the DW upper bound than that model does not have, or no longer has, serial correlation.

Upon initial inspection of an AR(0) model, the DW statistic of 0.7524 suggests that there is serial correlation between time observations. The ACF and PACF plots suggest that a lag of 1 is a strong candidate for serial correlation correction. When studying the PACF plot in more detail, the plot suggests that a lag of 5 may also be a candidate for serial correlation correction.

To screen for the correct number of lags to apply for time series, models for AR(1) – AR(5). The main requirement for selecting the best lag for time series requires that the DW statistic be greater than 1.59, and that ACF and PACF plots show that there is no longer serial correlation, AIC, and SBC statistics provide further evidence confirming the performance of a model. It is important to select the correct lag for prediction as it will adjust the standard error appropriately given the correlation between time values. If the incorrect lag is selected prediction estimates may not be as accurate and may also have larger error terms and confidence interval widths.

After running time series models for AR(1) – AR(5), the AR(1) model had a DW statistic of 2.1502 (Pr<DW) = 0.6422; AIC = 1340.68; and SBC = 1346.24. Although the PACF plots in the AR(0) suggested that there was a potential for a lag 5 model, the DW-statistics for AR(2) – AR(5) models all showed DW statistics that were less than the DW lower limit of 1.50. This confirmed that these models all still have serial correlation and are probably not the best models for describing this data. The AR(1) model's AIC and SBC statistics were also lower than the AR(2) – AR(5) models. This suggests that the AR(1) model is a better fit for the data. Please see the appendix for DW, AIC, SBC, ACF, and PACF statistics.

Time Series Predictions

Predictions were calculated using the AR(1) time series model based on criteria describe in the "Serial Correlation Verification" section. The predictions include an upper and lower bound for the 95% confidence intervals. There are no residuals as residuals cannot be calculated unless there are response values (AvgPrice) present.

Obs	prediction	resid	lower	upper	trend	MonthNumber	monthYear	AvgPrice

47	8160280.59	-68679.64	7396850.52	8923710.66	7999709.17	47	201506	8091600.9525
48	8103451.83	.	7338341.79	8868561.86	8046186.61	48	201507	.
49	8128350.65	.	7194375.86	9062325.43	8092664.05	49	201508	.
50	8161380.70	.	7148278.17	9174483.23	8139141.49	50	201509	.
51	8199477.98	.	7142064.79	9256891.18	8185618.93	51	201510	.
52	8240733.07	.	7154947.40	9326518.74	8232096.37	52	201511	.
53	8283956.03	.	7177810.24	9390101.83	8278573.81	53	201512	.
54	8328405.35	.	7206179.85	9450630.84	8325051.25	54	201601	.
55	8373618.90	.	7237681.70	9509556.09	8371528.68	55	201602	.
56	8419308.70	.	7270981.92	9567635.49	8418006.12	56	201603	.
57	8465295.31	.	7305301.15	9625289.46	8464483.56	57	201604	.
58	8511466.86	.	7340172.27	9682761.46	8510961.00	58	201605	.
59	8557753.68	.	7375311.02	9740196.35	8557438.44	59	201606	.

To see the full set of data including the training and test data, please see the attached “tsPrediction.csv” file included in the zip file.

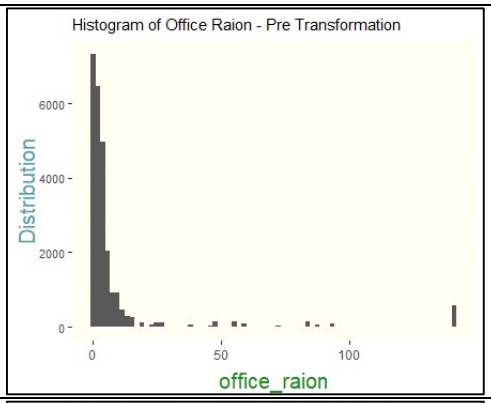
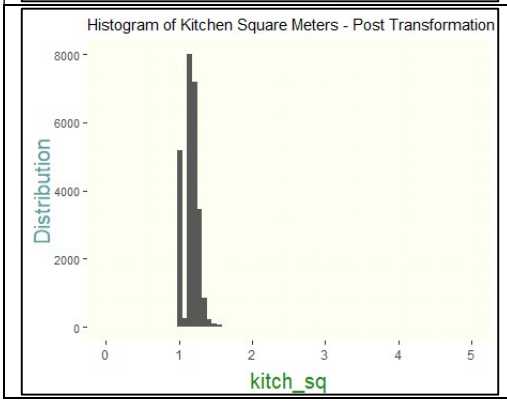
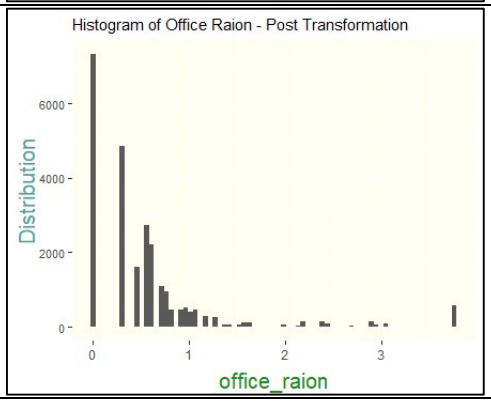
Appendix

Table of missing values

Feature Name	Count Missing	Feature Name	Count Missing
hospital_beds_raion	12,096	**build_count_before_1920**	4,195
build_year	11,392	**build_count_1921.1945**	4,195
max_floor	7,991	**build_count_1946.1970**	4,195
material	7,991	**build_count_1971.1995**	4,195
num_room	7,991	**build_count_after_1995**	4,195
kitch_sq	7,991	**floor**	146
preschool_quota	5,604	**metro_min_walk**	19
life_sq	5,333	**metro_km_walk**	19
build_count_block	4,195	**railroad_station_walk_km**	19
build_count_wood	4,195	**railroad_station_walk_min**	19
build_count_frame	4,195	**ID_railroad_station_walk**	19
build_count_brick	4,195		

Reference Table for Missing Values in modelingData.csv

Variable transformations samples: histograms

Kitchen Square Meters Log Transformation	**Office Raion Square Root Transformation**
	
	

Proportional Imputation Example

```
########## life_sq
df2.lifesq <- df[which(!is.na(df$life_sq)),]
life.Mean <- data.frame(df2.lifesq$life_sq/df2.lifesq$full_sq)
colnames(life.Mean) <- "percentofFull"
lifesqMultiplier <- mean(head(life.Mean$percentofFull,11000))
df[which(is.na(df$life_sq)),4] <- df[which(is.na(df$life_sq)),3]*lifesqMultiplier
df$life_sq <- as.numeric(df$life_sq)
dishonestLivingSpace <- (df$life_sq - df$full_sq)
df <- data.frame(dishonestLivingSpace, df)
df <- df[which(df$dishonestLivingSpace < 0),]
df <- subset(df, select = -c(dishonestLivingSpace))
##########
```

Correlation Matrix Example

Below is a portion of our correlation matrix used in the custom model. This table is too large to replicate in this document in its entirety. However, the code used to develop this is contained in section Goal 1 SAS Code: Modeling & Cross-Validation.

	price_doc	id	timestamp	full_sq	life_sq	floor	max_floor	num_room	kitch_sq	product_type	raion_popul	green_zone_part
price_doc price_doc	1.00000	0.12706	0.12363	0.30723	0.54569	0.11404	0.12010	0.26425	0.29385	-0.02040	0.04657	-0.01552
id id	0.12706	1.00000	0.97916	0.01663	0.03354	-0.00844	-0.01423	0.00516	-0.04901	0.05348	-0.00555	0.01299
timestamp timestamp	0.12363	0.97916	1.00000	0.01643	0.03472	0.00003	-0.01311	0.00529	-0.04990	0.07635	-0.01767	0.00584
full_sq full_sq	0.30723	0.01663	0.01643	1.00000	0.44234	0.08269	0.84351	0.94965	0.54860	0.10070	-0.03057	0.02087
life_sq life_sq	0.54569	0.03354	0.03472	0.44234	1.00000	0.10607	0.18300	0.39532	0.27885	0.19963	-0.07542	0.02520
floor floor	0.11404	-0.00844	0.00003	0.08269	0.10607	1.00000	0.20568	0.03315	0.06428	0.22320	-0.02927	-0.01262
max_floor max_floor	0.12010	-0.01423	-0.01311	0.84351	0.18300	0.20568	1.00000	0.79475	0.55589	0.09722	-0.02148	0.02654
num_room num_room	0.26425	0.00516	0.00529	0.94965	0.39532	0.03315	0.79475	1.00000	0.53215	0.02253	0.00402	0.01929
kitch_sq kitch_sq	0.29385	-0.04901	-0.04990	0.54860	0.27885	0.06428	0.55589	0.53215	1.00000	-0.35885	0.14393	0.04130
product_type product_type	-0.02040	0.05348	0.07635	0.10070	0.19963	0.22320	0.09722	0.02253	-0.35885	1.00000	-0.35263	-0.02990
raion_popul raion_popul	0.04657	-0.00555	-0.01767	-0.03057	-0.07542	-0.02927	-0.02148	0.00402	0.14393	-0.35263	1.00000	0.12366
green_zone_part green_zone_part	-0.01552	0.01299	0.00584	0.02087	0.02520	-0.01262	0.02654	0.01929	0.04130	-0.02990	0.12366	1.00000

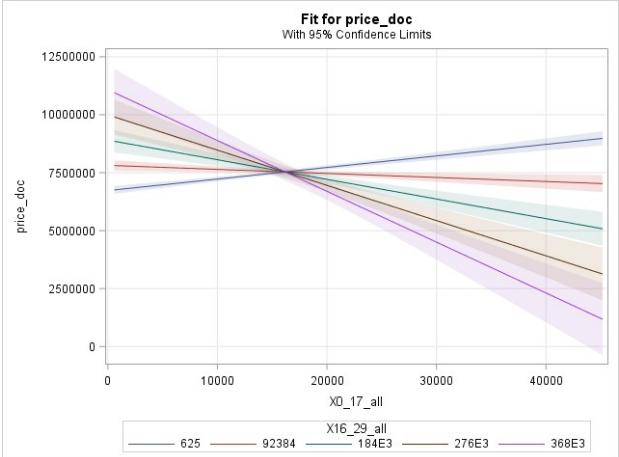
Variance Inflation Factor

Below is a portion of our variance inflation factor scores and table output. As with the correlation matrix, this table is too large to replicate in this document in its entirety. However, the code used to develop this is contained in section Goal 1 SAS Code: Modeling & Cross-Validation. For our custom model, we applied a threshold cutoff for most variables of a Tolerance value less than 0.1. Terms used in significant interactions that breached this threshold were still used in the model as this provides a degree of healthy error in a model, which in turn prevents over-fitting to the modeling data.

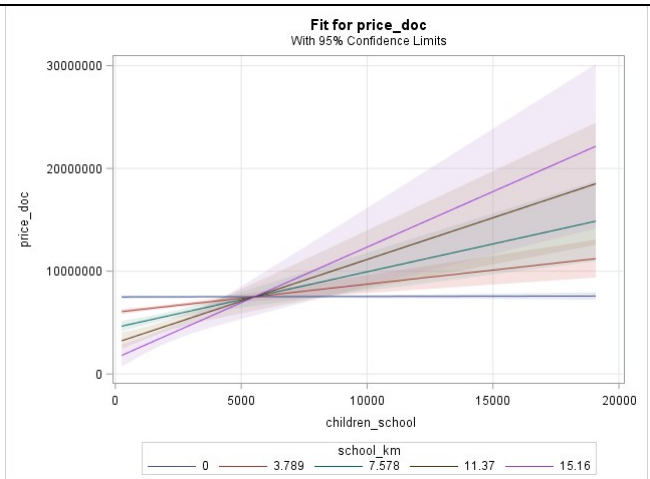
Parameter Estimates								
Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Tolerance	Variance Inflation
Intercept	Intercept	1	-6532056	525008	-12.44	<.0001	.	0
id	id	1	68.27688	3.14594	21.70	<.0001	0.97694	1.02361
life_sq	life_sq	B	148107	2186.27762	67.74	<.0001	0.63643	1.57126
floor	floor	B	765819	45893	16.69	<.0001	0.82679	1.20950
max_floor	max_floor	B	-60872	4940.04196	-12.32	<.0001	0.24349	4.10689
num_room	num_room	1	266811	32115	8.31	<.0001	0.26703	3.74495
kitch_sq	kitch_sq	B	8449982	411851	20.52	<.0001	0.43002	2.32547
product_type	product_type	B	440279	103931	4.24	<.0001	0.42628	2.34588
green_zone_part	green_zone_part	B	-1853163	255005	-7.27	<.0001	0.43994	2.27306
indust_part	indust_part	B	-3031984	310902	-9.75	<.0001	0.51052	1.95878
preschool_quota	preschool_quota	B	-12313	4002.28731	-3.08	0.0021	0.13412	7.45583
children_school	children_school	B	-408.75477	109.88984	-3.72	0.0002	0.00454	220.13674
healthcare_centers_raion	healthcare_centers_raion	1	25930	28220	0.92	0.3582	0.41394	2.41580
university_top_20_raion	university_top_20_raion	B	301476	100519	3.00	0.0027	0.33464	2.98830
shopping_centers_raion	shopping_centers_raion	B	-83135	11240	-7.40	<.0001	0.26419	3.78518
railroad_terminal_raion	railroad_terminal_raion	B	-1946728	209013	-9.31	<.0001	0.42896	2.33121
big_market_raion	big_market_raion	B	96335	158500	0.61	0.5433	0.72396	1.38129

Predictor Variables Interaction Plots

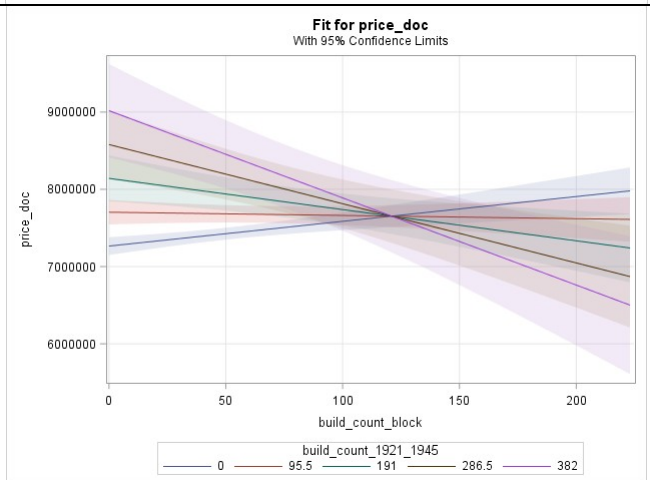
Interaction Plots between Predictor Variables	Estimates, Errors, Student-t Statistics, p-Values for Predictors



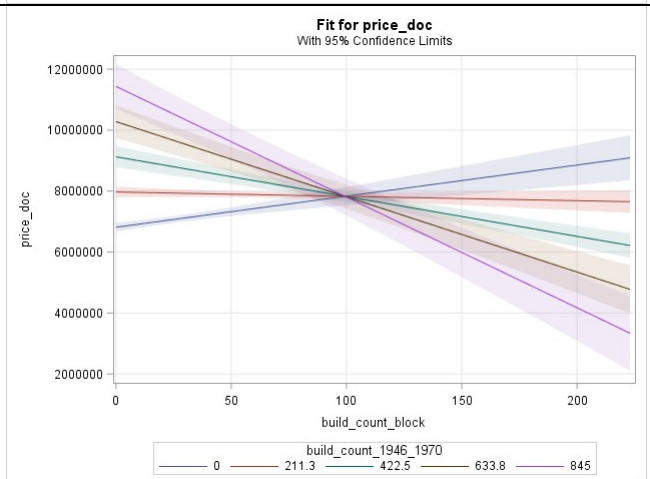
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6722496.745	89736.31432	74.91	<.0001
X0_17_all	50.317	5.12768	9.81	<.0001
X16_29_all	11.858	1.59684	7.43	<.0001
X0_17_all*X16_29_all	-0.001	0.00008	-8.74	<.0001



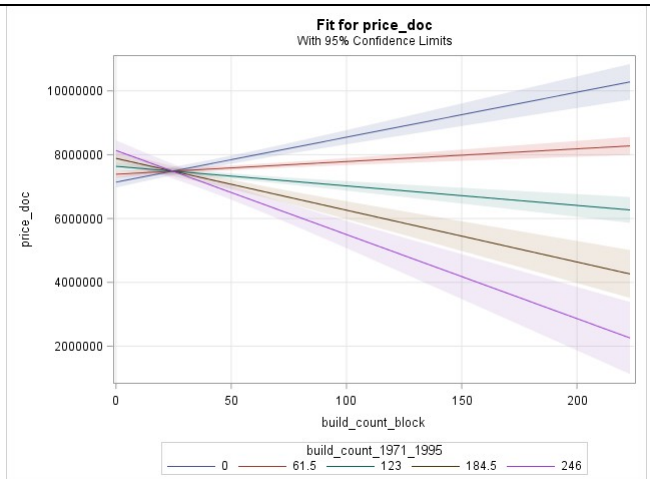
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7496958.248	81118.29796	92.42	<.0001
children_school	4.624	12.35193	0.37	0.7082
school_km	-392931.574	41312.60587	-9.51	<.0001
children_s*school_km	70.993	16.12351	4.40	<.0001



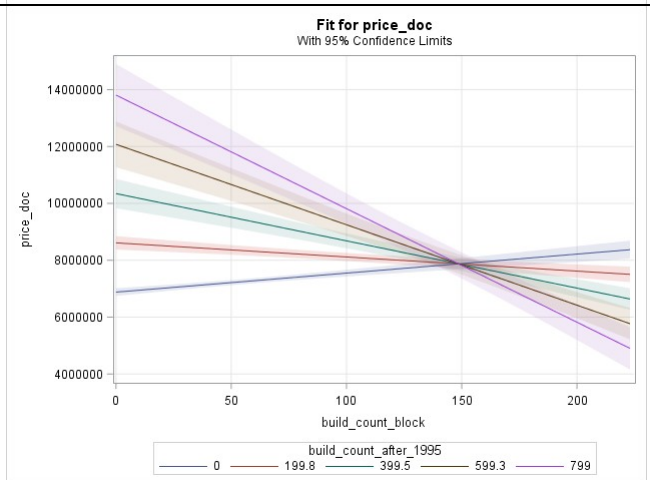
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7265815.776	58455.48390	124.30	<.0001
build_count_block	3204.990	863.31668	3.71	0.0002
build_count_1921_194	4585.718	852.93075	5.38	<.0001
build_cou*build_coun	-37.931	8.11408	-4.67	<.0001



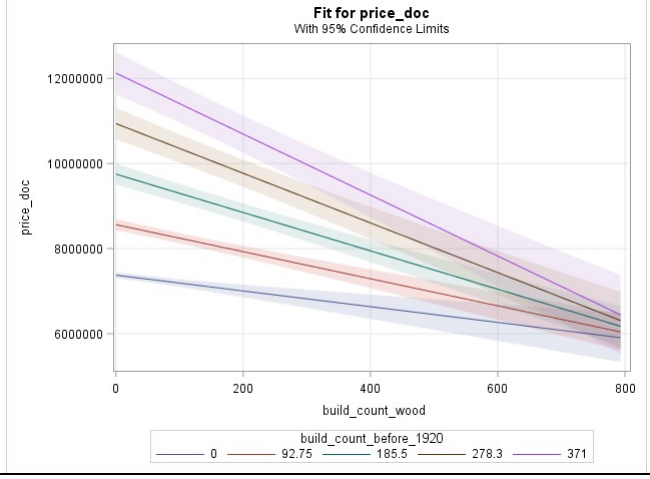
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6815357.931	73348.17177	92.92	<.0001
build_count_block	10205.651	1874.59132	5.44	<.0001
build_count_1946_197	5467.237	485.09060	11.27	<.0001
build_cou*build_coun	-55.043	5.82153	-9.46	<.0001



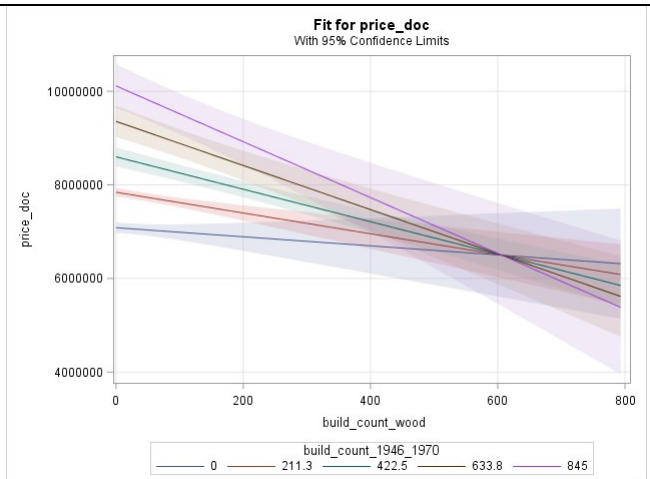
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7139221.473	84124.34752	84.87	<.0001
build_count_block	14093.678	1497.97823	9.41	<.0001
build_count_1971_199	4036.298	880.73819	4.58	<.0001
build_cou*build_coun	-164.376	17.30349	-9.50	<.0001



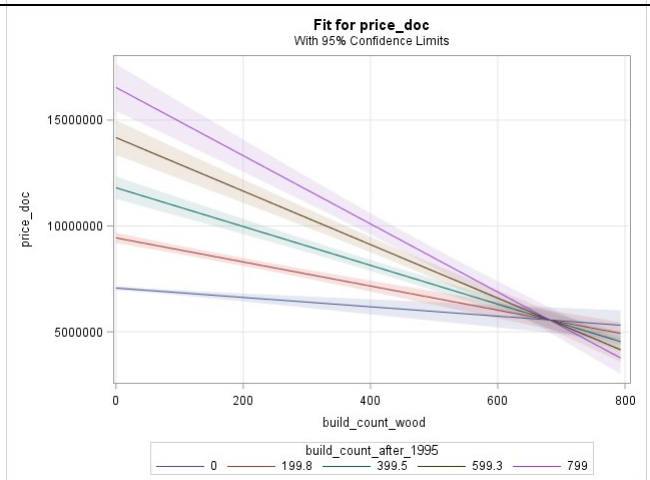
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6880680.118	68615.29625	100.28	<.0001
build_count_block	6686.790	929.48046	7.19	<.0001
build_count_after_19	8669.985	746.88577	11.61	<.0001
build_cou*build_coun	-58.349	5.02467	-11.61	<.0001



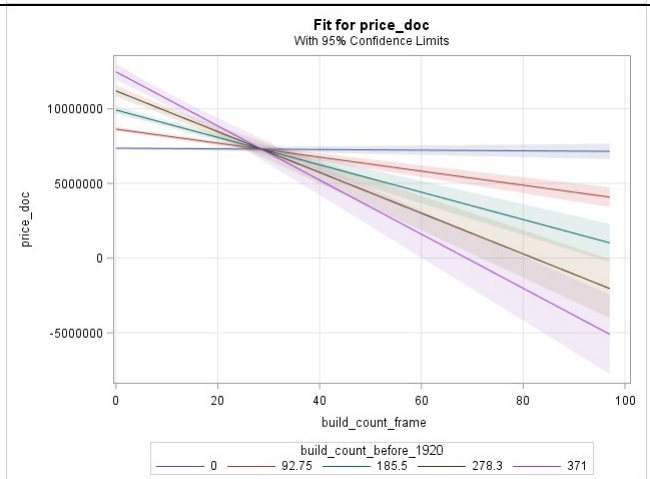
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7375296.649	38990.02960	189.16	<.0001
build_count_wood	-1849.435	379.43315	-4.87	<.0001
build_count_before_1	12798.671	701.03674	18.26	<.0001
build_cou*build_coun	-14.321	2.30077	-6.22	<.0001



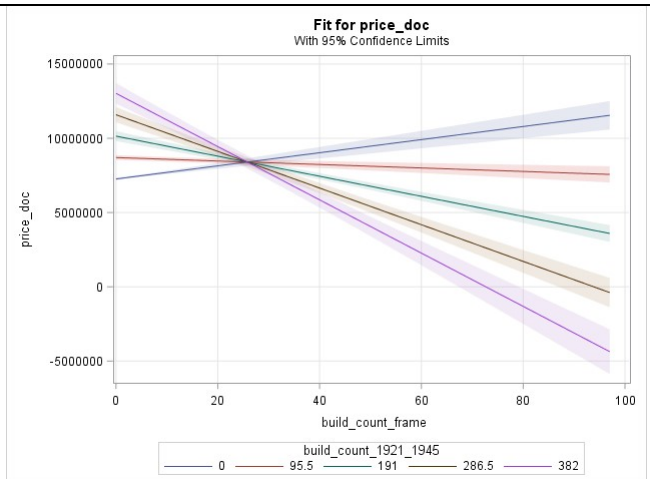
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7084682.959	57084.45097	124.11	<.0001
build_count_wood	-967.694	767.97015	-1.26	0.2077
build_count_1946_197	3586.585	329.25272	10.89	<.0001
build_cou*build_coun	-5.915	1.92820	-3.07	0.0022



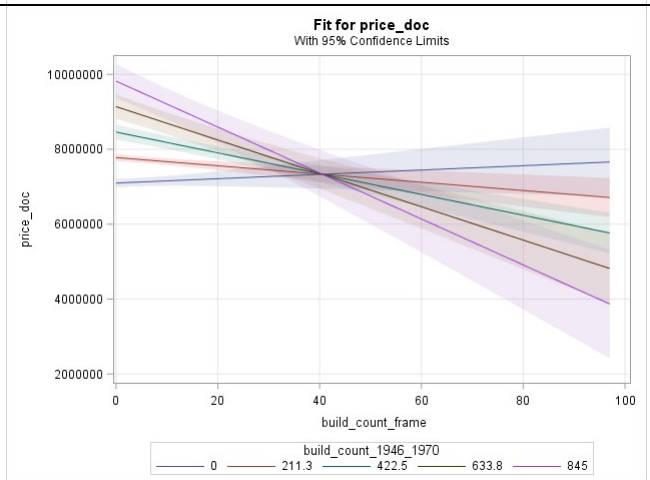
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7068296.374	48551.03322	145.58	<.0001
build_count_wood	-2211.654	457.76842	-4.83	<.0001
build_count_after_19	11859.290	744.18461	15.94	<.0001
build_cou*build_coun	-17.397	1.45424	-11.96	<.0001



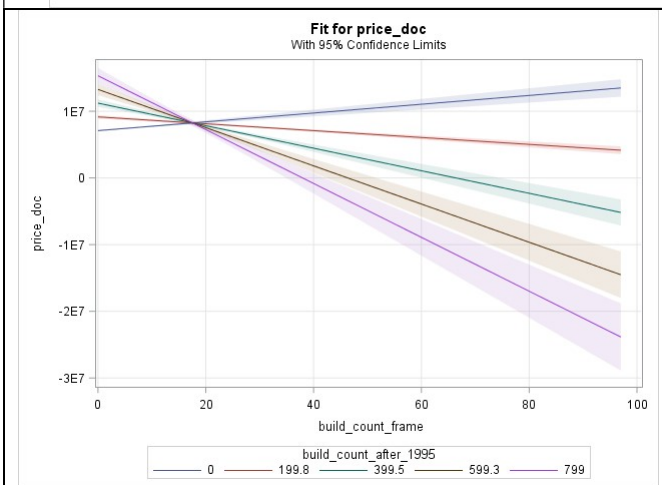
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7352865.081	38932.60132	188.86	<.0001
build_count_frame	-2157.968	2773.12042	-0.78	0.4365
build_count_before_1	13784.975	737.93733	18.68	<.0001
build_cou*build_coun	-482.289	44.18424	-10.92	<.0001



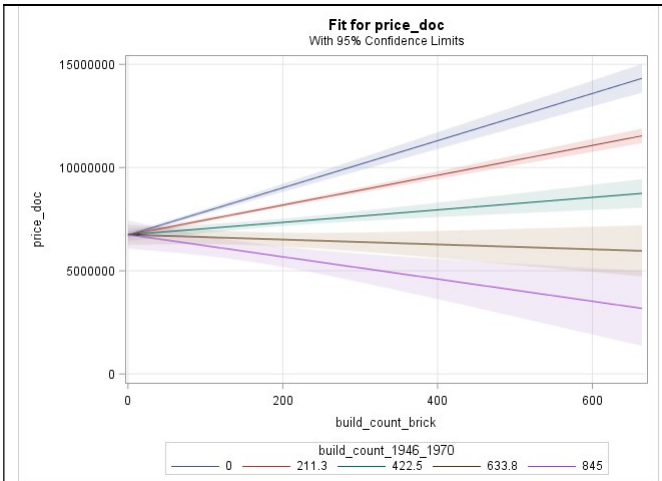
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7261397.781	41052.14304	176.88	<.0001
build_count_frame	44161.676	5150.99892	8.57	<.0001
build_count_1921_194	15089.430	958.40439	15.74	<.0001
build_cou*build_coun	-584.934	36.85870	-15.87	<.0001



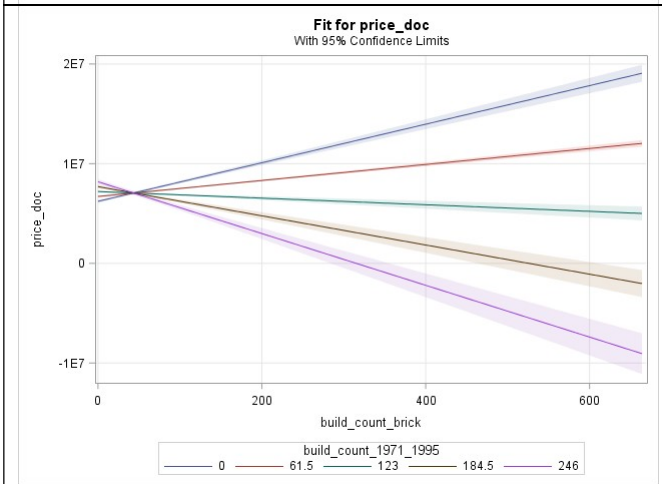
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7098142.417	57092.68093	124.33	<.0001
build_count_frame	5789.399	4918.46604	1.18	0.2392
build_count_1946_197	3220.803	322.22753	10.00	<.0001
build_cou*build_coun	-79.447	14.22334	-5.59	<.0001



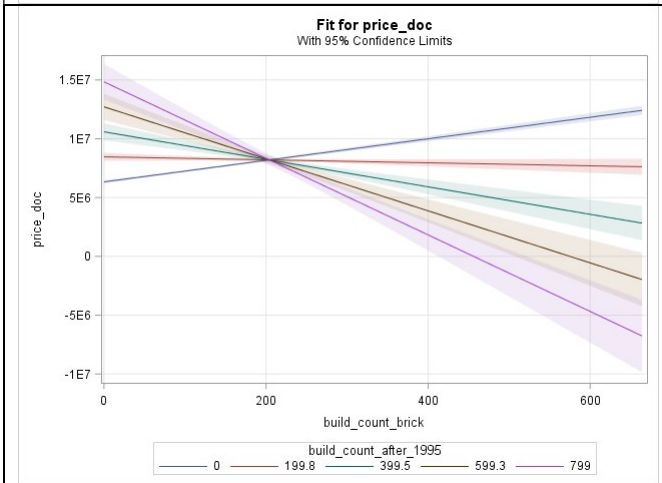
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	7096973.883	50126.38029	141.58	<.0001
build_count_frame	65939.662	7167.87457	9.20	<.0001
build_count_after_19	10300.697	778.04935	13.24	<.0001
build_cou*build_coun	-588.054	48.16212	-12.21	<.0001



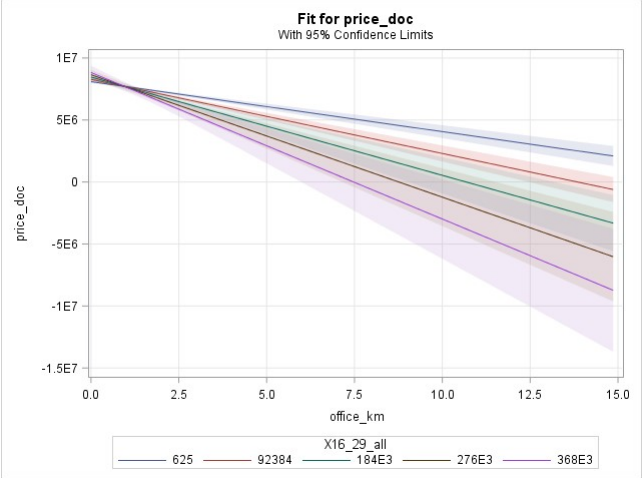
Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6731946.258	63487.06106	106.04	<.0001
build_count_brick	11432.941	573.49493	19.94	<.0001
build_count_1946_197	24.746	457.40429	0.05	0.9569
build_cou*build_coun	-19.902	2.56060	-7.77	<.0001



Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6223639.263	76734.10756	81.11	<.0001
build_count_brick	19349.055	710.87610	27.22	<.0001
build_count_1971_199	7963.297	742.11529	10.73	<.0001
build_cou*build_coun	-184.197	9.37028	-19.66	<.0001



Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	6336832.943	63310.22224	100.09	<.0001
build_count_brick	9156.069	364.71439	25.10	<.0001
build_count_after_19	10644.978	1018.97741	10.45	<.0001
build_cou*build_coun	-52.180	4.58595	-11.38	<.0001



Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	8094353.522	55356.23735	146.22	<.0001
office_km	-402150.050	30115.06210	-13.35	<.0001
X16_29_all	1.999	0.85603	2.34	0.0195
office_km*X16_29_all	-2.120	0.55557	-3.82	0.0001

Data Dictionary

Attribute	Attribute Description	Data Type - Post Transformation
id	transaction id	integer
timestamp	date of transaction	time series
full_sq	total area in square meters, including loggias, balconies and other non-residential areas	integer
life_sq	living area in square meters, excluding loggias, balconies and other non-residential areas	numeric
floor	for apartments, floor of the building	numeric
max_floor	number of floors in the building	numeric
material	wall material	numeric
build_year	year building was constructed	numeric
num_room	number of living rooms	numeric
kitch_sq	kitchen area	numeric
product_type	owner-occupier purchase or investment	categorical factor with two levels; one-hot encoded
raion_popul	Number of municipality population. district	integer
green_zone_part	Proportion of area of greenery in the total area	numeric
indust_part	Share of industrial zones in area of the total area	numeric
children_preschool	Number of pre-school age population	integer
preschool_quota	Number of seats in pre-school organizations	numeric
children_school	Population of school-age children	integer
hospital_beds_raion	Number of hospital beds for the district	integer
healthcare_centers_raion	Number of healthcare centers in district	integer
university_top_20_raion	Number of higher education institutions in the top ten ranking of the Federal rank	integer
shopping_centers_raion	Number of malls and shopping centers in district	integer

office_raion	Number of malls and shopping centers in district	numeric
railroad_terminal_raion	Presence of the railroad terminal in district	categorical factor with two levels; one-hot encoded
big_market_raion	Presence of large grocery / wholesale markets	categorical factor with two levels; one-hot encoded
full_all	Total number of population in the municipality	integer
0_6_all	Population count of age 0-6 years	integer
7_14_all	Population count of age 7-14 years	integer
0_17_all	Population count of age 0-17 years	integer
16_29_all	Population count of age 16-19 years	integer
0_13_all	Population count of age 0-13 years	integer
build_count_block	Share of block buildings	integer
build_count_wood	Share of wood buildings	integer
build_count_frame	Share of frame buildings	integer
build_count_brick	Share of brick buildings	integer
build_count_before_1920	Share of buildings constructed before year 1920	integer
build_count_1921-1945	Share of buildings constructed between years 1921 and 1945	integer
build_count_1946-1970	Share of buildings constructed between 1946 and 1970	integer
build_count_1971-1995	Share of buildings constructed between 1971 and 1995	integer
build_count_after_1995	Share of buildings constructed after 1995	integer
metro_min_avto	Time to subway by car, min.	numeric
metro_km_avto	Distance to subway by car, km	numeric
metro_min_walk	Time to metro by foot	numeric
metro_km_walk	Distance to the metro, km	numeric
school_km	Distance to high school	numeric
park_km	Distance to park	numeric
green_zone_km	Distance to green zone	numeric
industrial_km	Distance to industrial zone	numeric
railroad_station_walk_km	Distance to the railroad station (walking)	numeric
railroad_station_walk_min	Time to the railroad station (walking)	numeric
ID_railroad_station_walk	Nearest railroad station id (walk)	integer
railroad_station_avto_km	Distance to the railroad station	numeric
railroad_station_avto_min	Time to the railroad station	numeric
public_transport_station_km	Distance to the public transport station (walk)	numeric
public_transport_station_min_walk	Time to the public transport station (walk)	numeric
kremlin_km	Distance to the city center (Kremlin)	numeric
big_road1_km	Distance to Nearest major road	numeric
big_road2_km	The distance to next distant major road	numeric
railroad_km	Distance to the railway / Moscow Central Ring / open areas Underground	numeric
bus_terminal_avto_km	Distance to bus terminal	numeric

big_market_km	Distance to grocery / wholesale markets	numeric
market_shop_km	Distance to markets and department stores	numeric
fitness_km	Distance to fitness	numeric
swim_pool_km	Distance to swimming pool	numeric
ice_rink_km	Distance to ice palace	numeric
stadium_km	Distance to stadium	numeric
basketball_km	Distance to the basketball courts	numeric
public_healthcare_km	Distance to public healthcare	numeric
university_km	Distance to universities	numeric
workplaces_km	Distance to workplaces	numeric
shopping_centers_km	Distance to shopping centers	numeric
office_km	Distance to business centers/ offices	numeric
big_church_km	Distance to large church	numeric
price_doc	sale price (this is the target variable)	numeric

Model Cross-Validations

LASSO Regression Using LASSO-Selected Variables – Model 1

External Cross Validation Using Test/Train Split

Root MSE	3905728
Dependent Mean	7503500
R-Square	0.4055
Adj R-Sq	0.4050
AIC	470167
AICC	470167
SBC	455277
ASE (Train)	1.524047E13
ASE (Test)	1.574238E13
CV PRESS	2.300366E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	13	1.558466E17	1.19882E16	785.87
Error	14980	2.285156E17	1.525471E13	
Corrected Total	14993	3.843622E17		

Internal Cross Validation Using Full Data

Root MSE	3919496
Dependent Mean	7483904
R-Square	0.4075
Adj R-Sq	0.4071
AIC	621818
AICC	621818
SBC	602100
CV PRESS	3.066611E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	13	2.093126E17	1.610097E16	1048.07
Error	19812	3.043609E17	1.536245E13	
Corrected Total	19825	5.136735E17		

OLS Backward Elimination Using LASSO Variables and OLS – Model 2

External Cross Validation Using Test/Train Split	Internal Cross Validation Using Full Data

Root MSE	3905728
Dependent Mean	7503500
R-Square	0.4055
Adj R-Sq	0.4050
AIC	470167
AICC	470167
SBC	455277
ASE (Train)	1.524047E13
ASE (Test)	1.574238E13
CV PRESS	2.292747E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	13	1.558466E17	1.19882E16	785.87
Error	14980	2.285156E17	1.525471E13	
Corrected Total	14993	3.843622E17		

Root MSE	3919496
Dependent Mean	7483904
R-Square	0.4075
Adj R-Sq	0.4071
AIC	621818
AICC	621818
SBC	602100
CV PRESS	3.063805E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	13	2.093126E17	1.610097E16	1048.07
Error	19812	3.043609E17	1.536245E13	
Corrected Total	19825	5.136735E17		

OLS Backward Elimination Using OLS Backward Elimination Variables – Model 3

External Cross Validation Using Test/Train Split

Root MSE	3744159
Dependent Mean	7503500
R-Square	0.4554
Adj R-Sq	0.4532
AIC	468947
AICC	468948
SBC	454424
ASE (Train)	1.396076E13
ASE (Test)	1.531753E13
CV PRESS	5.208058E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	61	1.750347E17	2.86942E15	204.68
Error	14932	2.093276E17	1.401872E13	
Corrected Total	14993	3.843622E17		

Internal Cross Validation Using Full Data

Root MSE	3780340
Dependent Mean	7483904
R-Square	0.4501
Adj R-Sq	0.4484
AIC	620432
AICC	620433
SBC	601094
CV PRESS	5.184452E17

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	61	2.312267E17	3.790602E15	265.24
Error	19764	2.824468E17	1.429097E13	
Corrected Total	19825	5.136735E17		

Custom Model Using OLS Backward Elimination Selection, Correlation Matrix, & VIF – Custom

External Cross Validation Using Test/Train Split	Internal Cross Validation Using Full Data

Root MSE	3798015
Dependent Mean	7503500
R-Square	0.4399
Adj R-Sq	0.4373
AIC	469383
AICC	469383
SBC	454912
ASE (Train)	1.435854E13
ASE (Test)	1.490589E13
CV PRESS	2.307358E17

Analysis of Variance				
Source	DF	Sum of Squares	Mean Square	F Value
Model	68	1.690703E17	2.486328E15	172.36
Error	14925	2.152919E17	1.442492E13	
Corrected Total	14993	3.843622E17		

Root MSE	3810126
Dependent Mean	7483904
R-Square	0.4416
Adj R-Sq	0.4397
AIC	620750
AICC	620751
SBC	601467
CV PRESS	3.085475E17

Analysis of Variance				
Source	DF	Sum of Squares	Mean Square	F Value
Model	68	2.268599E17	3.336176E15	229.81
Error	19757	2.868136E17	1.451706E13	
Corrected Total	19825	5.136735E17		

Serial Correlation Verification

AR(0) Time Series Model

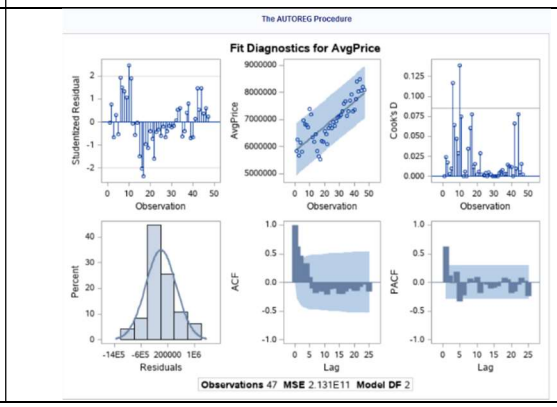
The AUTOREG Procedure

Ordinary Least Squares Estimates			
SSE	9.59068E12	DFE	45
MSE	2.13126E11	Root MSE	461656
SBC	1365.03878	AIC	1361.33848
MAE	351621.453	AICC	1361.61121
MAPE	5.22675722	HQC	1362.73093
Durbin-Watson	0.7524	Total R-Square	0.6558

Durbin-Watson Statistics			
Order	DW	Pr < DW	Pr > DW
1	0.7524	<.0001	1.0000

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates						
Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5824652	136857	42.56	<.0001	
MonthNumber	1	45969	4964	9.26	<.0001	MonthNumber



AR(1) Time Series Model

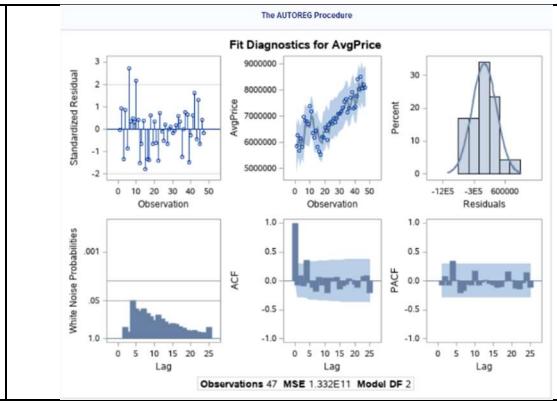
The AUTOREG Procedure

Yule-Walker Estimates			
SSE	5.86109E12	DFE	44
MSE	1.33206E11	Root MSE	364975
SBC	1346.23506	AIC	1340.68462
MAE	273459.117	AICC	1341.24276
MAPE	4.05914043	HQC	1342.77329
Durbin-Watson	2.1502	Transformed Regression R-Square	0.3568
		Total R-Square	0.7897

Durbin-Watson Statistics			
Order	DW	Pr < DW	Pr > DW
1	2.1502	0.6422	0.3578

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates						
Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5815270	263869	22.04	<.0001	
MonthNumber	1	46477	9408	4.94	<.0001	MonthNumber



AR(3) Time Series Model

AR(4) Time Series Model

The AUTOREG Procedure

Yule-Walker Estimates

SSE	8.58766E12	DFE	44
MSE	1.95174E11	Root MSE	441785
SBC	1364.01652	AIC	1358.46608
MAE	325594.201	AICC	1359.02422
MAPE	4.84375401	HQC	1360.55475
Durbin-Watson	1.0392	Transformed Regression R-Square	0.5470
		Total R-Square	0.6918

Durbin-Watson Statistics

Order	DW	Pr < DW	Pr > DW
1	1.0392	0.0001	0.9999

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5815387	178714	32.54	<.0001	
MonthNumber	1	46575	6390	7.29	<.0001	MonthNumber

The AUTOREG Procedure

Yule-Walker Estimates

SSE	8.58766E12	DFE	44
MSE	1.95174E11	Root MSE	441785
SBC	1364.01652	AIC	1358.46608
MAE	325594.201	AICC	1359.02422
MAPE	4.84375401	HQC	1360.55475
Durbin-Watson	1.0392	Transformed Regression R-Square	0.5470
		Total R-Square	0.6918

Durbin-Watson Statistics

Order	DW	Pr < DW	Pr > DW
1	1.0392	0.0001	0.9999

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5815387	178714	32.54	<.0001	
MonthNumber	1	46575	6390	7.29	<.0001	MonthNumber

AR(4) Time Series Model	AR(5) Time Series Model

The AUTOREG Procedure

Yule-Walker Estimates

SSE	8.40017E12	DFE	44
MSE	1.90913E11	Root MSE	436936
SBC	1363.14304	AIC	1357.5926
MAE	318949.314	AICC	1358.15074
MAPE	4.79463245	HQC	1359.68127
Durbin-Watson	0.7389	Transformed Regression R-Square	0.5635
		Total R-Square	0.6985

Durbin-Watson Statistics

Order	DW	Pr < DW	Pr > DW
1	0.7389	<.0001	1.0000

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5807384	176499	32.90	<.0001	
MonthNumber	1	47250	6269	7.54	<.0001	MonthNumber

The AUTOREG Procedure

Yule-Walker Estimates

SSE	9.50614E12	DFE	44
MSE	2.16049E11	Root MSE	464810
SBC	1368.51255	AIC	1362.96211
MAE	344670.439	AICC	1363.52025
MAPE	5.13820449	HQC	1365.05078
Durbin-Watson	0.7759	Transformed Regression R-Square	0.6333
		Total R-Square	0.6589

Durbin-Watson Statistics

Order	DW	Pr < DW	Pr > DW
1	0.7759	<.0001	1.0000

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	5817443	147434	39.46	<.0001	
MonthNumber	1	46384	5321	8.72	<.0001	MonthNumber

EDA R Code: Imputations & Transformations

```
library(pacman)
p_load(lmtest
       ,dplyr
       ,Hmisc
       ,skimr
       ,tidyr
       ,na.tools
       ,tidyverse
       ,olsrr
       ,caret
       ,multcomp
       ,ggthemes
       ,MASS# for OLS
       ,regclass# for VIF
       ,stats
       ,glmnet
       ,sjPlot
       ,sjmisc
       ,ggplot2
       ,xlsx)

#format dates:
df <- read.csv("./modelingData.csv",  header=T, sep=",", strip.white=T,
stringsAsFactors = F)

names(df)[36] <- "build_count_1921_1945"
```

```
names(df)[37] <- "build_count_1946_1970"
names(df)[38] <- "build_count_1971_1995"

########## Floor
#df$floor <- df$floor %>% replace_na(0)
df$floor[is.na(df$floor)] <- 0
df$floor <- log(df$floor+1)
##########
##########
#df[which(df$year == 2011),]
########## max floor
df2.maxfl <- df[which(!is.na(df$max_floor)),]
maxfl.Mean <- data.frame(df2.maxfl$max_floor/df2.maxfl$full_sq)
colnames(maxfl.Mean) <- "percentofFloor"
maxflMultiplier <- mean(head(maxfl.Mean$percentofFloor,7000))
df[which(is.na(df$max_floor)),6] <-
df[which(is.na(df$max_floor)),3]*maxflMultiplier
##########
#df[which(df$year == 2011),]
########## life sq
df2.lifesq <- df[which(!is.na(df$life_sq)),]
life.Mean <- data.frame(df2.lifesq$life_sq/df2.lifesq$full_sq)
colnames(life.Mean) <- "percentofFull"
lifesqMultiplier <- mean(head(life.Mean$percentofFull,11000))
df[which(is.na(df$life_sq)),4] <-
df[which(is.na(df$life_sq)),3]*lifesqMultiplier
df$life_sq <- as.numeric(df$life_sq)
dishonestLivingSpace <- (df$life_sq - df$full_sq)
df <- data.frame(dishonestLivingSpace, df)
df <- df[which(df$dishonestLivingSpace < 0),] #living space shouldn't be
greater than the full sq, considering lofts that have shared external
bathrooms
df <- subset(df, select = -c(dishonestLivingSpace)) # remove the counter
variable dishonestLivingSpace
##########
#df[which(df$year == 2011),]
########## kithcen sq
df2.kitchsq <- df[which(!is.na(df$kitch_sq)),]
kitch.Mean <- data.frame(df2.kitchsq$kitch_sq/df2.kitchsq$full_sq)
colnames(kitch.Mean) <- "percentofFullkitch"
kitchsqMultiplier <- mean(head(kitch.Mean$percentofFullkitch,7000))
df[which(is.na(df$kitch_sq)),10] <-
df[which(is.na(df$kitch_sq)),3]*kitchsqMultiplier
df$kitch_sq[is.na(df$kitch_sq)] <- 0
dishonestKitchens <- (df$kitch_sq - df$full_sq)
df <- data.frame(dishonestKitchens, df)
df <- df[which(df$dishonestKitchens < -2),] #kitchen space shouldn't be
greater than more than 2 meters less than the full sq
df <- subset(df, select = -c(dishonestKitchens)) # remove the counter
variable dishonestKitchens
df$kitch_sq <- sqrt(df$kitch_sq^1/16+1)
##########
#df[which(df$year == 2011),]
########## num room
df$num_room <- as.integer(df$num_room)
df2.numRm <- df[which(!is.na(df$num_room)),]
numRm.Mean <- as.numeric(df2.numRm$num_room)/as.numeric(df2.numRm$full_sq)
#colnames(numRm.Mean) <- "percentofFullrm"
class(numRm.Mean)
#numRm.Mean$percentofFullrm <- as.numeric(percentofFullrm)
numRmMultiplier <- mean(head(numRm.Mean,7000))
df[which(is.na(df$num_room)),9] <-
as.integer(df[which(is.na(df$num_room)),3])*numRmMultiplier
```

```
#############################################################################
######################df <- df[which(df$num_room < 30),]
####################

##########
#df[which(df$year == 2011),]
########## office raion
df$office_raion <- sqrt(df$office_raion^1/10)
##########
#df[which(df$year == 2011),]
########## big market raion
df$big_market_raion <- dplyr::recode(df$big_market_raion,  "no" = 0, "yes"=
1)
##########
########## Product_type
df$product_type <- dplyr::recode(df$product_type,  "Investment" = 0,
"OwnerOccupier"= 1)
##########

#df[which(df$year == 2011),]
########## railroad_terminal_raion
df$railroad_terminal_raion <- dplyr::recode(df$railroad_terminal_raion,  "no"
= 0, "yes"= 1)
##########
#df[which(df$year == 2011),]
##########
#df <- df %>% mutate(railroad_station_walk_min =
if_else(is.na(railroad_station_walk_min),0,railroad_station_walk_min))
df$railroad_station_walk_min[is.na(df$railroad_station_walk_min)] <- 0
##########
#df[which(df$year == 2011),]
##########
#df <- df %>% mutate(ID_railroad_station_walk =
if_else(is.na(ID_railroad_station_walk),0,ID_railroad_station_walk))
df$ID_railroad_station_walk[is.na(df$ID_railroad_station_walk)] <- 0
##########
#df[which(df$year == 2011),]
##########
#df$railroad_station_walk_km <- df$railroad_station_walk_km %>% replace_na(0)
df$railroad_station_walk_km[is.na(df$railroad_station_walk_km)] <- 0
##########
#df[which(df$year == 2011),]
##########
df <- df[which(!is.na(df$build_count_before_1920)),] #one fell swoop to take
out all NA 'build_count_{year range}' rows
##########
#df[which(df$year == 2011),]
##########
#df$metro_min_walk <- df$metro_min_walk %>% replace_na(0)
df$metro_min_walk[is.na(df$metro_min_walk)] <- 0
##########
#df[which(df$year == 2011),]
##########
#df$metro_km_walk <- df$metro_km_walk %>% replace_na(0)
df$metro_km_walk[is.na(df$metro_km_walk)] <- 0
##########
#df[which(df$year == 2011),]

################year_month <- as.factor(paste0(df$year,df$month))
################df <- data.frame(year_month,df)

############################## conversion to numeric and factor only for
modeling consistency ##############################
```

```
########## Material, Hospital bed raion
df <- subset(df, select = -c(material, hospital_beds_raion, build_year,
public_trans_station_time_walk, children_preschool, preschool_quota))

df <- df %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.character,
as.factor) %>% data.frame()

#df <- df[-c(2958, 1192, 2998,134, 3156, 1452, 2994, 3151, 98),]

write.csv(df,"cleanData.csv", row.names = F)

df <- read.csv("cleanData.csv", header = T)
################################################################################
############################ ignore the below
skim(df)
df2 <- df
df2$life_sq <- log(df$full_sq + 1)
df2$basketball_km <- log(df$basketball_km + 1)
df2$fitness_km <- log(df$fitness_km + 1)
df2$green_zone_km <- log(df$green_zone_km + 1)
df2$indust_part <- log(df$indust_part + 1)
df2$kitch_sq <- log(df$kitch_sq + 1)
df2$life_Sq <- log(df$life_Sq + 1)
df2$market_shop_km <- log(df$market_shop_km + 1)
df2$max_floor <- log(df$max_floor + 1)
df2$metro_km_avto <- log(df$metro_km_avto + 1)
df2$metro_km_walk <- log(df$metro_km_walk + 1)
df2$metro_min_avto <- log(df$metro_min_avto + 1)
df2$metro_min_walk <- log(df$metro_min_walk + 1)
df2$num_room <- log(df$num_room + 1)
df2$office_km <- log(df$office_km + 1)
df2$park_km <- log(df$park_km + 1)
df2$price_doc <- log(df$price_doc + 1)
df2$public_healthcare_km <- log(df$public_healthcare_km + 1)
df2$public_transport_station_km <- log(df$public_transport_station_km + 1)
df2$railroad_km <- log(df$railroad_km + 1)
df2$school_km <- log(df$school_km + 1)
df2$shopping_centers_km <- log(df$shopping_centers_km + 1)
skim(df2)
################################################################################
############################ ignore the above

lm.Model <- lm(log(price_doc) ~ id + life_sq + floor + max_floor + num_room +
kitch_sq + product_type + green_zone_part + indust_part + children_school +
healthcare_centers_raion +
                       university_top_20_raion + shopping_centers_raion +
railroad_terminal_raion + big_market_raion + X0_17_all + X16_29_all +
build_count_block + build_count_wood + build_count_frame +
                       build_count_brick + build_count_before_1920 +
build_count_1921_1945 + build_count_1946_1970 + build_count_1971_1995 +
build_count_after_1995 + metro_km_avto + school_km +
                       green_zone_km + industrial_km +
ID_railroad_station_walk + railroad_station_avto_km +
public_transport_station_km + kremlin_km + big_road1_km +
                       big_road2_km + railroad_km + bus_terminal_avto_km +
big_market_km + market_shop_km + fitness_km + swim_pool_km + ice_rink_km +
stadium_km + public_healthcare_km + university_km +
                       workplaces_km + shopping_centers_km + office_km +
big_church_km + X0_17_all*X16_29_all + children_school*school_km +
build_count_block*build_count_1921_1945 +
                       build_count_block*build_count_1946_1970 +
build_count_block*build_count_1971_1995 +
build_count_block*build_count_after_1995 +
build_count_wood*build_count_before_1920 +
```

```
                        build_count_wood*build_count_1946_1970 +
build_count_wood*build_count_after_1995 +
build_count_frame*build_count_before_1920 +
build_count_frame*build_count_1921_1945 +
                        build_count_frame*build_count_1946_1970 +
build_count_frame*build_count_after_1995 +
build_count_brick*build_count_1946_1970 +
build_count_brick*build_count_1971_1995 +
                        build_count_brick*build_count_after_1995 +
office_km*X16_29_all, data=df)

# viewing diagnostics residuals:
par(mfrow=c(2,2))
plot(lm.Model)

# Based on the suggested model above from SAS, we dropped outlier
observations:
df <- df[-c(12946, 16104,8678, 18323, 8924, 6425, 9728),]


# After updating to remove the outliers, we re-ran the model:
lm.Model <- lm(log(price_doc) ~ id + life_sq + floor + max_floor + num_room +
kitch_sq + product_type + green_zone_part + indust_part + children_school +
healthcare_centers_raion +
                        university_top_20_raion + shopping_centers_raion +
railroad_terminal_raion + big_market_raion + X0_17_all + X16_29_all +
build_count_block + build_count_wood + build_count_frame +
                        build_count_brick + build_count_before_1920 +
build_count_1921_1945 + build_count_1946_1970 + build_count_1971_1995 +
build_count_after_1995 + metro_km_avto + school_km +
                        green_zone_km + industrial_km +
ID_railroad_station_walk + railroad_station_avto_km +
public_transport_station_km + kremlin_km + big_road1_km +
                        big_road2_km + railroad_km + bus_terminal_avto_km +
big_market_km + market_shop_km + fitness_km + swim_pool_km + ice_rink_km +
stadium_km + public_healthcare_km + university_km +
                        workplaces_km + shopping_centers_km + office_km +
big_church_km + X0_17_all*X16_29_all + children_school*school_km +
build_count_block*build_count_1921_1945 +
                        build_count_block*build_count_1946_1970 +
build_count_block*build_count_1971_1995 +
build_count_block*build_count_after_1995 +
build_count_wood*build_count_before_1920 +
                        build_count_wood*build_count_1946_1970 +
build_count_wood*build_count_after_1995 +
build_count_frame*build_count_before_1920 +
build_count_frame*build_count_1921_1945 +
                        build_count_frame*build_count_1946_1970 +
build_count_frame*build_count_after_1995 +
build_count_brick*build_count_1946_1970 +
build_count_brick*build_count_1971_1995 +
                        build_count_brick*build_count_after_1995 +
office_km*X16_29_all, data=df)
# viewing post-outlier removal diagnostics residuals:
par(mfrow=c(2,2))
plot(lm.Model)
# As indicated by the residuals diagnostics, our model is better, with the
exception of issues with independence and non-constant
# variance from some of the imputations. This is also causing a left-skew in
the QQ plot.

#updating with removed variables
write.csv(df,"cleanData.csv", row.names = F)
```

Test Data Cleaning R Code: Preparation for External Cross-Validation

```
library(pacman)
p_load(lmtest
       ,dplyr
       ,Hmisc
       ,skimr
       ,tidyr
       ,na.tools
       ,tidyverse
       ,olsrr
       ,caret
       ,multcomp
       ,ggthemes
       ,MASS# for OLS
       ,regclass# for VIF
       ,stats
       ,glmnet
       ,sjPlot
       ,sjmisc
       ,ggplot2
       ,xlsx)

#format dates:
df <- Test

names(df)[35] <- "build_count_1921_1945"
names(df)[36] <- "build_count_1946_1970"
names(df)[37] <- "build_count_1971_1995"

###############df <- df %>% mutate(timestamp = as.Date(timestamp,
origin="1899-12-30"))
###############tryFormats = c("%Y-%m-%d", "%Y/%m/%d")

###############timestamp2 <- df$timestamp
###############df <- data.frame(timestamp2, df)

###############df <- df %>% mutate(timestamp = as.Date(timestamp,
origin="1899-12-30"))
###############tryFormats = c("%Y-%m-%d", "%Y/%m/%d")

###############df <- df %>% separate(timestamp2, sep="-", into = c("year",
"month", "day"))

#names(df$build_count_1921.1945)[36] <- "build_count_1921_1945"
#names(df$build_count_1946.1970)[37] <- "build_count_1946_1970"
#names(df$build_count_1971.1995)[38] <- "build_count_1971_1995"

#$colnames(df) <- c('year', 'month', 'day', 'id', 'timestamp', 'full_sq',
'life_sq', 'floor', 'max_floor', 'num_room', 'kitch_sq', 'product_type',
'raion_popul', 'green_zone_part', 'indust_part', 'children_school',
'healthcare_centers_raion', 'university_top_20_raion',
'shopping_centers_raion', 'office_raion', 'railroad_terminal_raion',
```

```
'big_market_raion', 'full_all', 'X0_6_all', 'X7_14_all', 'X0_17_all',
'X16_29_all', 'X0_13_all', 'build_count_block', 'build_count_wood',
'build_count_frame', 'build_count_brick', 'build_count_before_1920',
'build_count_1921_1945', 'build_count_1946_1970', 'build_count_1971_1995',
'build_count_after_1995', 'metro_min_avto', 'metro_km_avto',
'metro_min_walk', 'metro_km_walk', 'school_km', 'park_km', 'green_zone_km',
'industrial_km', 'railroad_station_walk_km', 'railroad_station_walk_min',
'ID_railroad_station_walk', 'railroad_station_avto_km',
'railroad_station_avto_min', 'public_transport_station_km', 'kremlin_km',
'big_road1_km', 'big_road2_km', 'railroad_km', 'bus_terminal_avto_km',
'big_market_km', 'market_shop_km', 'fitness_km', 'swim_pool_km',
'ice_rink_km', 'stadium_km', 'basketball_km', 'public_healthcare_km',
'university_km', 'workplaces_km', 'shopping_centers_km', 'office_km',
'big_church_km', 'price_doc')

########## Floor
#df$floor <- df$floor %>% replace_na(0)
df$floor[is.na(df$floor)] <- 0
df$floor <- log(df$floor+1)
##########

#df$product_type <- as.factor(df$product_type)

##########
#df[which(df$year == 2011),]
########## max floor
df2.maxfl <- df[which(!is.na(df$max_floor)),]
maxfl.Mean <- data.frame(df2.maxfl$max_floor/df2.maxfl$full_sq)
colnames(maxfl.Mean) <- "percentofFloor"
maxflMultiplier <- mean(head(maxfl.Mean$percentofFloor,7000))
df[which(is.na(df$max_floor)),7] <-
df[which(is.na(df$max_floor)),4]*maxflMultiplier
##########
#df[which(df$year == 2011),]
########## life sq
df2.lifesq <- df[which(!is.na(df$life_sq)),]
life.Mean <- data.frame(df2.lifesq$life_sq/df2.lifesq$full_sq)
colnames(life.Mean) <- "percentofFull"
lifesqMultiplier <- mean(head(life.Mean$percentofFull,11000))
df[which(is.na(df$life_sq)),5] <-
df[which(is.na(df$life_sq)),4]*lifesqMultiplier
df$life_sq <- as.numeric(df$life_sq)
dishonestLivingSpace <- (df$life_sq - df$full_sq)
df <- data.frame(dishonestLivingSpace, df)
df <- df[which(df$dishonestLivingSpace < 0),] #living space shouldn't be
greater than the full sq, considering lofts that have shared external
bathrooms
df <- subset(df, select = -c(dishonestLivingSpace)) # remove the counter
variable dishonestLivingSpace
##########
#df[which(df$year == 2011),]
########## kithcen sq
df2.kitchsq <- df[which(!is.na(df$kitch_sq)),]
kitch.Mean <- data.frame(df2.kitchsq$kitch_sq/df2.kitchsq$full_sq)
colnames(kitch.Mean) <- "percentofFullkitch"
kitchsqMultiplier <- mean(head(kitch.Mean$percentofFullkitch,7000))
```

```
df[which(is.na(df$kitch_sq)),9] <-
df[which(is.na(df$kitch_sq)),4]*kitchsqMultiplier
df$kitch_sq[is.na(df$kitch_sq)] <- 0
dishonestKitchens <- (df$kitch_sq - df$full_sq)
df <- data.frame(dishonestKitchens, df)
df <- df[which(df$dishonestKitchens < -2),] #kitchen space shouldn't be
greater than more than 2 meters less than the full sq
df <- subset(df, select = -c(dishonestKitchens)) # remove the counter
variable dishonestKitchens
df$kitch_sq <- sqrt(df$kitch_sq^1/16+1)
##########
#df[which(df$year == 2011),]
########## num room
df$num_room <- as.integer(df$num_room)
df2.numRm <- df[which(!is.na(df$num_room)),]
numRm.Mean <- as.numeric(df2.numRm$num_room)/as.numeric(df2.numRm$full_sq)
#colnames(numRm.Mean) <- "percentofFullrm"
class(numRm.Mean)
#numRm.Mean$percentofFullrm <- as.numeric(percentofFullrm)
numRmMultiplier <- mean(head(numRm.Mean,7000))
df[which(is.na(df$num_room)),8] <-
as.integer(df[which(is.na(df$num_room)),4])*numRmMultiplier
#############################################################################
######################df <- df[which(df$num_room < 30),]
##########
##########
#df[which(df$year == 2011),]
########## office raion
df$office_raion <- sqrt(df$office_raion^1/10)
##########
#df[which(df$year == 2011),]
########## big market raion
df$big_market_raion <- dplyr::recode(df$big_market_raion,  "no" = 0, "yes"=
1)
##########
########## Product_type
df$product_type <- dplyr::recode(df$product_type,  "Investment" = 0,
"OwnerOccupier"= 1)
##########

#df[which(df$year == 2011),]
########## railroad terminal raion
df$railroad_terminal_raion <- dplyr::recode(df$railroad_terminal_raion,  "no"
= 0, "yes"= 1)
##########
#df[which(df$year == 2011),]
##########
#df <- df %>% mutate(railroad_station_walk_min =
if_else(is.na(railroad_station_walk_min),0,railroad_station_walk_min))
df$railroad_station_walk_min[is.na(df$railroad_station_walk_min)] <- 0
##########
#df[which(df$year == 2011),]
##########
#df <- df %>% mutate(ID_railroad_station_walk =
if_else(is.na(ID_railroad_station_walk),0,ID_railroad_station_walk))
df$ID_railroad_station_walk[is.na(df$ID_railroad_station_walk)] <- 0
```

```
##########
#df[which(df$year == 2011),]
##########
#df$railroad_station_walk_km <- df$railroad_station_walk_km %>% replace_na(0)
df$railroad_station_walk_km[is.na(df$railroad_station_walk_km)] <- 0
##########
#df[which(df$year == 2011),]
##########
df <- df[which(!is.na(df$build_count_before_1920)),] #one fell swoop to take
out all NA 'build_count_{year range}' rows
##########
#df[which(df$year == 2011),]
##########
#df$metro_min_walk <- df$metro_min_walk %>% replace_na(0)
df$metro_min_walk[is.na(df$metro_min_walk)] <- 0
##########
#df[which(df$year == 2011),]
##########
#df$metro_km_walk <- df$metro_km_walk %>% replace_na(0)
df$metro_km_walk[is.na(df$metro_km_walk)] <- 0
##########
#df[which(df$year == 2011),]

################year_month <- as.factor(paste0(df$year,df$month))
################df <- data.frame(year_month,df)


############################## conversion to numeric and factor only for
modeling consistency ##############################

########## Material, Hospital bed raion
df <- subset(df, select = -c(material, hospital_beds_raion, build_year))

df <- df %>% mutate_if(is.integer, as.numeric) %>% mutate_if(is.character,
as.factor) %>% data.frame()

#df <- df[-c(2958, 1192, 2998,134, 3156, 1452, 2994, 3151, 98),]

write.csv(df,"cleanTestData.csv", row.names = F)
```

Goal 1 SAS Code: Modeling & Cross-Validation

```
FILENAME REFFILE 'C:/Users/Pablo/Desktop/KG6372/cleanData.xlsx';
PROC IMPORT DATAFILE=REFFILE
DBMS=XLSX
OUT= DF;
GETNAMES=YES;
run;
proc print data = DF(obs=10);
run;
/* backward elimination */
```

```
title "Backward Elimination";
proc glmselect data=DF
testdata=DF
seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot CRITERIONPANEL);
model PRICE_DOC = id timestamp full_sq life_sq floor max_floor num_room
kitch_sq product_type raion_popul green_zone_part indust_part children_school
healthcare_centers_raion university_top_20_raion shopping_centers_raion
office_raion railroad_terminal_raion big_market_raion full_all X0_17_all
X16_29_all X0_13_all build_count_block build_count_wood build_count_frame
build_count_brick build_count_before_1920 build_count_1946_1970
build_count_1971_1995 build_count_after_1995 metro_min_avto metro_km_avto
metro_min_walk school_km green_zone_km industrial_km railroad_station_walk_km
ID_railroad_station_walk railroad_station_avto_km railroad_station_avto_min
public_transport_station_km kremlin_km big_road1_km big_road2_km railroad_km
bus_terminal_avto_km big_market_km market_shop_km fitness_km swim_pool_km
ice_rink_km stadium_km basketball_km public_healthcare_km university_km
workplaces_km shopping_centers_km office_km big_church_km x16_29_all*x0_6_all
x16_29_all*x0_13_all x0_17_all*x0_13_all x0_17_all*x16_29_all
x7_14_all*x0_17_all x7_14_all*x0_17_all x7_14_all*x0_17_all
x16_29_all*x0_13_all x16_29_all*x0_17_all x16_29_all*x7_14_all
children_school*school_km build_count_block*build_count_before_1920
build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1921_1945 build_count_wood*build_count_1946_1970
build_count_wood*build_count_1971_1995
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_1971_1995
build_count_frame*build_count_after_1995
build_count_brick*build_count_before_1920
build_count_brick*build_count_1921_1945
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 fitness_km*X16_29_all
swim_pool_km*X0_6_all swim_pool_km*X7_14_all swim_pool_km*X0_17_all
swim_pool_km*X0_13_all swim_pool_km*X16_29_all ice_rink_km*X0_6_all
ice_rink_km*X7_14_all ice_rink_km*X0_17_all ice_rink_km*X0_13_all
ice_rink_km*X16_29_all stadium_km*X16_29_all office_km*X16_29_all
/ selection=backward( choose=CV stop=CV include = 107) CVdetails;
run;
quit;

title "Forward Selection";
proc glmselect data=DF
testdata=DF
seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot CRITERIONPANEL);
model PRICE_DOC = id life_sq floor max_floor kitch_sq indust_part kremlin_km
/ selection=forward( choose=CV stop=CV ) CVdetails;
run;
quit;
```

```
title "Stepwise Regression";
proc glmselect data=DF
testdata=DF
seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot CRITERIONPANEL);
model PRICE_DOC = id timestamp full_sq life_sq floor max_floor num_room
kitch_sq product_type raion_popul green_zone_part indust_part children_school
healthcare_centers_raion university_top_20_raion shopping_centers_raion
office_raion railroad_terminal_raion big_market_raion full_all X0_17_all
X16_29_all X0_13_all build_count_block build_count_wood build_count_frame
build_count_brick build_count_before_1920 build_count_1946_1970
build_count_1971_1995 build_count_after_1995 metro_min_avto metro_km_avto
metro_min_walk school_km green_zone_km industrial_km railroad_station_walk_km
ID_railroad_station_walk railroad_station_avto_km railroad_station_avto_min
public_transport_station_km kremlin_km big_road1_km big_road2_km railroad_km
bus_terminal_avto_km big_market_km market_shop_km fitness_km swim_pool_km
ice_rink_km stadium_km basketball_km public_healthcare_km university_km
workplaces_km shopping_centers_km office_km big_church_km x16_29_all*x0_6_all
x16_29_all*x0_13_all x0_17_all*x0_13_all x0_17_all*x16_29_all
x7_14_all*x0_17_all x7_14_all*x0_17_all x7_14_all*x0_17_all
x16_29_all*x0_13_all x16_29_all*x0_17_all x16_29_all*x7_14_all
children_school*school_km build_count_block*build_count_before_1920
build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1921_1945 build_count_wood*build_count_1946_1970
build_count_wood*build_count_1971_1995
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_1971_1995
build_count_frame*build_count_after_1995
build_count_brick*build_count_before_1920
build_count_brick*build_count_1921_1945
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 fitness_km*X16_29_all
swim_pool_km*X0_6_all swim_pool_km*X7_14_all swim_pool_km*X0_17_all
swim_pool_km*X0_13_all swim_pool_km*X16_29_all ice_rink_km*X0_6_all
ice_rink_km*X7_14_all ice_rink_km*X0_17_all ice_rink_km*X0_13_all
ice_rink_km*X16_29_all stadium_km*X16_29_all office_km*X16_29_all
/ selection=stepwise(choose=CV stop=CV include = 30) CVdetails;
run;
quit;
/* LASSO selection */
title "LASSO Selection";
/* R-squared is not a great measure for LASSO here */
proc glmselect data =DF
        seed=1 plots(stepAxis = number)=(criterionPanel ASEPlot
CRITERIONPANEL);
        model price_doc = id timestamp full_sq life_sq floor max_floor
num_room kitch_sq product_type raion_popul green_zone_part indust_part
children_school healthcare_centers_raion university_top_20_raion
shopping_centers_raion office_raion railroad_terminal_raion big_market_raion
```

```
full_all X0_6_all X7_14_all X0_17_all X16_29_all X0_13_all build_count_block
build_count_wood build_count_frame build_count_brick build_count_before_1920
build_count_1921_1945 build_count_1946_1970 build_count_1971_1995
build_count_after_1995 metro_min_avto metro_km_avto metro_min_walk
metro_km_walk school_km park_km green_zone_km industrial_km
railroad_station_walk_km railroad_station_walk_min ID_railroad_station_walk
railroad_station_avto_km railroad_station_avto_min
public_transport_station_km kremlin_km big_road1_km big_road2_km railroad_km
bus_terminal_avto_km big_market_km market_shop_km fitness_km swim_pool_km
ice_rink_km stadium_km basketball_km public_healthcare_km university_km
workplaces_km shopping_centers_km office_km big_church_km price_doc
/selection = LASSO(choose = CV stop=CV) CVdetails;
run;
quit;
/* The start of our custom model (below) uses interaction terms and all
variables suggested by our best OLS linear model (from backward) */
proc glmselect data=DF
testdata=DF
seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot CRITERIONPANEL);
model PRICE_DOC = id life_sq floor max_floor num_room kitch_sq product_type
green_zone_part indust_part children_school healthcare_centers_raion
university_top_20_raion shopping_centers_raion railroad_terminal_raion
big_market_raion X0_17_all X16_29_all build_count_block build_count_wood
build_count_frame build_count_brick build_count_before_1920
build_count_1921_1945 build_count_1946_1970 build_count_1971_1995
build_count_after_1995 metro_km_avto school_km green_zone_km industrial_km
ID_railroad_station_walk railroad_station_avto_km public_transport_station_km
kremlin_km big_road1_km big_road2_km railroad_km bus_terminal_avto_km
big_market_km market_shop_km fitness_km swim_pool_km ice_rink_km stadium_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km x16_29_all*x0_6_all x16_29_all*x0_13_all
X0_17_all*X0_13_all X0_17_all*X16_29_all X0_17_all*X7_14_all
X16_29_all*X7_14_all children_school*school_km
build_count_block*build_count_before_1920
build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1921_1945 build_count_wood*build_count_1946_1970
build_count_wood*build_count_1971_1995
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_1971_1995
build_count_frame*build_count_after_1995
build_count_brick*build_count_before_1920
build_count_brick*build_count_1921_1945
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 X16_29_all*fitness_km
swim_pool_km*X0_6_all swim_pool_km*X7_14_all X0_17_all*swim_pool_km
X0_13_all*swim_pool_km ice_rink_km*X0_6_all ice_rink_km*X7_14_all
X0_17_all*ice_rink_km X0_13_all*ice_rink_km office_km*X16_29_all
/ selection=backward( choose=CV stop=CV include = 40) CVdetails;
run;
```

```
quit;
/* first, a correlation matrix to check collinearity */
proc corr data = DF noprob output=OutCorr nomiss
cov;
var price_doc id timestamp full_sq life_sq floor max_floor num_room kitch_sq
product_type raion_popul green_zone_part indust_part children_school
healthcare_centers_raion university_top_20_raion shopping_centers_raion
office_raion railroad_terminal_raion big_market_raion full_all X0_17_all
X16_29_all X0_13_all build_count_block build_count_wood build_count_frame
build_count_brick build_count_before_1920 build_count_1921_1945
build_count_1946_1970 build_count_1971_1995 build_count_after_1995
metro_min_avto metro_km_avto school_km green_zone_km industrial_km
railroad_station_walk_km ID_railroad_station_walk railroad_station_avto_km
railroad_station_avto_min public_transport_station_km kremlin_km big_road1_km
big_road2_km railroad_km bus_terminal_avto_km big_market_km market_shop_km
fitness_km swim_pool_km ice_rink_km stadium_km basketball_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km;
run;

/* then, a variance importance factor matrix to also check collinearity. We
don't want anything that falls below a tolerance of 0.1*/
proc reg data=df;
model price_doc = id life_sq floor max_floor num_room kitch_sq product_type
green_zone_part indust_part children_school healthcare_centers_raion
university_top_20_raion shopping_centers_raion railroad_terminal_raion
big_market_raion X0_17_all X16_29_all build_count_block build_count_wood
build_count_frame build_count_brick build_count_before_1920
build_count_1921_1945 build_count_1946_1970 build_count_1971_1995
build_count_after_1995 metro_km_avto school_km green_zone_km industrial_km
ID_railroad_station_walk railroad_station_avto_km public_transport_station_km
kremlin_km big_road1_km big_road2_km railroad_km bus_terminal_avto_km
big_market_km market_shop_km fitness_km swim_pool_km ice_rink_km stadium_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km / vif tol collin;
run;

/* start interaction term visuals */
/************************************************* Plotting Interaction
Terms Below ************************************************/
ods graphics on;
proc glm data=DF;
   model price_doc = X0_17_all | X16_29_all / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=X0_17_all sliceby=X16_29_all) / clm;
run;
proc glm data=DF;
   model price_doc = children_school | school_km / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
```

```
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=children_school sliceby=school_km) / clm;
run;
proc glm data=DF;
   model price_doc = build_count_block | build_count_1921_1945 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_block sliceby=build_count_1921_1945) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_block | build_count_1946_1970 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_block sliceby=build_count_1946_1970) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_block | build_count_1971_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_block sliceby=build_count_1971_1995) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_block | build_count_after_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_block sliceby=build_count_after_1995) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_wood | build_count_before_1920 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_wood sliceby=build_count_before_1920) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_wood | build_count_1946_1970 / solution;
   ods select ParameterEstimates ContourFit;
```

```
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_wood sliceby=build_count_1946_1970) / clm;
run;
proc glm data=DF;
   model price_doc = build_count_wood | build_count_after_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_wood sliceby=build_count_after_1995) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_frame | build_count_before_1920 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_frame sliceby=build_count_before_1920) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_frame | build_count_1921_1945 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_frame sliceby=build_count_1921_1945) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_frame | build_count_1946_1970 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_frame sliceby=build_count_1946_1970) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_frame | build_count_after_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_frame sliceby=build_count_after_1995) /
clm;
run;
```

```
proc glm data=DF;
   model price_doc = build_count_brick | build_count_1946_1970 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_brick sliceby=build_count_1946_1970) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_brick | build_count_1971_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_brick sliceby=build_count_1971_1995) /
clm;
run;
proc glm data=DF;
   model price_doc = build_count_brick | build_count_after_1995 / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=build_count_brick sliceby=build_count_after_1995) /
clm;
run;
proc glm data=DF;
   model price_doc = office_km | X16_29_all / solution;
   ods select ParameterEstimates ContourFit;
   store GLMModel;
run;
/* step two (final step) for plotting interaction terms */
proc plm restore=GLMModel noinfo;
 effectplot slicefit(x=office_km sliceby=X16_29_all) / clm;
run;
ods graphics off;

/************************************************* Plotting Interaction
Terms Above *************************************************/

/* After updating using the suggestion from Backward on our interactions,
filtering using correlation matrix and dropping using VIF threshold,
we continued removing interaction terms based on interaction plots. The above
plots visualize the terms we left in the model below, which is
our best model. Although we reduced our adjusted R-squared by using the above
methods (corr matrix, VIF, interaction plots), we feel this
was because the models were overfit. Below, we are confident in the fit.*/
ods graphics on;
proc glm data = DF plots(unpack) = ALL;
proc glm data = DF plots(unpack) = (DIAGNOSTICS RESIDUALS);
model price_doc = id life_sq floor max_floor num_room kitch_sq product_type
green_zone_part indust_part children_school healthcare_centers_raion
```

```
university_top_20_raion shopping_centers_raion railroad_terminal_raion
big_market_raion X0_17_all X16_29_all build_count_block build_count_wood
build_count_frame build_count_brick build_count_before_1920
build_count_1921_1945 build_count_1946_1970 build_count_1971_1995
build_count_after_1995 metro_km_avto school_km green_zone_km industrial_km
ID_railroad_station_walk railroad_station_avto_km public_transport_station_km
kremlin_km big_road1_km big_road2_km railroad_km bus_terminal_avto_km
big_market_km market_shop_km fitness_km swim_pool_km ice_rink_km stadium_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km X0_17_all*X16_29_all children_school*school_km
build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1946_1970
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_after_1995
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 office_km*X16_29_all / cli;
run;

/*************************************************************************
*************************************************************/
/*************************************************************************
*************************************************************/
/********************************************** After updating model using
OLS ************************************************************/
/********************************************** correlation matrix
*************************************************************/
/*****************************************************and VIF model
*************************************************************/
/*************************************************************************
*************************************************************/
/*************************************************************************
*************************************************************/
%let inputData = DF;
%let numObs = 19836; *number of our cleanData observations (19,835) + 1;
%let numVarsLasso = 13;
%let lassoVars = id life_sq floor kitch_sq indust_part office_raion
green_zone_km ID_railroad_station_walk kremlin_km fitness_km swim_pool_km
university_km office_km;
*below are the linear variables selected outright by our best performing
Ordinary Least Squares method (Backward Elimination);
%let numVarsOLS = 62;
%let OLSVars = id full_sq life_sq floor max_floor num_room kitch_sq
product_type raion_popul green_zone_part indust_part children_school
healthcare_centers_raion university_top_20_raion shopping_centers_raion
office_raion railroad_terminal_raion big_market_raion full_all X0_17_all
X16_29_all X0_13_all build_count_block build_count_wood build_count_frame
build_count_brick build_count_before_1920 build_count_1921_1945
build_count_1946_1970 build_count_1971_1995 build_count_after_1995
metro_min_avto metro_km_avto school_km green_zone_km industrial_km
```

```
railroad_station_walk_km ID_railroad_station_walk railroad_station_avto_km
railroad_station_avto_min public_transport_station_km kremlin_km big_road1_km
big_road2_km railroad_km bus_terminal_avto_km big_market_km market_shop_km
fitness_km swim_pool_km ice_rink_km stadium_km basketball_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km;
%let numVarsOLSFwd = 7; * we may want to use these in the future to compare,
but for now, we just need four models (same for OLSVarsFwd below);
%let OLSVarsFwd = id life_sq floor max_floor kitch_sq indust_part kremlin_km;
```

```
/* below we are setting the variables for the SQL we will eventually use for
our custom model */
%let numVarsCustom = 69;
%let customOLSVars = id life_sq floor max_floor num_room kitch_sq
product_type green_zone_part indust_part children_school
healthcare_centers_raion university_top_20_raion shopping_centers_raion
railroad_terminal_raion big_market_raion X0_17_all X16_29_all
build_count_block build_count_wood build_count_frame build_count_brick
build_count_before_1920 build_count_1921_1945 build_count_1946_1970
build_count_1971_1995 build_count_after_1995 metro_km_avto school_km
green_zone_km industrial_km ID_railroad_station_walk railroad_station_avto_km
public_transport_station_km kremlin_km big_road1_km big_road2_km railroad_km
bus_terminal_avto_km big_market_km market_shop_km fitness_km swim_pool_km
ice_rink_km stadium_km public_healthcare_km university_km workplaces_km
shopping_centers_km office_km big_church_km X0_17_all*X16_29_all
children_school*school_km build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1946_1970
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_after_1995
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 office_km*X16_29_all;
/* dependent variable is price_doc */
%let depVar = price_doc;
```

```
     /*************************************************************
    ****************************************/
    /*************************************************************
    ****************************************/
                                  /********* External Cross-
    Validation Steps Here *********/
data DF;
set &inputData;
randNumber = ranuni(11);
```

```
if _n_ < &numObs;
run;
/* build training data set for external cross-validation. We will train in
25% blocks, test on 75%. We feel this is reasonable.
   75% for training */
data dfTrain;
set &inputData;
if randNumber <= 1/4 then delete;
run;
/* build our test data set for external cross-validation; 25% for testing */
data dfTest;
set &inputData;
if randNumber > 1/4 then delete;
run;

ods graphics on;
title "Selection Method LASSO Using LASSO Variables and Cross Validation";
proc glmselect data=dfTrain testdata = dfTest
                     seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &lassoVars
               / selection=LASSO( choose=CV stop=CV ) CVdetails;
               score data=dfTest out=scoredLASSO;
run;
quit;
ods graphics off;

ods graphics on;
title "Selection Method Backward Elimination Using LASSO Variables and OLS";
proc glmselect data=dfTrain testdata = dfTest
                     plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &lassoVars
               / selection=backward( choose=CV stop=CV include =
&numVarsLASSO ) CVdetails;
               score data=dfTest out=scoredOLSLASSO;
run;
quit;
ods graphics off;

ods graphics on;
title "Selection Method Backward Elimination Using OLS Variables and OLS";
proc glmselect data=dfTrain testdata = dfTest
                     plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &OLSVars
               / selection=backward( choose=CV stop=CV include = &numVarsOLS
) CVdetails; /*default to 5 folds*/
               score data=dfTest out=scoredOLS;
run;
quit;
ods graphics off;

ods graphics on;/* This sets up the SQL for our custom model cross validation
*/
title "Selection Method Custom Combined Backward Elimination, Corr Matrix,
VIF Using OLS Variables";
```

```
proc glmselect data=dfTrain testdata = dfTest
                    plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &customOLSVars
               / selection=backward( choose=CV stop=CV include =
&numVarsCustom ) CVdetails;
               score data=dfTest out=scoredOLSvarsCustom;
run;
quit;
ods graphics off;
/*****************************************************************************
*****************************/
/*****************************************************************************
*****************************/
/******************************** End of External Cross-Validation
*************************************/
/*****************************************************************************
*****************************/
/*****************************************************************************
*****************************/

                                  /********* Internal Cross-Validation Steps
Here *********/
ods graphics on;
title "Selection Method LASSO Using LASSO Variables and Cross Validation";
proc glmselect data=df
                    seed=1 plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &lassoVars
               / selection=LASSO( choose=CV stop=CV ) CVdetails;
               score data=df out=scoredLASSO;
run;
quit;
ods graphics off;

ods graphics on;
title "Selection Method Backward Elimination Using LASSO Variables and OLS";
proc glmselect data=df
                    plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &lassoVars
               / selection=backward( choose=CV stop=CV include =
&numVarsLASSO ) CVdetails;
               score data=df out=scoredOLSLASSO;
run;
quit;
ods graphics off;

ods graphics on;
title "Selection Method Backward Elimination Using OLS Variables and OLS";
proc glmselect data=df
                    plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &OLSVars
               / selection=backward( choose=CV stop=CV include = &numVarsOLS
) CVdetails; /*default to 5 folds*/
               score data=df out=scoredOLS;
```

```
run;
quit;
ods graphics off;

ods graphics on;/* This sets up the SQL for our custom model cross validation
*/
title "Selection Method Custom Combined Backward Elimination, Corr Matrix,
VIF Using OLS Variables";
proc glmselect data=df
                      plots(stepAxis=number)=(criterionPanel ASEPlot
CRITERIONPANEL);
model &depVar = &customOLSVars
               / selection=backward( choose=CV stop=CV include =
&numVarsCustom ) CVdetails;
               score data=df out=scoredOLSvarsCustom;
run;
quit;
ods graphics off;
/*****************************************************************************
*****************************/
/*****************************************************************************
*****************************/
/******************************** End of Internal Cross-Validation
*************************************/
/*****************************************************************************
*****************************/
/*****************************************************************************
*****************************/


/* Calculate Sums of Squares from LASSO and OLS outputs */
proc sql;
 create table fitLasso as
   select
      count(&depVar) as n
      ,css(&depVar) as totSS
      ,sum((&depVar - p_&depvar)**2) as errSSLasso
   from scoredLasso;
  create table fitOLSLasso as
    select sum((&depVar - p_&depvar)**2) as errSSOLSLasso /*THIS IS THE SAME
AS BELOW */
      from scoredOLSLasso;
  create table fitOLS as
    select sum((&depVar - p_&depvar)**2) as errSSOLS /* THIS IS THE SAME AS
ABOVE */
      from scoredOLS;
  create table customModel as /* this is for our custom model, which will be
our Regression Model */
    select sum((&depVar - p_&depvar)**2) as errSScustomOLS
      from scoredOLSvarsCustom;
quit;
run;

/* Calculate rsq and adjRsq using sums of squares from LASSO and OLS outputs
*/
data allMeasures;
merge fitLasso fitOLSLasso fitOLS customModel;
```

```
  rsqLasso = (1 - errSSLasso / totSS);
  rsqOLSLasso = (1 - errSSOLSLasso / totSS);
  rsqOLS = (1 - errSSOLS / totSS);
  rsqCustomOLS = (1 - errSScustomOLS / totSS); /* this is for our custom
model */
  adjRsqLasso = (1 - errSSLasso / totSS)*((n - 1) / (n - &numVarsLasso - 1));
  adjRsqOLSLasso = (1 - errSSOLSLasso / totSS)*((n - 1) / (n - &numVarsLasso
- 1));
  adjRsqOLS = (1 - errSSOLS / totSS)*((n - 1) / (n - &numVarsOLS - 1));
  adjRsqCustomOLS = (1 - errSScustomOLS / totSS)*((n - 1) / (n -
&numVarsCustom - 1)); /* this is for our custom model */
  run;

  title "Goodness of Fit Measures Using Test Data - All Models (LASSO with
LASSO Vars, OLS with LASSO Vars, OLS, and Custom OLS)";
  proc print data = allMeasures;
  run;
```

```
/*Prediction*********************************************************************
************************************************************/
FILENAME REFFILE '/home/u38493344/Applied Stats/Project
1/cleanTestData2.csv';
PROC IMPORT DATAFILE=REFFILE
        DBMS=csv
        OUT=cleanTest2;
        getnames = yes;
RUN;
proc print data = cleanTest2(obs=10);
run;

data forPred1;
set DF cleanTest2;
run;

proc print data = forPred1(obs=10);
run;

/*********************************************************************************
**/
/*********************************************************************************
**/
/*Best Model - Backwards with Interactions*/
proc glm data = forPred1 plots = ALL;
class PRODUCT_TYPE;
model price_doc = id life_sq floor max_floor num_room kitch_sq product_type
green_zone_part indust_part children_school healthcare_centers_raion
university_top_20_raion shopping_centers_raion railroad_terminal_raion
big_market_raion X0_17_all X16_29_all build_count_block build_count_wood
build_count_frame build_count_brick build_count_before_1920
build_count_1921_1945 build_count_1946_1970 build_count_1971_1995
build_count_after_1995 metro_km_avto school_km green_zone_km industrial_km
ID_railroad_station_walk railroad_station_avto_km public_transport_station_km
kremlin_km big_road1_km big_road2_km railroad_km bus_terminal_avto_km
big_market_km market_shop_km fitness_km swim_pool_km ice_rink_km stadium_km
public_healthcare_km university_km workplaces_km shopping_centers_km
office_km big_church_km X0_17_all*X16_29_all children_school*school_km
```

```
build_count_block*build_count_1921_1945
build_count_block*build_count_1946_1970
build_count_block*build_count_1971_1995
build_count_block*build_count_after_1995
build_count_wood*build_count_before_1920
build_count_wood*build_count_1946_1970
build_count_wood*build_count_after_1995
build_count_frame*build_count_before_1920
build_count_frame*build_count_1921_1945
build_count_frame*build_count_1946_1970
build_count_frame*build_count_after_1995
build_count_brick*build_count_1946_1970
build_count_brick*build_count_1971_1995
build_count_brick*build_count_after_1995 office_km*X16_29_all;
output out = predsMLR p = prediction lcl = lower ucl = upper;
run;
proc print data = predsMLR(obs=10);
run;

proc export data = predsMLR
dbms = csv outfile="/home/u38493344/Applied Stats/Project
1/mlrPredictions.csv" replace;
run;
```

Goal 2 SAS Code: Time-Series Modeling & Predictions

```
FILENAME REFFILE '/home/u38493344/Applied Stats/Project 1/timeSeriesB.xlsx';

PROC IMPORT DATAFILE=REFFILE
        DBMS=xlsx
        OUT=ts;
        getnames = yes;
RUN;
proc print data = ts;
run;


/*Goal2 - 4a.*/
proc reg data=ts;
model AvgPrice = MonthNumber;
output out = residualData residual= residual;
run;


/*Goal2 - 4b.*/
proc sgplot data = residualData;
scatter x = MonthNumber y = residual;
series x = MonthNumber y = residual;
xaxis label= "Month";
run;

/*Goal2 - 4c.(No Lag)*/
```

```
proc autoreg data = ts;
model AvgPrice = monthNumber/dwprob;
run;

/*************************************************************************
**/
/*************************************************************************
**/
/*Goal2 - 4c.(AR(1))*/
proc autoreg data = ts plots = residual;
model AvgPrice = MonthNumber/nlag=(1) dwprob;
output out = residData residual= residual;
run;

proc sgplot data = residData;
scatter x = MonthNumber y = residual;
series x = MonthNumber y = residual;
xaxis label= "Month";
run;
/*************************************************************************
**/
/*************************************************************************
**/

/*Goal2 - 4d. & 5*/
data predict;
input MonthNumber monthYear @@;
cards;
48 201507
49 201508
50 201509
51 201510
52 201511
53 201512
54 201601
55 201602
56 201603
57 201604
58 201605
59 201606
;
run;

data forPred;
set ts predict;
run;

proc autoreg data = forPred plots(unpack);
model AvgPrice = MonthNumber/nlag=(1) dwprob;
output out = preds p = prediction lcl = lower ucl = upper pm = trend
residual=resid;
run;
proc print data = preds; run;

proc sgplot data = preds;
band x = MonthNumber upper = upper lower = lower;
```

```
scatter x = MonthNumber y = prediction;
series x = MonthNumber y = prediction;
series x = MonthNumber y = trend/lineattrs = (color=black);
xaxis label= "Month";
run;

/*Question 6*/
proc export data = preds
dbms = csv outfile="/home/u38493344/Applied Stats/Project
1/tsPredictions.csv" replace;
run;
```