House Prices: Advanced Regression Techniques

EE 660 Project Type: Individual

Fangwei Lin, fangweil@usc.edu
4/12/2018

1. Abstract

My project is about "house prices prediction" and the dataset contains 79 features describing nearly every aspect of residential homes in Ames, Iowa and the label is "SalesPrice". This project is according to these 79 variables to train a model to predict house prices. The approaches and techniques used in this project including:

(1) EDA with Pandas and Seaborn; Find features with strong correlation to target;
(2) Data Wrangling, convert categorical to numerical;
(3) Apply the basic Regression models of sklearn;
(4) "GridsearchCV" to find the best parameters for each model;
(5) Compare the performance of the Regressors and choose best one.

I used "train_test_split" from "sklearn" to extract 30% data randomly from training data which has been cleaned as test data. In training regression models process, many advanced regression techniques are applied like Linear Regression, Ridge, Lasso, Decision Tree Regressor, Random Forest Regressor. The best model is Random Forest Regressor after compared all these models.

2. Introduction

2.1.

Problem Type, Statement and Goals

Type of this problem: Regression
Variable to predict: SalesPrice

Difficulty (nontriviality) and goals to achieve include:

(1) Missing values: In this project, it has two kinds of missing value. After reading data description, we know that in some features that “NaN” does not mean missing data. Like for “PoolQC”, it means that house has no pool. So, I replaced “NaN” with “None” in these columns. After that, I filled the remaining values with mean.
(2) Not normally distributed: This can reduce the performance of the ML models because they assume to be normal distribution. The target variable “SalePrice” is not normally distributed and I made log transform to it. And for not normally distributed features, we do not need to make log transform for all of them. This one is a little tricky, because after I log transform all of them, some features are still highly skewed. Besides not for all of them the correlation coefficient (to “SalePrice”) increases after log transform. So, I only pick “GrLivArea” which did increase. Surely, I surely did not check all possible settings here, since the performance also depends on the models and their hyperparameters.
(3) Multicollinearity of features: Some features are correlated strongly to each other. I plotted heat map to find these strongly correlated features and dropped one of them which is much weaker correlated to target.
(4) Find features with strong correlation to target: For numerical: Calculated a correlation matrix and plotted a heatmap Chose the most correlated features (correlation coefficient>0.4) and dropped those features below this threshold. For categorical: I plotted box plot to find strongly correlated features. I just look if the “SalePrice” (mean value and distribution) is significantly different for the different categories of each feature. More different means stronger correlated.
(5) Convert categorical to numerical: Each feature has its different categories. I made violin plots for these categorical features. And look at the mean of “SalePrice” as function of categories of each feature. Then I group by these categories in each feature according to their relation to target and labeled their as “1”,”2”,”3”,”4”.
(6) Hyperparameters tuning: Hyperparameters is important for training a good model. For “Random Forest Regressor”, I used “Random Hyperparameter Grid” to select best Hyperparameters in a much wider range. Then do grid search for the nearby of those hyperparameter values.
(7) Select best model

2.2.

2.3. Prior and Related Work (Mandatory)

Prior and Related Work - None

2.4. Overview of Approach

Overview of the models and algorithms:

(1) Linear Regression
(2) Ridge
(3) Lasso
(4) Decision Tree Regressor
(5) Random Forest Regressor
(6) RandomizedSearchCV
(7) GridresearchCV
(8) StandardScaler

And I chose "Root Mean Square Error" (RMSE) as performance metric. The lower value the better model.

3. Implementation

The following models are all imported from "sklearn". And for all of models I used "fit()" and "score()" methods and chose score as "neg_mean_squared_error". "fit(X,y)" fits linear model.

(1) Linear Regression:
(2) Ridge
(3) Lasso
(4) Decision Tree Regressor
(5) Random Forest Regressor

"RandomizedSearchCV" and "GridSearchCV" are imported from "sklearn". For both of them I used "best_estimator_" and "best_params_" methods. "best_estimator_" returns the estimator that was chosen by the search. "best_params_": returns parameter setting that gave the best results on the hold out data.
"StandardScaler" is also imported from "sklearn". I used "fit_transform" on training data to standardize features.
For one-hot encoding, I imported "get_dummies" from "pandas", for more details, I stated in the next part "3.1. Data Set".

3.1. Data Set

The dataset has 79 features which includes 36 numerical features and 43 categorical features, and one output label is "SalePrice". And it has 1460 data points and I chose 30% as testing data which is 438 data points.

Note: The following cells in shade means this is the feature used for final regression models training. The others are dropped.

79 features:

Feature's name	Meaning
MSSubClass	Identifies the type of dwelling involved in the sale.
LotFrontage	Linear feet of street connected to property.
LotArea	Lot size in square feet.
OverallQual	Rates the overall material and finish of the house
OverallCond	Rates the overall condition of the house
YearBuilt	Original construction date
YearRemodAdd	Remodel date (same as construction date if no remodeling or additions)
MasVnrArea	Masonry veneer area in square feet
BsmtFinSF1	Type 1 finished square feet
BsmtFinSF2	Type 2 finished square feet
BsmtUnfSF	Unfinished square feet of basement area
TotalBsmtSF	Total square feet of basement area
1stFlrSF	First Floor square feet
2ndFlrSF	Second floor square feet
LowQualFinSF	Low quality finished square feet (all floors)
GrLivArea	Above grade (ground) living area square feet
BsmtFullBath	Basement full bathrooms
BsmtHalfBath	Basement half bathrooms
FullBath	Full bathrooms above grade
HalfBath	Half baths above grade
BedroomAbvGr	Bedrooms above grade (does NOT include basement bedrooms)
KitchenAbvGr	Kitchens above grade
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms)
Fireplaces	Number of fireplaces
GarageYrBlt	Year garage was built
GarageCars	Size of garage in car capacity
GarageArea	Size of garage in square feet
WoodDeckSF	Wood deck area in square feet
OpenPorchSF	Open porch area in square feet
EnclosedPorch	Enclosed porch area in square feet
3SsnPorch	Three season porch area in square feet
ScreenPorch	Screen porch area in square feet
PoolArea	Pool area in square feet
MiscVal	$Value of miscellaneous feature
MoSold	Month Sold (MM)
YrSold	Year Sold (YYYY)
MSZoning	Identifies the general zoning classification of the sale.
Street	Type of road access to property
Alley	Type of alley access to property

LotShape	General shape of property
LandContour	Flatness of the property
Utilities	Type of utilities available
LotConfig	Lot configuration
LandSlope	Slope of property
Neighborhood	Physical locations within Ames city limits
Condition1	Proximity to various conditions
Condition2	Proximity to various conditions (if more than one is present)
BldgType	Type of dwelling
HouseStyle	Style of dwelling
RoofStyle	Type of roof
RoofMatl	Roof material
Exterior1st	Exterior covering on house
Exterior2nd	Exterior covering on house (if more than one material)
MasVnrType	Masonry veneer type
ExterQual	Evaluates the quality of the material on the exterior
ExterCond	Evaluates the present condition of the material on the exterior
Foundation	Type of foundation
BsmtQual	Evaluates the height of the basement
BsmtCond	Evaluates the general condition of the basement
BsmtExposure	Refers to walkout or garden level walls
BsmtFinType1	Rating of basement finished area
BsmtFinType2	Rating of basement finished area (if multiple types)
Heating	Type of heating
HeatingQC	Heating quality and condition
CentralAir	Central air conditioning
Electrical	Electrical system
KitchenQual	Kitchen quality
Functional	Home functionality (Assume typical unless deductions are warranted)
FireplaceQu	Fireplace quality
GarageType	Garage location
GarageFinish	Interior finish of the garage
GarageQual	Garage quality
GarageCond	Garage condition
PavedDrive	Paved driveway
PoolQC	Pool quality
Fence	Fence quality
MiscFeature	Miscellaneous feature not covered in other categories
SaleType	Type of sale
SaleCondition	Condition of sale

36 input variables(numerical):

Feature's name	Type
MSSubClass	Integer
LotFrontage	Float
LotArea	Integer
OverallQual	Integer
OverallCond	Integer
YearBuilt	Integer
YearRemodAdd	Integer
MasVnrArea	Float
BsmtFinSF1	Integer
BsmtFinSF2	Integer
BsmtUnfSF	Integer
TotalBsmtSF	Integer
1stFlrSF	Integer
2ndFlrSF	Integer
LowQualFinSF	Integer
GrLivArea	Integer
BsmtFullBath	Integer
BsmtHalfBath	Integer
FullBath	Integer
HalfBath	Integer
BedroomAbvGr	Integer
KitchenAbvGr	Integer
TotRmsAbvGrd	Integer
Fireplaces	Integer
GarageYrBlt	Float
GarageCars	Integer
GarageArea	Integer
WoodDeckSF	Integer
OpenPorchSF	Integer
EnclosedPorch	Integer
3SsnPorch	Integer
ScreenPorch	Integer
PoolArea	Integer
MiscVal	Integer
MoSold	Integer
YrSold	Integer

36 input variables(categorical):

Note: “Group and One-hot coding”: for those remaining categorical features (in shade) used for regression models training, I group those categories has similar relation to target output and encode them as 1,2,3,4. And for features 'NbHd_num', 'MSZ_num’, using one-hot encoding

Feature's name	Range	Group and encoding
MSZoning	A Agriculture C Commercial FV Floating Village Residential I Industrial RH Residential High Density RL Residential Low Density RP Residential Low Density Park RM Residential Medium Density	[‘A’, ’C’, ’I’] = 1 ['RM', 'RH'] = 2 ['RL', 'FV'] = 3
Street	Grvl Gravel Pave Paved	
Alley	Grvl Gravel Pave Paved NA No alley access	
LotShape	Reg Regular IR1 Slightly irregular IR2 Moderately Irregular IR3 Irregular	
LandContour	Lvl Near Flat/Level	

	Bnk Banked - Quick and significant rise from street grade to building HLS Hillside - Significant slope from side to side Low Depression	
Utilities	AllPub All public Utilities (E,G,W,& S) NoSewr Electricity, Gas, and Water (Septic Tank) NoSeWa Electricity and Gas Only ELO Electricity only	
LotConfig	Inside Inside lot Corner Corner lot CulDSac Cul-de-sac FR2 Frontage on 2 sides of property FR3 Frontage on 3 sides of property	
LandSlope	Gtl Gentle slope Mod Moderate Slope Sev Severe Slope	
Neighborhood	Blmngtn Bloomington Heights Blueste Bluestem BrDale Briardale BrkSide Brookside ClearCr Clear Creek CollgCr College Creek	[‘Blueste’,’ BrDale’, ’BrkSide’,’ Crawfor’, ’Edwards’,’ Gilbert’,’ IDOTRR’,’ MeadowV’, ’Mitchel’,’ Names’, ’NPkVill’,’ OldTown’, ’SWISU’,’ Sawyer’, ’SawyerW’] = 1 ['Blmngtn', 'ClearCr', 'CollgCr', 'Crawfor', 'Gilbert', 'NWAmes',

	Crawfor Crawford Edwards Edwards Gilbert Gilbert IDOTRR Iowa DOT and Rail Road MeadowV Meadow Village Mitchel Mitchell Names North Ames NoRidge Northridge NPkVill Northpark Villa NridgHt Northridge Heights NWAmes Northwest Ames OldTown Old Town SWISU South & West of Iowa State University Sawyer Sawyer SawyerW Sawyer West Somerst Somerset StoneBr Stone Brook Timber Timberland Veenker Veenker	'Somerst', 'Timber', 'Veenker'] = 2 ['NoRidge', 'NridgHt', 'StoneBr'] = 3
Condition1	Artery Adjacent to arterial street Feedr Adjacent to feeder street Norm Normal	

	RRNn Within 200' of North-South Railroad RRAn Adjacent to North-South Railroad PosN Near positive off-site feature--park, greenbelt, etc. PosA Adjacent to postive off-site feature RRNe Within 200' of East-West Railroad RRAe Adjacent to East-West Railroad	
Condition2	Artery Adjacent to arterial street Feedr Adjacent to feeder street Norm Normal RRNn Within 200' of North-South Railroad RRAn Adjacent to North-South Railroad PosN Near positive off-site feature--park, greenbelt, etc. PosA Adjacent to postive off-site feature RRNe Within 200' of East-West Railroad RRAe Adjacent to East-West Railroad	[‘Artery’,’ Feedr’,’ RRNn’,’ RRAn’,’ RRNe’] = 1 ['Norm', 'RRAe'] = 2 ['PosA', 'PosN'] = 3
BldgType	1Fam Single-family Detached	

	2FmCon Two-family Conversion; originally built as one-family dwelling Duplx Duplex TwnhsE Townhouse End Unit TwnhsI Townhouse Inside Unit	
HouseStyle	1Story One story 1.5Fin One and one-half story: 2nd level finished 1.5Unf One and one-half story: 2nd level unfinished 2Story Two story 2.5Fin Two and one-half story: 2nd level finished 2.5Unf Two and one-half story: 2nd level unfinished SFoyer Split Foyer SLvl Split Level	
RoofStyle	Flat Flat Gable Gable Gambrel Gabrel (Barn) Hip Hip Mansard Mansard Shed Shed	
RoofMatl	ClyTile Clay or Tile	

	CompShg	Standard (Composite) Shingle	
	Membran	Membrane	
	Metal	Metal	
	Roll	Roll	
	Tar&Grv	Gravel & Tar	
	WdShake	Wood Shakes	
	WdShngl	Wood Shingles	
Exterior1st	AsbShng	Asbestos Shingles	
	AsphShn	Asphalt Shingles	
	BrkComm	Brick Common	
	BrkFace	Brick Face	
	CBlock	Cinder Block	
	CemntBd	Cement Board	
	HdBoard	Hard Board	
	ImStucc	Imitation Stucco	
	MetalSd	Metal Siding	
	Other	Other	
	Plywood	Plywood	
	PreCast	PreCast	
	Stone	Stone	
	Stucco	Stucco	
	VinylSd	Vinyl Siding	
	Wd Sdng	Wood Siding	

	WdShing Wood Shingles	
Exterior2nd	AsbShng Asbestos Shingles AsphShn Asphalt Shingles BrkComm Brick Common BrkFace Brick Face CBlock Cinder Block CemntBd Cement Board HdBoard Hard Board ImStucc Imitation Stucco MetalSd Metal Siding Other Other Plywood Plywood PreCast PreCast Stone Stone Stucco Stucco VinylSd Vinyl Siding Wd Sdng Wood Siding WdShing Wood Shingles	
MasVnrType	BrkCmn Brick Common BrkFace Brick Face CBlock Cinder Block None None Stone Stone	[‘BrkCmn’,’ BrkFace’,’ CBlock’,’ None’] = 1 [‘Stone’] = 2

ExterQual	Ex Excellent Gd Good TA Average/Typical Fa Fair Po Poor	[‘Fa’, ‘Po’] = 1 [‘TA’] = 2 [‘Gd’] = 3 [‘Ex’] = 4
ExterCond	Ex Excellent Gd Good TA Average/Typical Fa Fair Po Poor	
Foundation	BrkTil Brick & Tile CBlock Cinder Block PConc Poured Contrete Slab Slab Stone Stone Wood Wood	
BsmtQual	Ex Excellent (100+ inches) Gd Good (90-99 inches) TA Typical (80-89 inches) Fa Fair (70-79 inches) Po Poor (<70 inches NA No Basement	[‘TA’,’Fa’,’Po’,’NA’] = 1 ['Gd'] = 2 ['Ex'] = 3
BsmtCond	Ex Excellent	

	Gd Good TA Typical - slight dampness allowed Fa Fair - dampness or some cracking or settling Po Poor - Severe cracking, settling, or wetness NA No Basement	
BsmtExposure	Gd Good Exposure Av Average Exposure (split levels or foyers typically score average or above) MnMimimum Exposure No No Exposure NA No Basement	
BsmtFinType1	GLQ Good Living Quarters ALQ Average Living Quarters BLQ Below Average Living Quarters Rec Average Rec Room LwQ Low Quality Unf Unfinshed NA No Basement	
BsmtFinType2	GLQ Good Living Quarters ALQ Average Living Quarters BLQ Below Average Living Quarters Rec Average Rec Room	

	LwQ Low Quality Unf Unfinshed NA No Basement	
Heating	Floor Floor Furnace GasA Gas forced warm air furnace GasW Gas hot water or steam heat Grav Gravity furnace OthW Hot water or steam heat other than gas Wall Wall furnace	
HeatingQC	Ex Excellent Gd Good TA Average/Typical Fa Fair Po Poor	
CentralAir	N No Y Yes	[‘N’] = 1 [‘Y’] = 2
Electrical	SBrkr Standard Circuit Breakers & Romex FuseA Fuse Box over 60 AMP and all Romex wiring (Average) FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair) FuseP 60 AMP Fuse Box and mostly knob & tube wiring (poor) Mix Mixed	[‘FuseA’,’ FuseF’,’ FuseP’,’ Mix’] = 1 [‘SBrkr’] = 2
KitchenQual	Ex Excellent	[‘Fa’, ‘Po’] = 1

	Gd Good TA Typical/Average Fa Fair Po Poor	['TA'] = 2 ['Gd'] = 3 ['Ex'] = 4
Functional	Typ Typical Functionality Min1 Minor Deductions 1 Min2 Minor Deductions 2 Mod Moderate Deductions Maj1 Major Deductions 1 Maj2 Major Deductions 2 SevSeverely Damaged Sal Salvage only	
FireplaceQu	Ex Excellent - Exceptional Masonry Fireplace Gd Good - Masonry Fireplace in main level TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement Fa Fair - Prefabricated Fireplace in basement Po Poor - Ben Franklin Stove NA No Fireplace	
GarageType	2Types More than one type of garage Attchd Attached to home Basment Basement Garage	

	BuiltIn Built-In (Garage part of house - typically has room above garage) CarPort Car Port Detchd Detached from home NA No Garage	
GarageFinish	Fin Finished RFn Rough Finished Unf Unfinished NA No Garage	
GarageQual	Ex Excellent Gd Good TA Typical/Average Fa Fair Po Poor NA No Garage	
GarageCond	Ex Excellent Gd Good TA Typical/Average Fa Fair Po Poor NA No Garage	
PavedDrive	Y Paved P Partial Pavement N Dirt/Gravel	
PoolQC	Ex Excellent	

	Gd Good TA Average/Typical Fa Fair NA No Pool	
Fence	GdPrv Good Privacy MnPrv Minimum Privacy GdWo Good Wood MnWw Minimum Wood/Wire NA No Fence	
MiscFeature	Elev Elevator Gar2 2nd Garage (if not described in garage section) Othr Other Shed Shed (over 100 SF) TenC Tennis Court NA None	
SaleType	WD Warranty Deed - Conventional CWD Warranty Deed - Cash VWD Warranty Deed - VA Loan New Home just constructed and sold COD Court Officer Deed/Estate Con Contract 15% Down payment regular terms ConLw Contract Low Down payment and low interest	['Oth'] = 1 [‘WD’,’ VWD’,’ COD’,’ ConLw’,’ ConLI’,’ ConLD’] = 2 ['CWD'] = 3 ['New', 'Con'] = 4

	ConLI Contract Low Interest ConLD Contract Low Down Oth Other	
SaleCondition	Normal Normal Sale Abnorml Abnormal Sale - trade, foreclosure, short sale AdjLand Adjoining Land Purchase Alloca Allocation - two linked properties with separate deeds, typically condo with a garage unit Family Sale between family members Partial Home was not completed when last assessed (associated with New Homes)	

1 output variables:

Name	Type
SalePrice_Log	Float

Note: “SalePrice_Log” is “SalePrice” after log transform

3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

Pre-processing and feature extraction techniques:

(1) Checked the distribution of target variable “SalePrice”, and found it is not normally distributed (Skewness:1.882876 Kurtosis: 6.536282) which can reduce the performance of the machine learning models. I made a log transformation for “SalePrice” to transform it to normal distribution more likely. (Skewness:0.121347 Kurtosis:0.809519).
(2) Dealt with missing data
(3) Checked skewness and kurtosis of numerical features. Some of the feature values are not normally distributed and it is better to use log values. I only chose features

"GrLivArea" and "LotArea" to make a log transform because many of the features are skewed, but not for all of them the correlation coefficient (to SalePrice) increases after log transform. And I did not check all features but for "GrLivArea" it did increase. In order to not spend too much work on this part, since the performance also depends on the models and their hyperparameters, I only chose these two features.

(4) Calculated a correlation matrix and plotted a heatmap. And make a list of numerical features and their correlation coefficient to target in descending order. Chose most correlated features (correlation coefficient>0.4) and dropped those features below this threshold.

(5) Plotted box plots for all categorical features to "SalePrice". From the box plot we can see that for many of the categorical features have no strong relation to the target. So, I would drop those categorical features which has weak relation to target. After dropping all columns with weak correlation to target, only 24 features left.

(6) Converted categorical to numerical. To investigate the relation of the categories to "SalePrice" in more detail, I made violin plots for these features. And I look at the mean of "SalePrice" as function of category. I put those categories together of each feature which has similar mean value to form a new category. And encode these categories as 1, 2, 3, and 4.

(7) After converted categorical columns to numerical. I checked these new numerical features correlation to "SalePrice" and dropped the converted categorical columns and the new numerical columns with weak correlation.

(8) Created a correlation matrix of those remaining features with strong correlation to target and made a heatmap. Then checked for multicollinearity. Selected strong correlation of these features to other (similar features) and drop the one has smaller correlation coefficient to target.

(9) For those left categorical features (notice that we have converted these features to numerical in (6)), I used one-hot encoding. Please notice that there are two classes of categorical data, nominal and ordinal. So, actually I only one-hot encoding those nominal categorical features. But for ordinal categorical features like "kitchenQual" which means the quality of kitchen I left unchanged because "4" stands for excellent,"3" stands for good, "2" stands for Typical/Average and "1" stands for poor/fair which is meaningful.

(10) Standardize features except those features after one-hot encoding.

(11) Using "train_test_split" to extract 30% data randomly from training data which has been cleaned as testing data.

Dealt with missing data:

(1) I found some features with 'NaN' values have meaning, like for feature 'PoolQC', 'NaN'means no pool. So, for those features I replaced 'NaN' with 'None'.

(2) After that, I found that only a few numerical features have missing data and I filled the missing values with mean value.

3.3. Dataset Methodology

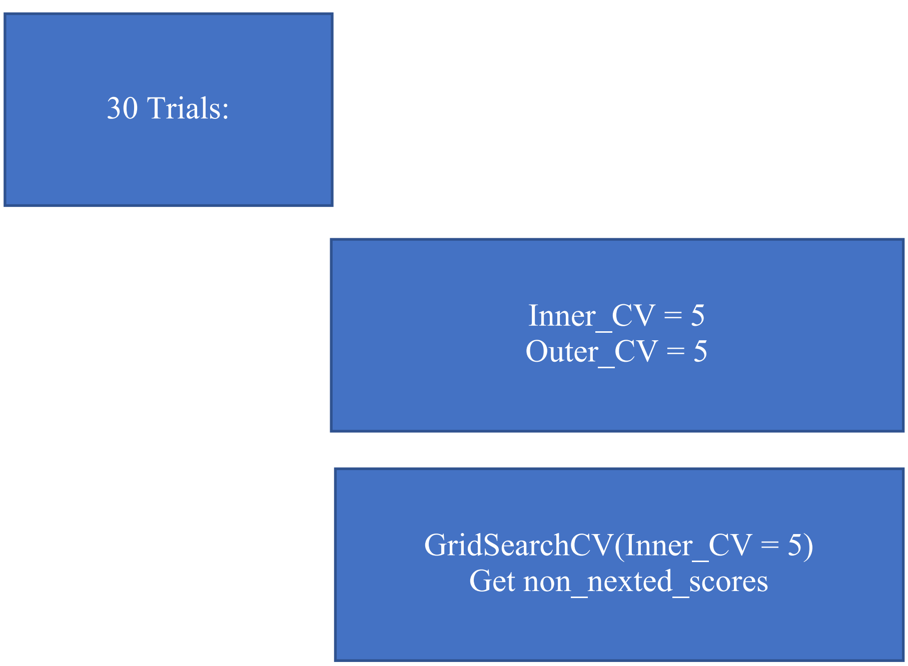
When training my regression models, I used “GridSearchCV” which has parameter “cv”. “cv” determines the cross-validation splitting strategy. And I chose cv = 5 which means 5-fold validation is used.

The training set is split into k smaller sets. And the following procedure is followed for each of the k”fold”:

(1) A model is trained using k−1 of the folds as training data;
(2) The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive but does not waste too much data.

The cross validation here of my project is nested cross validation because it trains a model in which hyperparameters also need to be optimized. I used Nested CV to estimate the generalization error of the underlying model and its hyperparameter search. Choosing the parameters that maximize non-nested CV biases the model to the dataset, yielding an overly-optimistic score.
In the inner loop (executed by “GridSearchCV”), the score is approximately maximized by fitting a model to each training set, and then directly maximized in selecting hyperparameters over the validation set. After the best hyperparameters were chosen, I applied their parameters to the regression models. In the outer loop(“cross_val_score”), generalization error is estimated by averaging test set scores (here is RMSE) over several dataset splits (here is CV = 5). I made these two inner and outer loops 30 trials and chose the best scores of these 30 trials and chose its corresponding parameters.



Cross_val_score(Outer_CV = 5)
Get non_nexted_scores

3.4. Training Process

(1) Linear regression：

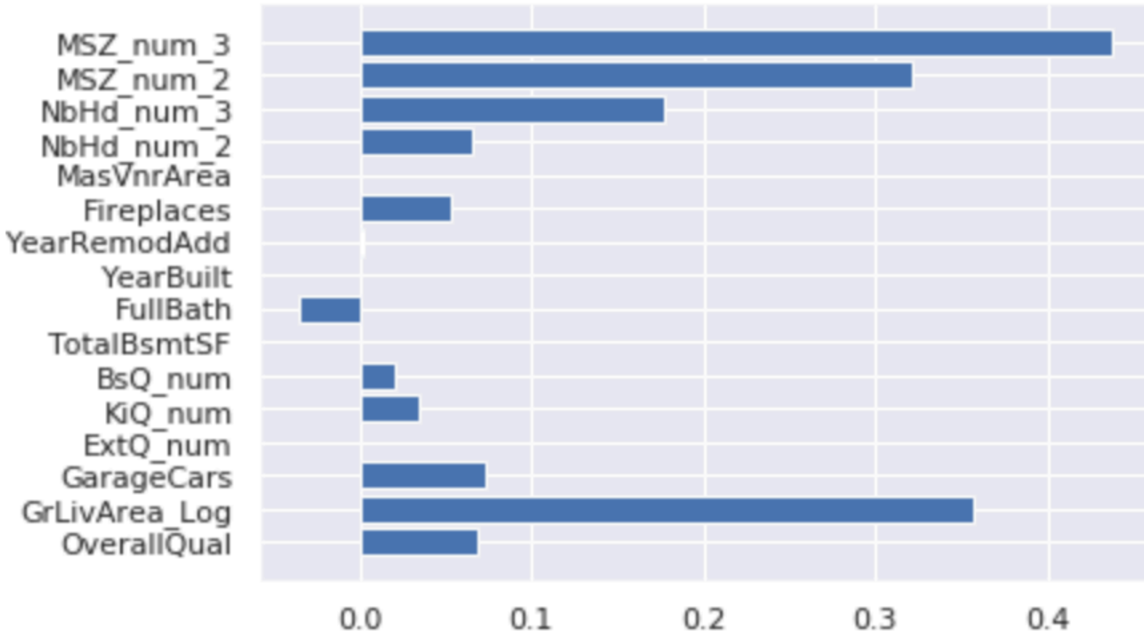
• Linear regression is a linear approach to modelling the relationship between a scalar response and one or more independent variables.

Given a data set $\{y_i , x_{i1},\ldots,x_{ip}\}$ (i=1 to n) statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus, the model takes the form

$$Y_i = \beta_0 1 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \varepsilon_i,\ i = 1,\ldots,n$$

• The goal of my project is to predict “SalesPrice” which is the response of 79 input variables. So, I chose linear regression.
• The parameters are chosen by nested cross validation, since I used “GridSearchCV” to train a model in which hyperparameters also need to be optimized. And the hyperparameters are {'cop y_X': True, 'fit_intercept': True, 'normalize': False}, and the coefficients of the linear model I trai ned are:
6.88058773e-02 , 3.56044415e-01 ,7.39500204e-02, -8.84556791e-05,
3.43553140e-02 , 1.98871920e-02 , 8.75932425e-05 ,-3.44931865e-02,
4.33341578e-04 , 1.71874422e-03 , 5.38244984e-02 ,-1.29350179e-06,

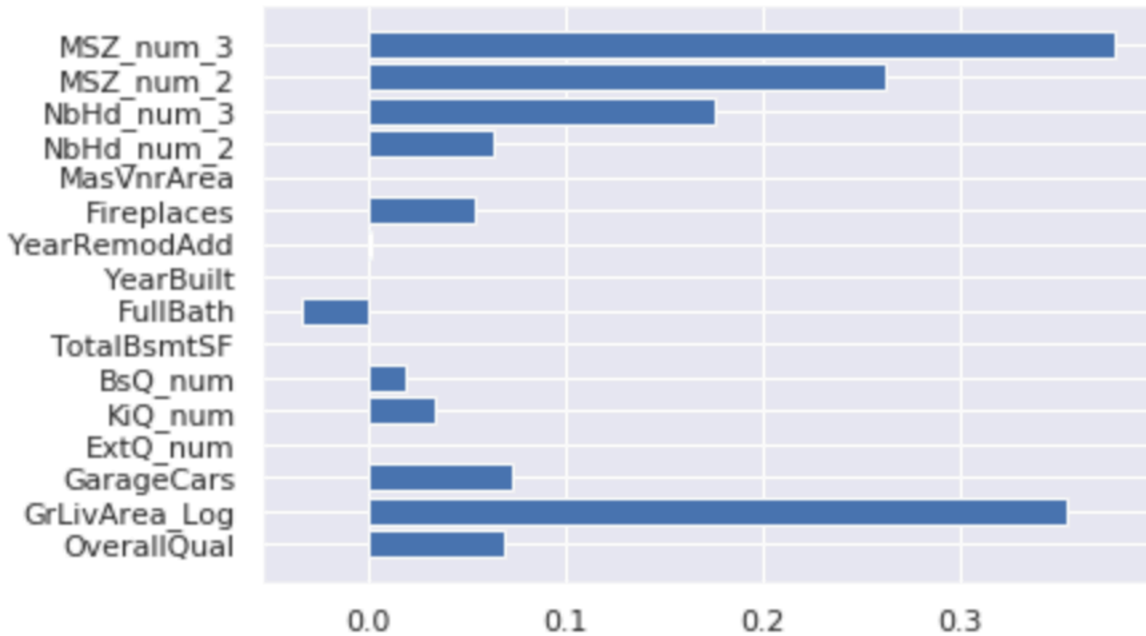
6.51776805e-02 , 1.77861383e-01 , 3.21457098e-01 ,4.36995978e-01



• To avoid overfitting cross validation was used.
• The RMSE (nested CV) of this model is 0.153411

(2) Ridge regression:

• Ridge regression is an extension for linear regression. It's basically a regularized linear regression model. This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. And the formula is:
L=∑(Ŷi-Yi)2+λ∑β2
• Compared to linear regression, it has regularization which can "punish" the loss function for high values of coefficients β. So, it can avoid overfitting. Cross validation can also prevent overfitting.
• The parameters are chosen by nested cross validation, just the same as the linear regression. And the coefficients of the Ridge regression model I trained are:
6.87345962e-02 , 3.53130801e-01 ,7.30701587e-02 , 1.10768672e-03,
3.43354382e-02 , 1.98722035e-02 , 8.74652773e-05, -3.24581171e-02,
4.50194250e-04 , 1.72653758e-03 , 5.45383551e-02, 5.51201247e-07,
6.40145025e-02 , 1.75459123e-01 , 2.61434994e-01 , 3.77349055e-01

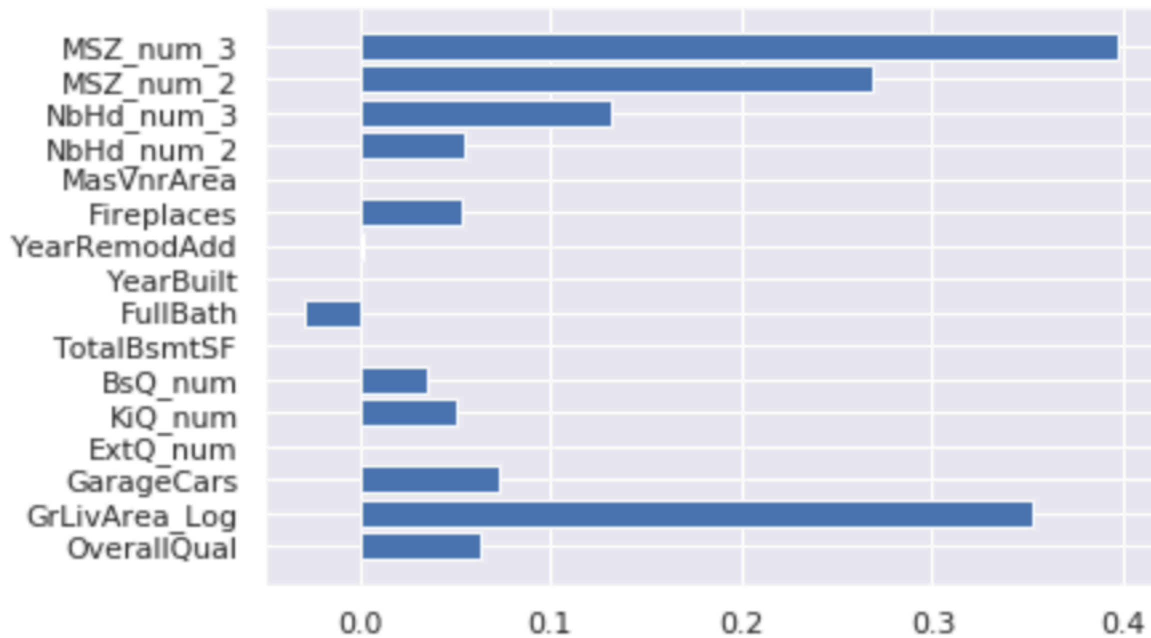


• The RMSE (nested CV) of this model is 0.156399

(3) LASSO regression:

• Lasso is another extension built on regularized linear regression, but with a small twist. The loss function of Lasso is in the form: L=∑(Ŷi-Yi)2+λ∑|β|, The only difference from Ridge regression is that the regularization term is in absolute value. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant.

• The parameters are chosen by nested cross validation, just the same as the linear regression. And the coefficients of the LASSO regression model I trained are:
6.39494386e-02 , 3.52096941e-01 , 7.34645453e-02 , 1.09733930e-03,
5.06300481e-02 , 3.46560700e-02 , 8.71855867e-05, -2.83620391e-02,
3.35148259e-04 , 1.92712516e-03 , 5.37091348e-02 ,6.76171521e-06,
5.52087386e-02, 1.31624505e-01 , 2.69014680e-01 , 3.96810017e-01



• The RMSE (nested CV) of this model is 0.155921

(4) Decision Tree Regressor

• In a regression tree the idea is this: since the target variable does not have classes, we fit a regression model to the target variable using each of the independent variables. Then for each independent variable, the data is split at several split points. At each split point, the "error" between the predicted value and the actual values is squared to get a "Sum of Squared Errors (SSE)". The split point errors across the variables are compared and the variable/point yielding the lowest SSE is chosen as the root node/split point. This process is recursively continued.
• Decision Tree is commonly used tool in machine learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. So, I used it in this regressor problem.
•. I used “GridSearchCV” to search the best parameters for Decision Tree Regressor and I set CV = 5. And the parameters are: { 'max_depth': 10, 'max_features': 12, 'max_leaf_nodes': None, 'min_samples_split': 20, 'presort': False}
•. To avoid overfitting, cross validation wad applied when choosing parameters in “GridSearchCV”.
• The RMSE of this model is 0.207669

(5) Random Forest Regressor

• Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.
• Since I have used decision tree, no doubt that Random Forest Regressor will be used next. Random Forest is a flexible, easy to use machine learning algorithm that produces, even

without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks.

• Firstly, chose a wide range of parameters using "RandomizedSearchCV" which is imported form sklearn. In contrast to "GridSearchCV", not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. After parameters were chosen by "RandomizedSearchCV" setting a narrow range of parameters near those parameters in "GridSearchCV". Then chose the best parameters from "GridSearchCV". The parameters are:

• Most of the time overfitting won't happen that easy to a random forest regressor. That's because if there are enough trees in the forest, the regressor won't overfit the model. Besides, cross validation was applied. And the parameters are:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=27,
        max_features='sqrt', max_leaf_nodes=None,
        min_impurity_decrease=0.0, min_impurity_split=None,
        min_samples_leaf=1, min_samples_split=3,
        min_weight_fraction_leaf=0.0, n_estimators=800, n_jobs=None,
        oob_score=False, random_state=None, verbose=0, warm_start=False)
```

• The RMSE of this model is 0.139160

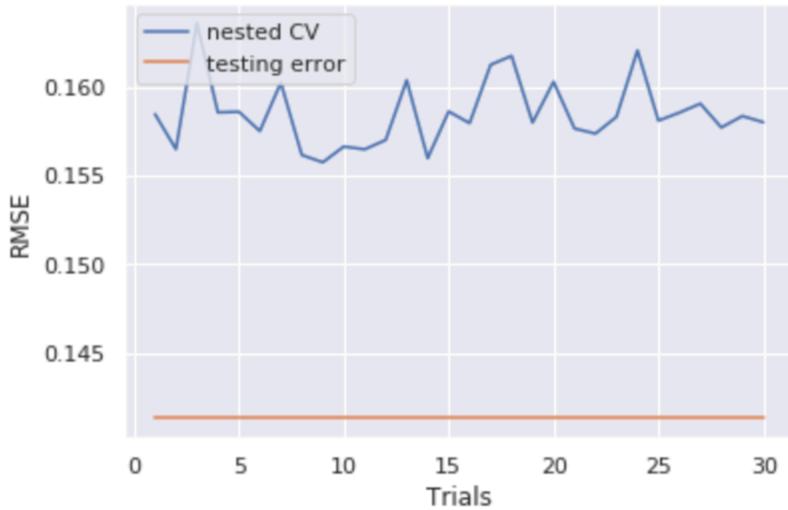
3.5. Model Selection and Comparison of Results

Obviously, house prices prediction is a classic regression problem. So, I chose linear regression first. Then I chose LASSO and Ridge Regression which are both extensions for linear regression. Then I chose "Decision Tree Regressor" and "Random Forest Regressor" which were covered in class and wanted to find out the performances of these two models.

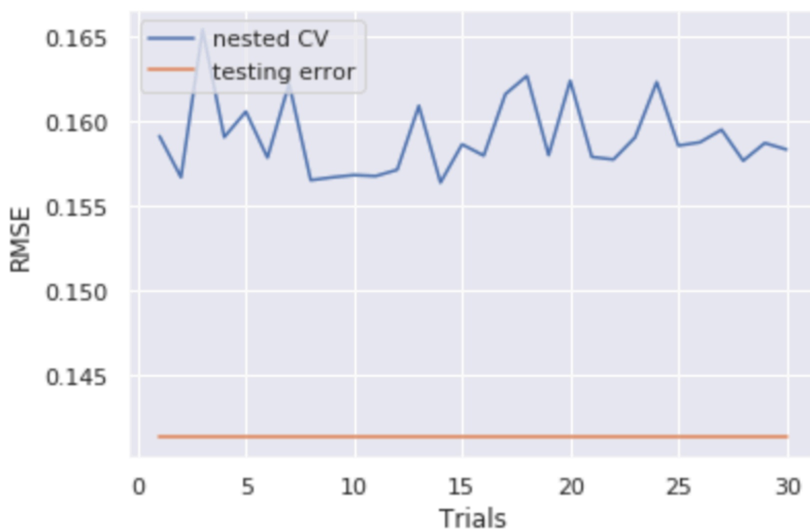
For linear, ridge and LASSO regression I repeated calculating error 30 trials and for Decision Tree Regressor I did 5 trials and random forest 1 trail since these two are time consuming.
"nested CV": how I get "nested CV" have been stated in 3.3.

"testing error": using those 30% data I extracted as testing data and using got the error from the model I trained.

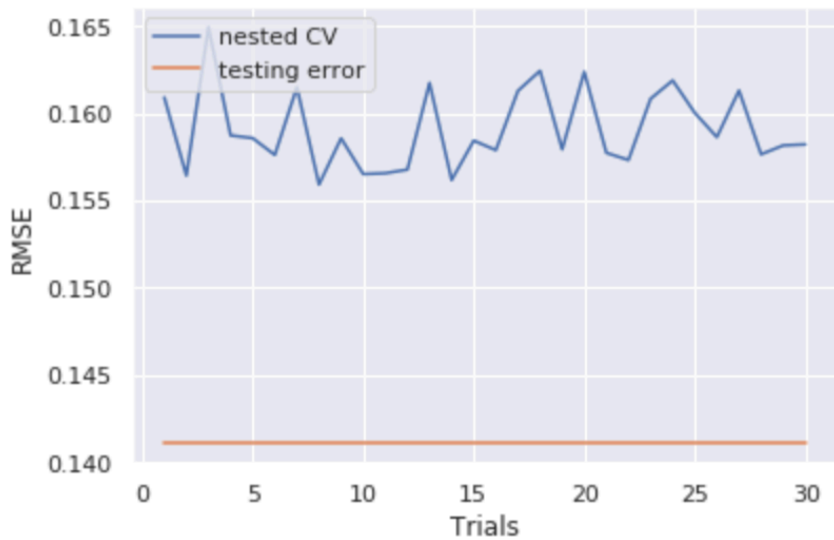
(1) Linear regression



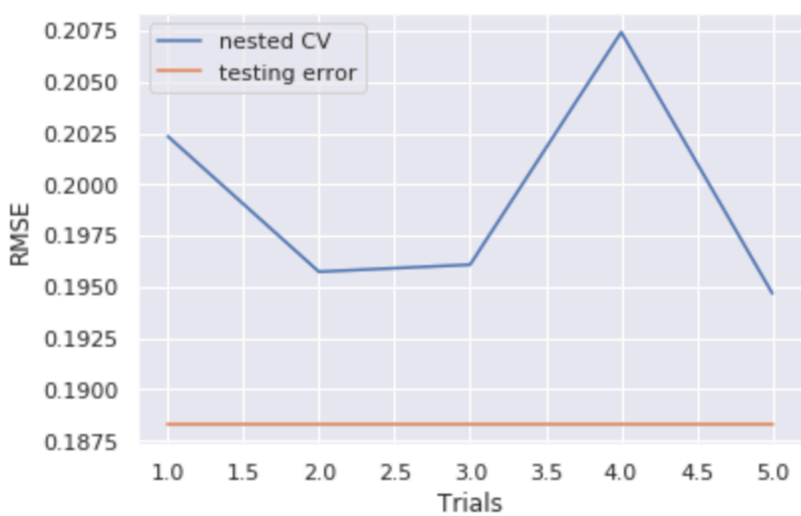
(2) Ridge regression



(3) LASSO regression



(4) Decision Tree Regressor



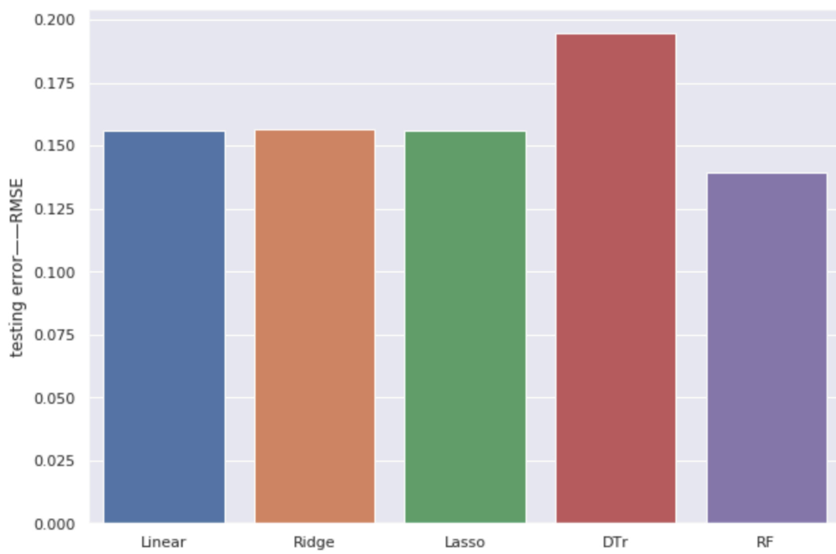
(5) Random Forest Regressor

The testing RMSE of this model is `0.139160`

4. Final Results and Interpretation

The final results:

Comparing all these models (linear regression, ridge regression, LASSO regression, decision tree regressor and random forest regressor), the best model is random forest regressor which has th e smallest testing RMSE as the bar chart shows below. And the parameters for this model are {' bootstrap': True, 'max_depth': 27, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_s plit': 3, 'n_estimators': 800}



Clearly that random forest regressor did better job than the other regression models I trained. I think training data has 16 features and do not has a strong linear shape. Random forest can capture the non-linear features better. And random forest is also a strong algorithm, even with default hyperparameters often produce a good prediction result. And we can see that though decision tree regressor did the worst job, a collection of decision trees (random forest) can get the best result.

5. Contributions of each team member

Not a team project

6. Summary and conclusions

I found that many techniques need to be applied to data cleaning and sometimes it is even a little tricky. And random forest regressor is a very handy and easy algorithm, without too much parameters tuning, it can always get good results. In this project, I did not apply deep learning technique, I think that deep learning would be better to this problem.

7. References

Machine learning A Probabilistic Perspective, Kevin P. Murphy
Learning from data, Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin

Your code

Please submit your code in a separate file. All your code should be in one pdf file, machine readable (no screen shots or scans).