```
#This notebook is a bit messy but I put the results in the first few cells to make it easier


display(final_train)
```

	Id	SalePrice	ridge_model_price	linear_model_price	lasso_model_price	elastic_model_price
1	1	208500	214797.43788388127	214406.10067660263	214411.971238948	225584.95143811055
2	2	181500	215731.41824487317	217751.89047108532	217732.24891400692	178909.27589976997
3	3	223500	220143.55405740358	219796.6012088054	219801.62907978118	231821.00226445473
4	4	140000	191402.1133883536	191584.24601617083	191584.22104466893	174410.58857925516
5	5	250000	316767.9806316928	318026.9493148204	318025.5002663215	285585.3972052932
6	6	143000	144782.69406765082	140299.1632785655	140355.826934329	171927.2970941502
7	7	307000	259116.1538563643	259481.36658941134	259472.1051557493	251035.7951268186

Showing the first 1000 rows.

```
display(final_result)
```

	Id	linear_house_price	ridge_house_price	lasso_house_price	elastic_house_price
1	1461	101441.97865741357	101368.64538626082	101443.261228049	104985.99221427762
2	1462	141068.59115862392	141063.4904776786	141069.6398907305	149038.16735018743
3	1463	145560.34190601483	145172.03160546592	145559.04011565103	169486.29642034625
4	1464	162000.8161554266	161653.69167058612	161997.8499768888	186649.1019133909
5	1465	212975.59592510242	211355.2678532838	212975.92301015125	188420.4727978215
6	1466	150850.23631544388	150457.9390844038	150846.95018223824	177062.32997173304
7	1467	150326.94098243752	149504.9743346785	150318.34024494264	163901.68500267924

Showing the first 1000 rows.

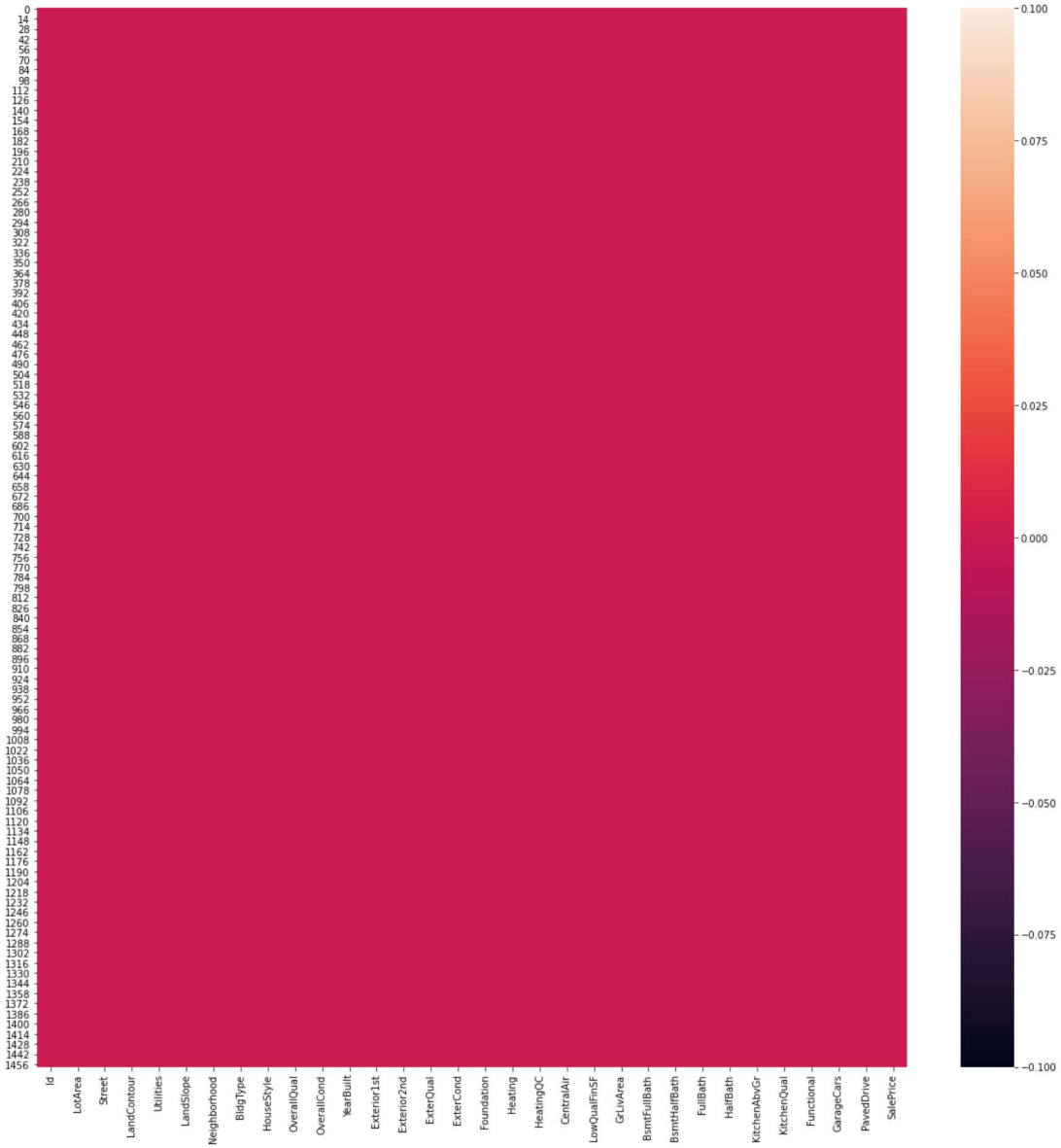
```
ridge_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['ridge_model_price'].to_numpy())
linear_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['linear_model_price'].to_numpy())
lasso_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['lasso_model_price'].to_numpy())
elastic_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['elastic_model_price'].to_numpy())
print('ridge_mse: ', ridge_mse)
print('linear_mse: ', linear_mse)
print('lasso_mse: ', lasso_mse)
print('elastic_mse: ', elastic_mse)

ridge_mse:  824021563.1103175
linear_mse:  826596480.5097375
lasso_mse:  821371880.6783286
elastic_mse:  1301876336.6182091

###BEGGINNING OF MY CODE TO FIND THE ABOVE VALUES


%sql
Create Database MLMod_2
```

```
sample_submission = spark.read.format("csv").load("dbfs:/FileStore/shared_uploads/MichaelRocchio2023@u.northwestern.edu/sample_submission-1.csv", header=True, inferSchema=True)
test = spark.read.format("csv").load("dbfs:/FileStore/shared_uploads/MichaelRocchio2023@u.northwestern.edu/test-1.csv", header=True, inferSchema=True)
train = spark.read.format("csv").load("dbfs:/FileStore/shared_uploads/MichaelRocchio2023@u.northwestern.edu/train-1.csv", header=True, inferSchema=True)

sample_submission.write.saveAsTable('MLMod_2.sample_submission', format='parquet', mode='overwrite')
test.write.saveAsTable('MLMod_2.test', format='parquet', mode='overwrite')
train.write.saveAsTable('MLMod_2.train', format='parquet', mode='overwrite')


from pyspark.sql import SparkSession
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import seaborn as sns
from sklearn.preprocessing import scale
import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smf

train=spark.sql("""
Select *
From MLMod_2.train""").toPandas()
test=spark.sql("""
Select *
From MLMod_2.test""").toPandas()

train_var=train[['Id', 'LotArea', 'Street', 'LandContour', 'Utilities', 'LandSlope', 'Neighborhood', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'Exterior1st', 'Exterior2nd',
'ExterQual', 'ExterCond', 'Foundation', 'Heating', 'HeatingQC', 'CentralAir', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'KitchenAbvGr', 'KitchenQual',
'Functional', 'GarageCars', 'PavedDrive', 'SalePrice']]

test_var=test[['Id', 'LotArea', 'Street', 'LandContour', 'Utilities', 'LandSlope', 'Neighborhood', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'Exterior1st', 'Exterior2nd',
'ExterQual', 'ExterCond', 'Foundation', 'Heating', 'HeatingQC', 'CentralAir', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'KitchenAbvGr', 'KitchenQual',
'Functional', 'GarageCars', 'PavedDrive']]
test_var=test_var.set_index(['Id'])
train_var

Out[2]:
```

	Id	LotArea	Street	LandContour	Utilities	LandSlope	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	Exterior1st	Exterior2nd	ExterQual	ExterCond	Foundation	Heating	HeatingQC	CentralAir	LowQualFinSF	GrLiv
0	1	8450	Pave	Lvl	AllPub	Gtl	CollgCr	1Fam	2Story	7	5	2003	VinylSd	VinylSd	Gd	TA	PConc	GasA	Ex	Y	0	
1	2	9600	Pave	Lvl	AllPub	Gtl	Veenker	1Fam	1Story	6	8	1976	MetalSd	MetalSd	TA	TA	CBlock	GasA	Ex	Y	0	
2	3	11250	Pave	Lvl	AllPub	Gtl	CollgCr	1Fam	2Story	7	5	2001	VinylSd	VinylSd	Gd	TA	PConc	GasA	Ex	Y	0	
3	4	9550	Pave	Lvl	AllPub	Gtl	Crawfor	1Fam	2Story	7	5	1915	Wd Sdng	Wd Shng	TA	TA	BrkTil	GasA	Gd	Y	0	
4	5	14260	Pave	Lvl	AllPub	Gtl	NoRidge	1Fam	2Story	8	5	2000	VinylSd	VinylSd	Gd	TA	PConc	GasA	Ex	Y	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	7917	Pave	Lvl	AllPub	Gtl	Gilbert	1Fam	2Story	6	5	1999	VinylSd	VinylSd	TA	TA	PConc	GasA	Ex	Y	0	
1456	1457	13175	Pave	Lvl	AllPub	Gtl	NWAmes	1Fam	1Story	6	6	1978	Plywood	Plywood	TA	TA	CBlock	GasA	TA	Y	0	
1457	1458	9042	Pave	Lvl	AllPub	Gtl	Crawfor	1Fam	2Story	7	9	1941	CemntBd	CmentBd	Ex	Gd	Stone	GasA	Ex	Y	0	
1458	1459	9717	Pave	Lvl	AllPub	Gtl	NAmes	1Fam	1Story	5	6	1950	MetalSd	MetalSd	TA	TA	CBlock	GasA	Gd	Y	0	
1459	1460	9937	Pave	Lvl	AllPub	Gtl	Edwards	1Fam	1Story	5	6	1965	HdBoard	HdBoard	Gd	TA	CBlock	GasA	Gd	Y	0	

1460 rows × 32 columns

```
plt.figure(figsize=(20, 20))
sns.heatmap(train_var.isnull())
plt.show()
```




```
train_var.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 32 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Id            1460 non-null   int32
 1   LotArea       1460 non-null   int32
 2   Street        1460 non-null   object
 3   LandContour   1460 non-null   object
 4   Utilities     1460 non-null   object
 5   LandSlope     1460 non-null   object
 6   Neighborhood  1460 non-null   object
 7   BldgType      1460 non-null   object
 8   HouseStyle    1460 non-null   object
 9   OverallQual   1460 non-null   int32
 10  OverallCond   1460 non-null   int32
 11  YearBuilt     1460 non-null   int32
 12  Exterior1st   1460 non-null   object
 13  Exterior2nd   1460 non-null   object
```

```
 14  ExterQual     1460 non-null   object
 15  ExterCond     1460 non-null   object
```

```
test_var.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1459 entries, 1461 to 2919
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   LotArea       1459 non-null   int32
 1   Street        1459 non-null   object
 2   LandContour   1459 non-null   object
 3   Utilities     1459 non-null   object
 4   LandSlope     1459 non-null   object
 5   Neighborhood  1459 non-null   object
 6   BldgType      1459 non-null   object
 7   HouseStyle    1459 non-null   object
 8   OverallQual   1459 non-null   int32
 9   OverallCond   1459 non-null   int32
 10  YearBuilt     1459 non-null   int32
 11  Exterior1st   1459 non-null   object
 12  Exterior2nd   1459 non-null   object
 13  ExterQual     1459 non-null   object
 14  ExterCond     1459 non-null   object
 15  Foundation    1459 non-null   object
```

```
bad_var=['GarageCars', 'BsmtFullBath', 'BsmtHalfBath']
test_var['GarageCars']=test_var['GarageCars'].fillna(0)
test_var['GarageCars']=test_var.GarageCars.apply(
  lambda x:0 if type(x) != 'int32' else x)
test_var['GarageCars']=pd.to_numeric(test_var['GarageCars'])

test_var['BsmtFullBath']=test_var['BsmtFullBath'].fillna(0)
test_var['BsmtFullBath']=test_var.BsmtFullBath.apply(
  lambda x:0 if type(x) != 'int32' else x)
test_var['BsmtFullBath']=pd.to_numeric(test_var['BsmtFullBath'])

test_var['BsmtHalfBath']=test_var['BsmtHalfBath'].fillna(0)
test_var['BsmtHalfBath']=test_var.BsmtHalfBath.apply(
  lambda x:0 if type(x) != 'int32' else x)
test_var['BsmtHalfBath']=pd.to_numeric(test_var['BsmtHalfBath'])

test_var.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1459 entries, 1461 to 2919
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   LotArea       1459 non-null   int32
 1   Street        1459 non-null   object
 2   LandContour   1459 non-null   object
 3   Utilities     1459 non-null   object
 4   LandSlope     1459 non-null   object
 5   Neighborhood  1459 non-null   object
 6   BldgType      1459 non-null   object
 7   HouseStyle    1459 non-null   object
 8   OverallQual   1459 non-null   int32
 9   OverallCond   1459 non-null   int32
 10  YearBuilt     1459 non-null   int32
 11  Exterior1st   1459 non-null   object
 12  Exterior2nd   1459 non-null   object
 13  ExterQual     1459 non-null   object
 14  ExterCond     1459 non-null   object
 15  Foundation    1459 non-null   object
```

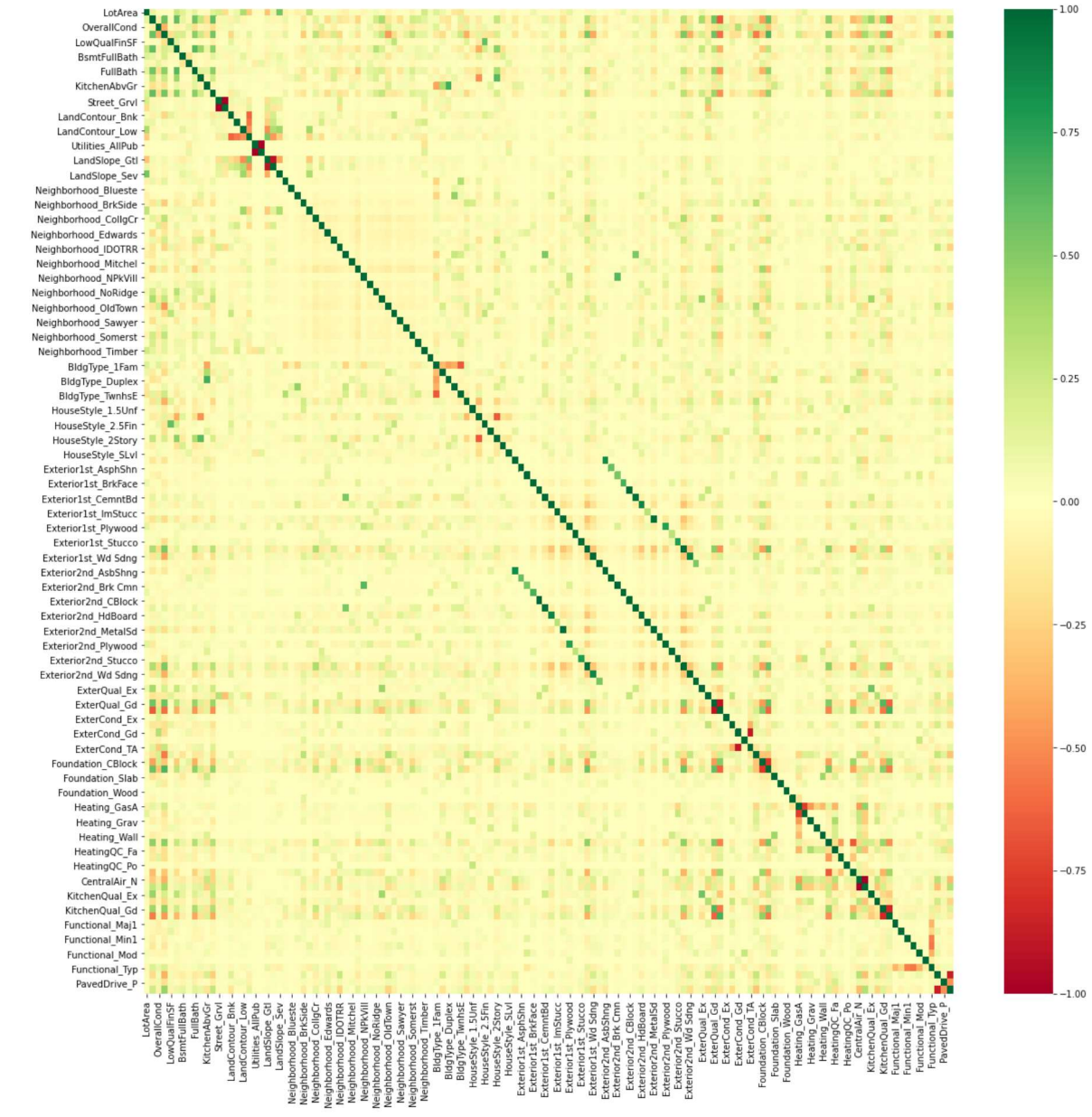
```
train_var_cat=train_var.select_dtypes(['object'])

train_var_num=train_var.select_dtypes(['int32', 'int64'])

train_dummies = pd.get_dummies(train_var_cat)

train_var_fin=pd.concat([train_var_num, train_dummies], axis=1)
train_var_fin=train_var_fin.set_index(['Id'])
train_var_price=train_var_fin[['SalePrice']]
del train_var_fin['SalePrice']
train_var_price
```

Out[7]:

Id	SalePrice
1	208500
2	181500
3	223500
4	140000
5	250000
...	...
1456	175000
1457	210000
1458	266500
1459	142125
1460	147500

1460 rows × 1 columns

```
plt.figure(figsize=(20, 20))
g = sns.heatmap(round(train_var_fin.corr(),2),annot=False,cmap="RdYlGn")
```




```
train_var_finf
```

Out[13]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandContour_Low	LandContour_L
1	8450	7	5	2003	0	1710	1	0	2	1	1	2	0	1	0	0	0	
2	9600	6	8	1976	0	1262	0	1	2	0	1	2	0	1	0	0	0	
3	11250	7	5	2001	0	1786	1	0	2	1	1	2	0	1	0	0	0	
4	9550	7	5	1915	0	1717	1	0	1	0	1	3	0	1	0	0	0	
5	14260	8	5	2000	0	2198	1	0	2	1	1	3	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1456	7917	6	5	1999	0	1647	0	0	2	1	1	2	0	1	0	0	0	
1457	13175	6	6	1978	0	2073	1	0	2	0	1	2	0	1	0	0	0	
1458	9042	7	9	1941	0	2340	0	0	2	0	1	1	0	1	0	0	0	
1459	9717	5	6	1950	0	1078	1	0	1	0	1	1	0	1	0	0	0	
1460	9937	5	6	1965	0	1256	1	0	1	1	1	1	0	1	0	0	0	

1460 rows × 134 columns

```
regr = skl_lm.LinearRegression()

y = train_var_price['SalePrice'].to_numpy()

X = train_var_fin.to_numpy()



regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[ 4.93788383e-01  1.08985693e+04  5.59064573e+03  2.22100973e+02
 -2.25255302e+01  5.85801974e+01  1.36170148e+04  8.21016077e+03
  6.64178851e+03  5.64422047e+03 -1.82842382e+04  9.72384310e+03
 -1.60383608e+04  1.60383608e+04 -1.73484983e+04  9.68020412e+03
```

```
  4.84832494e+03  2.81996922e+03  1.88906334e+04 -1.88906334e+04
 -8.16725398e+02  1.07764501e+04 -9.95972469e+03  5.26391188e+03
 -2.10948211e+03  7.62746740e+03 -7.25579017e+03 -2.86216794e+03
 -7.75001809e+03  1.32384182e+04 -2.39711030e+04 -1.39210654e+04
 -1.62674179e+04  1.28919641e+03 -1.82235908e+04 -1.85463860e+04
  2.86470916e+03 -2.16706714e+04  4.44586742e+04  3.82847487e+04
 -1.97597718e+04 -1.31827975e+04 -1.67519384e+04 -9.60282256e+03
  1.07284116e+04  5.16953209e+04  1.67312308e+03  1.47510416e+04
  1.49739782e+04  1.47171946e+04  4.58447609e+03 -2.17589245e+04
 -1.25167244e+04 -3.98107941e+03  6.39277619e+03  1.07008528e+04
 -1.08529472e+04 -7.32642079e+03 -1.05499666e+04  1.18894525e+04
  3.72733259e+03  1.46934018e+04 -1.11015324e+04 -7.53042670e+03
  2.44244424e+04 -3.37706943e+03  2.00333784e+04  2.66740851e+03
 -5.73062727e+04  9.04646491e+03  6.07213181e+03 -1.46261383e+04
  4.78145151e+03 -7.75634440e+02  3.21820891e+03  9.78018570e+03
 -6.29464871e+03  4.15164146e+03  6.91273128e+03 -6.25033050e+03
 -3.37706943e+03 -1.40800562e+04  1.67923308e+03  3.02174771e+04
```

```
test_var_cat=test_var.select_dtypes(['object'])

test_var_num=test_var.select_dtypes(['int32', 'int64'])

test_dummies = pd.get_dummies(test_var_cat)

test_var_fin=pd.concat([test_var_num, test_dummies], axis=1)
test_var_fin
```

Out[17]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandContour_Low	LandContour_L
1461	11622	5	6	1961	0	896	0	0	1	0	1	0	0	1	0	0	0	
1462	14267	6	6	1958	0	1329	0	0	1	1	1	0	0	1	0	0	0	
1463	13830	5	5	1997	0	1629	0	0	2	1	1	0	0	1	0	0	0	
1464	9978	6	6	1998	0	1604	0	0	2	1	1	0	0	1	0	0	0	
1465	5005	8	5	1992	0	1280	0	0	2	0	1	0	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2915	1936	4	7	1970	0	1092	0	0	1	1	1	0	0	1	0	0	0	
2916	1894	4	5	1970	0	1092	0	0	1	1	1	0	0	1	0	0	0	
2917	20000	5	7	1960	0	1224	0	0	1	0	1	0	0	1	0	0	0	
2918	10441	5	5	1992	0	970	0	0	1	0	1	0	0	1	0	0	0	
2919	9627	7	5	1993	0	2000	0	0	2	1	1	0	0	1	0	0	0	

1459 rows × 132 columns

```
test_var_fin
```

Out[18]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandContour_Low	LandContour_L
1461	11622	5	6	1961	0	896	0	0	1	0	1	0	0	1	0	0	0	
1462	14267	6	6	1958	0	1329	0	0	1	1	1	0	0	1	0	0	0	
1463	13830	5	5	1997	0	1629	0	0	2	1	1	0	0	1	0	0	0	
1464	9978	6	6	1998	0	1604	0	0	2	1	1	0	0	1	0	0	0	
1465	5005	8	5	1992	0	1280	0	0	2	0	1	0	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2915	1936	4	7	1970	0	1092	0	0	1	1	1	0	0	1	0	0	0	
2916	1894	4	5	1970	0	1092	0	0	1	1	1	0	0	1	0	0	0	
2917	20000	5	7	1960	0	1224	0	0	1	0	1	0	0	1	0	0	0	
2918	10441	5	5	1992	0	970	0	0	1	0	1	0	0	1	0	0	0	
2919	9627	7	5	1993	0	2000	0	0	2	1	1	0	0	1	0	0	0	

1459 rows × 132 columns

```
train_var_fin0=train_var_fin
```

```
coeff_df0=train_var_fin0[0:0]
coeff_df0=coeff_df0.T.reset_index()
coeff_df0['coeff']=regr.coef_
coeff_df1=test_var_fin[0:0].T.reset_index()
coeff_df=pd.merge(coeff_df1, coeff_df0, how='left', left_on='index', right_on='index').fillna(0)
coeff_df=coeff_df.set_index(['index'])
coeff_df
```

Out[24]:

Id	coeff
index	
LotArea	0.493788
OverallQual	10898.569270
OverallCond	5590.645730
YearBuilt	222.100973
LowQualFinSF	-22.525530
...	...
Functional_Sev	-55277.846028
Functional_Typ	20459.440387
PavedDrive_N	535.816508
PavedDrive_P	-1697.292179
PavedDrive_Y	1161.475672

132 rows × 1 columns

```
#double check
test_var_check=test_var_fin[0:0].T.reset_index()
test_var_check['name']=test_var_check['index']
test_var_check0=pd.merge(test_var_check, coeff_df, how='left', left_on='index', right_on='index').fillna(0)
test_var_check0['not_alligned']=np.where(test_var_check0['index']!=test_var_check0['name'],1,0)
print(test_var_check0['not_alligned'].sum())
```

```
0
```

```
# #allign x
train_var_check0=train_var_fin[0:0].T.reset_index()
train_var_mod=pd.merge(train_var_check0, coeff_df, how='left', left_on='index', right_on='index').fillna(0)
train_var_mod
```

Out[27]:

Id	index	coeff
0	LotArea	0.493788
1	OverallQual	10898.569270
2	OverallCond	5590.645730
3	YearBuilt	222.100973
4	LowQualFinSF	-22.525530
...	...	...
129	Functional_Sev	-55277.846028
130	Functional_Typ	20459.440387
131	PavedDrive_N	535.816508
132	PavedDrive_P	-1697.292179
133	PavedDrive_Y	1161.475672

134 rows × 2 columns

```
train_var_fin
```

Out[28]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandContour_Low	LandContour_L
1	8450	7	5	2003	0	1710	1	0	2	1	1	2	0	1	0	0	0	
2	9600	6	8	1976	0	1262	0	1	2	0	1	2	0	1	0	0	0	
3	11250	7	5	2001	0	1786	1	0	2	1	1	2	0	1	0	0	0	
4	9550	7	5	1915	0	1717	1	0	1	0	1	3	0	1	0	0	0	
5	14260	8	5	2000	0	2198	1	0	2	1	1	3	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1456	7917	6	5	1999	0	1647	0	0	2	1	1	2	0	1	0	0	0	
1457	13175	6	6	1978	0	2073	1	0	2	0	1	2	0	1	0	0	0	
1458	9042	7	9	1941	0	2340	0	0	2	0	1	1	0	1	0	0	0	
1459	9717	5	6	1950	0	1078	1	0	1	0	1	1	0	1	0	0	0	
1460	9937	5	6	1965	0	1256	1	0	1	1	1	1	0	1	0	0	0	

1460 rows × 134 columns

```
mod_check.info()
```

```
  NameError: name 'mod_check' is not defined
```

```
mod_check = train_var_fin.mul(train_var_mod['coeff'].to_numpy(), axis='columns')
mod_check['beta_zero']=regr.intercept_
mod_check['house_price_model']=mod_check.sum(axis=1)
mod_check_fin=mod_check[['house_price_model']].reset_index()
train_var_price=train_var_price.reset_index()
mod_check_fin=pd.merge(mod_check_fin, train_var_price, how='inner')
mod_check_fin=mod_check_fin.set_index(['Id'])
```

```
import sklearn
from sklearn import metrics
linear_mse = sklearn.metrics.mean_squared_error(mod_check_fin['SalePrice'].to_numpy(), mod_check_fin['house_price_model'].to_numpy())
linear_mse
```

Out[120]: 826596480.5097375

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1, solver="auto")
ridge_reg.fit(X, y)
```

Out[164]: Ridge(alpha=1)

```
train_var_fin
```

Out[44]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandContour_Low	LandContour_L
1	8450	7	5	2003	0	1710	1	0	2	1	1	2	0	1	0	0	0	
2	9600	6	8	1976	0	1262	0	1	2	0	1	2	0	1	0	0	0	
3	11250	7	5	2001	0	1786	1	0	2	1	1	2	0	1	0	0	0	
4	9550	7	5	1915	0	1717	1	0	1	0	1	3	0	1	0	0	0	
5	14260	8	5	2000	0	2198	1	0	2	1	1	3	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1456	7917	6	5	1999	0	1647	0	0	2	1	1	2	0	1	0	0	0	
1457	13175	6	6	1978	0	2073	1	0	2	0	1	2	0	1	0	0	0	
1458	9042	7	9	1941	0	2340	0	0	2	0	1	1	0	1	0	0	0	
1459	9717	5	6	1950	0	1078	1	0	1	0	1	1	0	1	0	0	0	
1460	9937	5	6	1965	0	1256	1	0	1	1	1	1	0	1	0	0	0	

1460 rows × 134 columns

```
print(ridge_reg.coef_)
print(ridge_reg.intercept_)
```

```
[ 4.92807137e-01  1.09998940e+04  5.63324368e+03  2.42631883e+02
 -2.48480589e+01  5.85726310e+01  1.36987263e+04  8.14133656e+03
  6.65506808e+03  5.61440880e+03 -1.70148150e+04  9.81484179e+03
 -1.40312615e+04  1.40312615e+04 -1.70920592e+04  9.71995667e+03
  4.57086527e+03  2.80123726e+03  1.18128754e+04 -1.18128754e+04
 -1.33962403e+03  1.03518435e+04 -9.01221945e+03  3.55663560e+03
 -1.93393473e+03  6.72438263e+03 -5.57262599e+03 -2.06801884e+03
 -8.10475324e+03  1.34752228e+04 -2.28017573e+04 -1.41201749e+04
 -1.46742235e+04  1.03367306e+03 -1.71841477e+04 -1.75456107e+04
  3.97352484e+03 -2.07417873e+04  4.23322959e+04  3.64757174e+04
 -1.83793566e+04 -1.16654139e+04 -1.54909333e+04 -9.51470609e+03
  9.62209089e+03  4.89281297e+04  3.34157104e+02  1.33416141e+04
  1.49125758e+04  1.35369966e+04  3.49461505e+03 -2.02420442e+04
 -1.17021432e+04 -3.98521399e+03  5.10585099e+03  1.05859360e+04
 -8.52588446e+03 -7.41556007e+03 -1.05097340e+04  1.12454420e+04
  3.49916356e+03  8.31189135e+03 -5.30939524e+03 -4.83730346e+03
  2.03687284e+04 -2.42423207e+03  1.01873351e+04  1.48846078e+02
 -2.64856330e+04  5.82696563e+03  3.21111250e+03 -1.22834038e+04
 -7.53339662e+02 -2.78743706e+03  2.94652017e+02  6.53121319e+03
 -3.69584222e+03  6.02911952e+02  3.16062179e+03 -4.76487412e+03
 -2.42423207e+03 -7.17539592e+03  1.16726500e+03  2.43800729e+04
```

```
result_fin=result_fin.set_index(['Id'])
result_fin
```

Out[166]:

Id	house_price
1461	101441.978657
1462	141068.591159
1463	145560.341906
1464	162000.816155
1465	212975.595925
...	...
2915	69406.717640
2916	67622.632629
2917	133776.822936
2918	98167.360769
2919	192533.426645

1459 rows × 1 columns

```
mod_check_fin
```

Out[167]:

Id	linear_model_price
1	214406.100677
2	217751.890471
3	219796.601209
4	191584.246016
5	318026.949315
...	...
1456	177953.458132
1457	208519.313696
1458	274445.442188
1459	131079.980316
1460	142915.805056

1460 rows × 1 columns

```
%sql
Select *
From MLMod_2.train
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
1															

1	1	60	RL	65	8450	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
2	2	20	RL	80	9600	Pave	NA	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm
3	3	60	RL	68	11250	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
4	4	70	RL	60	9550	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm
5	5	60	RL	84	14260	Pave	NA	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm
6	6	50	RL	85	14115	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm
7	7	20	RL	75	10084	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	Norm

Showing the first 1000 rows.

```
ridge_rmse=train_var_fin.mul(ridge_reg.coef_, axis='columns').fillna(0)
ridge_rmse['intercept']=ridge_reg.intercept_
ridge_rmse['ridge_model_price']=ridge_rmse.sum(axis=1)
# train_var_price=train_var_price.set_index(['Id'])
ridge_rmse=pd.merge(ridge_rmse, train_var_price, right_index=True, left_index=True, how='inner')
ridge_rmse=ridge_rmse.reset_index()
final_train=ridge_rmse[['Id', 'SalePrice', 'ridge_model_price']]
final_train=final_train.set_index(['Id'])
ridge_rmse=ridge_rmse.set_index(['Id'])
mod_check_finjn=mod_check_fin
final_train=pd.merge(final_train, mod_check_finjn, how='inner', right_index=True, left_index=True)
# del final_train['index']
final_train
```

Out[259]:

Id	SalePrice	ridge_model_price	linear_model_price
1	208500	214797.437884	214406.100677
2	181500	215731.418245	217751.890471
3	223500	220143.554057	219796.601209
4	140000	191402.113388	191584.246016
5	250000	316767.980632	318026.949315
...	...	...	...
1456	175000	177766.675194	177953.458132
1457	210000	208400.237101	208519.313696
1458	266500	273262.192240	274445.442188
1459	142125	130926.729687	131079.980316
1460	147500	143738.879936	142915.805056

1460 rows × 3 columns

```
ridge_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['ridge_model_price'].to_numpy())
linear_mse = sklearn.metrics.mean_squared_error(final_train['SalePrice'].to_numpy(), final_train['linear_model_price'].to_numpy())
print('ridge_mse: ', ridge_mse)
print('linear_mse: ', linear_mse)

ridge_mse:  11836345405.18292
linear_mse:  827139122.357104

rcoeff_df0=train_var_fin[0:0]
rcoeff_df0=rcoeff_df0.T.reset_index()
rcoeff_df0['r_coeff']=ridge_reg.coef_
rcoeff_df1=test_var_fin[0:0].T.reset_index()
rcoeff_df=pd.merge(rcoeff_df1, rcoeff_df0, how='left', left_on='index', right_on='index').fillna(0)
rcoeff_df=rcoeff_df.set_index(['index'])
rcoeff_df
```

Out[173]:

index	Id	r_coeff
LotArea		0.492807
OverallQual		10999.894033
OverallCond		5633.243684
YearBuilt		242.631883
LowQualFinSF		-24.848059
...		...
Functional_Sev		-29547.016754
Functional_Typ		16263.222879
PavedDrive_N		726.171614
PavedDrive_P		-1784.138997
PavedDrive_Y		1057.967383

132 rows × 1 columns

```
coeff_df
```

Out[179]:

index	coeff
LotArea	0.493788
OverallQual	10898.569270
OverallCond	5590.645730
YearBuilt	222.100973
LowQualFinSF	-22.525530
...	...
Functional_Sev	-55277.846028
Functional_Typ	20459.440387
PavedDrive_N	535.816508
PavedDrive_P	-1697.292179
PavedDrive_Y	1161.475672

132 rows × 1 columns

```
print(ridge_reg.coef_)
print(ridge_reg.intercept_)
print(regr.coef_)
print(regr.intercept_)
```

```
[ 4.92807137e-01  1.09998940e+04  5.63324368e+03  2.42631883e+02
 -2.48480589e+01  5.85726310e+01  1.36987263e+04  8.14133656e+03
  6.65506808e+03  5.61440880e+03 -1.70148150e+04  9.81484179e+03
 -1.40312615e+04  1.40312615e+04 -1.70920592e+04  9.71995667e+03
  4.57086527e+03  2.80123726e+03  1.18128754e+04 -1.18128754e+04
 -1.33962403e+03  1.03518435e+04 -9.01221945e+03  3.55663560e+03
 -1.93393473e+03  6.72438263e+03 -5.57262599e+03 -2.06801884e+03
 -8.10475324e+03  1.34752228e+04 -2.28017573e+04 -1.41201749e+04
 -1.46742235e+04  1.03367306e+03 -1.71841477e+04 -1.75456107e+04
  3.97352484e+03 -2.07417873e+04  4.23322959e+04  3.64757174e+04
 -1.83793566e+04 -1.16654139e+04 -1.54909333e+04 -9.51470609e+03
  9.62209089e+03  4.89281297e+04  3.34157104e+02  1.33416141e+04
  1.49125758e+04  1.35369966e+04  3.49461505e+03 -2.02420442e+04
 -1.17021432e+04 -3.98521399e+03  5.10585099e+03  1.05859360e+04
 -8.52588446e+03 -7.41556007e+03 -1.05097340e+04  1.12454420e+04
  3.49916356e+03  8.31189135e+03 -5.30939524e+03 -4.83730346e+03
  2.03687284e+04 -2.42423207e+03  1.01873351e+04  1.48846078e+02
 -2.64856330e+04  5.82696563e+03  3.21111250e+03 -1.22834038e+04
 -7.53339662e+02 -2.78743706e+03  2.94652017e+02  6.53121319e+03
 -3.69584222e+03  6.02911952e+02  3.16062179e+03 -4.76487412e+03
 -2.42423207e+03 -7.17539592e+03  1.16726500e+03  2.43800729e+04
```

```
result2 = test_var_fin.mul(rcoeff_df['r_coeff'], axis='columns')
result2['beta_zero']=ridge_reg.intercept_
result2['ridge_house_price']=result2.sum(axis=1)
result2
```

Out[232]:

Id	LotArea	OverallQual	OverallCond	YearBuilt	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	KitchenAbvGr	GarageCars	Street_Grvl	Street_Pave	LandContour_Bnk	LandContour_HLS	LandCont
1461	5727.404549	54999.470164	33799.462102	475801.121615	-0.0	52481.077355	0.0	0.0	6655.068085	0.000000	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
1462	7030.879427	65999.364197	33799.462102	475073.225967	-0.0	77843.026568	0.0	0.0	6655.068085	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
1463	6815.522708	54999.470164	28166.218418	484535.869385	-0.0	95414.815862	0.0	0.0	13310.136170	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
1464	4917.229615	65999.364197	33799.462102	484778.501268	-0.0	93950.500087	0.0	0.0	13310.136170	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
1465	2466.499722	87999.152263	28166.218418	483322.709973	-0.0	74972.967651	0.0	0.0	13310.136170	0.000000	-17014.814977	0.0	-0.0	14031.261487	-0.0	9719.956672	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2915	954.074618	43999.576131	39432.705785	477984.808557	-0.0	63961.313027	0.0	0.0	6655.068085	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
2916	933.376718	43999.576131	28166.218418	477984.808557	-0.0	63961.313027	0.0	0.0	6655.068085	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
2917	9856.142745	54999.470164	39432.705785	475558.489732	-0.0	71692.900316	0.0	0.0	6655.068085	0.000000	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
2918	5145.399320	54999.470164	28166.218418	483322.709973	-0.0	56815.452048	0.0	0.0	6655.068085	0.000000	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	
2919	4744.254310	76999.258230	28166.218418	483565.341855	-0.0	117145.261954	0.0	0.0	13310.136170	5614.408796	-17014.814977	0.0	-0.0	14031.261487	-0.0	0.000000	

1459 rows × 134 columns

```
result1 = test_var_fin.mul(coeff_df['coeff'], axis='columns')
result1['beta_zero']=regr.intercept_
result1['linear_house_price']=result1.sum(axis=1)
result1=result1.reset_index()
result1_fin=result1[['Id', 'linear_house_price']]
result2 = test_var_fin.mul(rcoeff_df['r_coeff'], axis='columns')
result2['beta_zero']=ridge_reg.intercept_
result2['ridge_house_price']=result2.sum(axis=1)
result2=result2.reset_index()
result2_fin=result2[['Id', 'ridge_house_price']]
final_result=pd.merge(result1_fin, result2_fin, how='inner')
final_result
```

Out[233]:

	Id	linear_house_price	ridge_house_price
0	1461	101441.978657	101368.645386
1	1462	141068.591159	141063.490478
2	1463	145560.341906	145172.031605
3	1464	162000.816155	161653.691671
4	1465	212975.595925	211355.267853
...	...	...	...
1454	2915	69406.717640	70039.512673
1455	2916	67622.632629	67405.637979
1456	2917	133776.822936	133687.335131
1457	2918	98167.360769	99423.964631
1458	2919	192533.426645	193448.531908

1459 rows × 3 columns

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)
lcoeff_df0=train_var_fin[0:0]
lcoeff_df0=lcoeff_df0.T.reset_index()
lcoeff_df0['l_coeff']=lasso_reg.coef_
lcoeff_df1=test_var_fin[0:0].T.reset_index()
lcoeff_df=pd.merge(lcoeff_df1, lcoeff_df0, how='left', left_on='index', right_on='index').fillna(0)
lcoeff_df=lcoeff_df.set_index(['index'])

result3 = test_var_fin.mul(lcoeff_df['l_coeff'], axis='columns')
result3['beta_zero']=lasso_reg.intercept_
result3['lasso_house_price']=result3.sum(axis=1)
result3=result3.reset_index()
result3_fin=result3[['Id', 'lasso_house_price']]
final_result=pd.merge(final_result, result3_fin, how='inner')
display(final_result)
```

```
/databricks/python/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duali
ty gap: 587398875126.7362, tolerance: 920791133.4609975
  model = cd_fast.enet_coordinate_descent(
```

	Id	linear_house_price	ridge_house_price	lasso_house_price	elastic_house_price
1	1461	101441.97865741357	101368.64538626082	101443.261228049	104985.99221427762
2	1462	141068.59115862392	141063.4904776786	141069.6398907305	149038.16735018743
3	1463	145560.34190601483	145172.03160546592	145559.04011565103	169486.29642034625
4	1464	162000.8161554266	161653.69167058612	161997.8499768888	186649.1019133909
5	1465	212975.59592510242	211355.2678532838	212975.92301015125	188420.4727978215
6	1466	150850.23631544388	150457.9390844038	150846.95018223824	177062.32997173304
7	1467	150326.94098243752	149504.9743346785	150318.34024494264	163901.68500267924

Showing the first 1000 rows.

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet()
elastic_net.fit(X, y)

ecoeff_df0=train_var_fin[0:0]
ecoeff_df0=ecoeff_df0.T.reset_index()
ecoeff_df0['e_coeff']=elastic_net.coef_
ecoeff_df1=test_var_fin[0:0].T.reset_index()
ecoeff_df=pd.merge(ecoeff_df1, ecoeff_df0, how='left', left_on='index', right_on='index').fillna(0)
ecoeff_df=ecoeff_df.set_index(['index'])
resulte = test_var_fin.mul(ecoeff_df['e_coeff'], axis='columns')
resulte['beta_zero']=elastic_net.intercept_
resulte['elastic_house_price']=resulte.sum(axis=1)
resulte=resulte.reset_index()
resulte_fin=resulte[['Id', 'elastic_house_price']]
# final_result=pd.merge(final_result, resulte_fin, how='inner')
final_result
```

Out[253]:

	Id	linear_house_price	ridge_house_price	lasso_house_price	elastic_house_price
0	1461	101441.978657	101368.645386	101443.261228	104985.992214
1	1462	141068.591159	141063.490478	141069.639891	149038.167350
2	1463	145560.341906	145172.031605	145559.040116	169486.296420
3	1464	162000.816155	161653.691671	161997.849977	186649.101913
4	1465	212975.595925	211355.267853	212975.923010	188420.472798
...	...	...	...	...	...
1454	2915	69406.717640	70039.512673	69408.818502	97996.127756
1455	2916	67622.632629	67405.637979	67614.469286	87637.213843
1456	2917	133776.822936	133687.335131	133770.086047	142665.129677
1457	2918	98167.360769	99423.964631	98183.617334	121703.474178
1458	2919	192533.426645	193448.531908	192536.860692	220445.929420

1459 rows × 5 columns

```
lasso_train=train_var_fin.mul(lasso_reg.coef_, axis='columns').fillna(0)
lasso_train['intercept']=lasso_reg.intercept_
lasso_train['lasso_model_price']=lasso_train.sum(axis=1)
lasso_train=lasso_train.reset_index()
lasso_train_fin=lasso_train[['Id', 'lasso_model_price']]
lasso_train_fin=lasso_train_fin.set_index(['Id'])
final_train=pd.merge(final_train, lasso_train_fin, right_index=True, left_index=True, how='inner')
# final_train=final_train.reset_index()
save=final_train
display(final_train)
```

	SalePrice	ridge_model_price	linear_model_price	lasso_model_price

1	208500	214797.43788388127	214406.10067660263	214411.971238948
2	181500	215731.41824487317	217751.89047108532	217732.24891400692
3	223500	220143.55405740358	219796.6012088054	219801.62907978118
4	140000	191402.1133883536	191584.24601617083	191584.22104466893
5	250000	316767.9806316928	318026.9493148204	318025.5002663215
6	143000	144782.69406765082	140299.1632785655	140355.826934329
7	307000	259116.1538563643	259481.36658941134	259472.1051557493

Showing the first 1000 rows.

```
elastic_train=train_var_fin.mul(elastic_net.coef_, axis='columns').fillna(0)
elastic_train['intercept']=elastic_net.intercept_
elastic_train['elastic_model_price']=elastic_train.sum(axis=1)
elastic_train=elastic_train.reset_index()
elastic_train_fin=elastic_train[['Id', 'elastic_model_price']]
elastic_train_fin=elastic_train_fin.set_index(['Id'])
final_train=pd.merge(final_train, elastic_train_fin, right_index=True, left_index=True, how='inner')
# final_train=final_train.reset_index()
display(final_train)
```

	Id	SalePrice	ridge_model_price	linear_model_price	lasso_model_price	elastic_model_price
1	1	208500	214797.43788388127	214406.10067660263	214411.971238948	225584.95143811055
2	2	181500	215731.41824487317	217751.89047108532	217732.24891400692	178909.27589976997
3	3	223500	220143.55405740358	219796.6012088054	219801.62907978118	231821.00226445473
4	4	140000	191402.1133883536	191584.24601617083	191584.22104466893	174410.58857925516
5	5	250000	316767.9806316928	318026.9493148204	318025.5002663215	285585.3972052932
6	6	143000	144782.69406765082	140299.1632785655	140355.826934329	171927.2970941502
7	7	307000	259116.1538563643	259481.36658941134	259472.1051557493	251035.7951268186

Showing the first 1000 rows.

```
ridge_mse:  824021563.1103175
linear_mse:  826596480.5097375
lasso_mse:  821371880.6783286
```

Mike Rocchio

MSDS 422

Assignment 2

General Notes – I choose to do my assignment in DataBricks community edition, its community edition has free cloud computing and storage, because I am used to the environment from work. So, a few commands in the notebook may be unfamiliar. I was curious so I decided to apply all 4 regression methods (linear, ridge, lasso, & elastic net). I am really need to do a deep

Data preparation, exploration, visualization – This is where I struggled quite a bit which is surprising because I have a lot of experience with data wrangling in python. I noted in the class discussion that I was having issues aligning columns when multiplying β values to data sets. I had to do a fix because I didn't plan ahead. In the future I am just going to make sure I append the training and data set before I create the dummy variables to ensure that the number of columns aligns between the two sets. I don't have that much professional experience working with categorical data. Most telematics driving variables are numeric.

Review research design and modeling methods – I think that, like a lot of individuals without more intensive math backgrounds, this is where I am going to struggle the most. As said above I coded all 4 methods and that was the easy part but I need to dive further into the math to understand what I am doing. And even understanding that I am finding it really difficult to know what method to use in each case. Perhaps the one with the lowest RMSE. This, I believe is the largest struggle that I will face in this course, I can currently do most of the coding already but I would have very little idea what I am doing when building these models because I lack a deeper understanding of them. I am going to work on that.

Review results, evaluate models – I think what I said above applies to this metric too, I suppose I can evaluate metrics like RMSE to determine model efficacy but I need to dive into the math for each and use

cases for each of the methods. I also tried batch gradient decent on a single variable and I really liked that one and thought it showed promise.

Implementation and programming – The biggest programming takeaway I got from this assignment was to PLAN AHEAD! I had a lot of issues getting the matrices to align when cross multiplying them and ended up coding way more small fixes than I should have needed to.

Exposition, problem description and management recommendations – My final recommendation for this model that I built is to throw it in the trash and start over. I choose a lot of variables and I am skeptical that any of these regression methods are worth using due to mostly poor variable weighting.