Assignment – Advanced Regression

Subjective Questions

Question 1:

What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

Based on the model building exercise in the python notebook, below are the optimal values of alpha for ridge and lasso:

Ridge : 20
Lasso : 0.001

Please see screenshots below:

```
# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    4.1s finished
```

```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 20}
```

```
# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
```

```
model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    3.7s finished
```

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

```
cv_results = pd.DataFrame(model_cv.cv_results_)
```

```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 0.001}
```

```
print(model_cv.best_score_)
```

```
-0.08285521889305081
```

<u>Doubling the value of alpha for Ridge</u>

Current value of alpha for Ridge is : 20

New Value of alpha for Ridge : 40

- The absolute value of the coefficients seems to shrink a bit.
  Earlier values: (Not all values are visible. Please refer to the jupyter notebook for all the values.

```
In [147]: #final ridge model
          alpha = 20
          ridge = Ridge(alpha=alpha)

          ridge.fit(X_train, y_train)
          ridge.coef_

Out[147]: array([-9.29394643e-03,  8.81852327e-04,  1.87381588e-02, -4.13566268e-03,
                  7.11920410e-02,  4.88470514e-02,  5.26065963e-02,  1.94102216e-02,
                  3.08412967e-03,  3.42171728e-02,  7.56667860e-03,  4.28132092e-03,
                  4.32034772e-02,  1.52859718e-02,  3.42258779e-02,  3.58232043e-02,
                  1.40143234e-03,  5.89618790e-02,  1.45243321e-02, -1.66200880e-03,
```

  New Values:

Doubling the value of alpha for Ridge and Lasso

Ridge

```
In [184]: alpha = 40
          ridge = Ridge(alpha=alpha)

          ridge.fit(X_train, y_train)
          ridge.coef_

Out[184]: array([-7.95773223e-03,  1.77943266e-03,  1.88820895e-02, -3.57258515e-03,
                  7.21477766e-02,  4.83476399e-02,  4.76313750e-02,  2.00095396e-02,
                  4.58105291e-03,  3.40048920e-02,  6.56535821e-03,  4.04027008e-03,
                  4.23234723e-02,  1.55890084e-02,  3.41566055e-02,  3.29321907e-02,
                  1.22777623e-03,  5.62862968e-02,  1.48389621e-02, -1.58341576e-03,
```

Metrics:

	Old R2 score	New R2 score	Old RSS Value	New RSS Value	Old RMSE	New RMSE
Train	0.926364778675 5947	0.921349028042 6234	10.53825784283 7492	11.25608380295 4606	0.104882167917 20123	0.108395412985 60584
Test	0.938121325175 959	0.936798460196 487	3.975987961060 694	4.060988088284 584	0.098356172378 3339	0.099401959772 08817

Top 20 Features :

Old Features	**New Features**
OverallQual	1stFlrSF
Neighborhood_Crawfor	BldgType_Twnhs
GrLivArea	Condition1_Norm
Functional_Typ	Condition1_RRAe
YearBuilt	Functional_Maj2
SaleCondition_Normal	Functional_Typ
OverallCond	GrLivArea
Neighborhood_StoneBr	HeatingQC_Fa
Condition1_Norm	HeatingQC_TA
TotalBsmtSF	KitchenQual_Gd
KitchenQual_Gd	Neighborhood_Crawfor
SaleType_WD	Neighborhood_Edwards
Neighborhood_Edwards	Neighborhood_IDOTRR
HeatingQC_Fa	Neighborhood_MeadowV
Neighborhood_MeadowV	OverallCond
HeatingQC_TA	OverallQual
Functional_Maj2	SaleCondition_Normal
Condition1_RRAe	SaleType_WD
BldgType_Twnhs	TotalBsmtSF
Neighborhood_IDOTRR	YearBuilt

Doubling the value of alpha for Lasso

Current value of alpha for Lasso is : 0.001

New Value of alpha for Ridge : 0.002

- The absolute value of the coefficients seems to shrink a bit.
  Earlier values: (Not all values are visible. Please refer to the jupyter notebook for all the values.

```
In [147]: #final ridge model
          alpha = 20
          ridge = Ridge(alpha=alpha)

          ridge.fit(X_train, y_train)
          ridge.coef_

Out[147]: array([-9.29394643e-03,  8.81852327e-04,  1.87381588e-02, -4.13566268e-03,
                  7.11920410e-02,  4.88470514e-02,  5.26065963e-02,  1.94102216e-02,
                  3.08412967e-03,  3.42171728e-02,  7.56667860e-03,  4.28132092e-03,
                  4.32034772e-02,  1.52859718e-02,  3.42258779e-02,  3.58232043e-02,
                  1.40143234e-03,  5.89618790e-02,  1.45243321e-02, -1.66200880e-03,
```

New Values:

Doubling the value of alpha for Ridge and Lasso

Ridge

```
In [184]: alpha = 40
          ridge = Ridge(alpha=alpha)

          ridge.fit(X_train, y_train)
          ridge.coef_

Out[184]: array([-7.95773223e-03,  1.77943266e-03,  1.88820895e-02, -3.57258515e-03,
                  7.21477766e-02,  4.83476399e-02,  4.76313750e-02,  2.00095396e-02,
                  4.58105291e-03,  3.40048920e-02,  6.56535821e-03,  4.04027008e-03,
                  4.23234723e-02,  1.55890084e-02,  3.41566055e-02,  3.29321907e-02,
                  1.22777623e-03,  5.62862968e-02,  1.48389621e-02, -1.58341576e-03,
```

Metrics:

	Old R2 score	New R2 score	Old RSS Value	New RSS Value	Old RMSE	New RMSE
Train	0.918656433146 9806	0.908524774584 4889	11.64143280803 9859	13.09141867603 5651	0.110235242897 44892	0.116898947653 34258
Test	0.939915830116 6365	0.933172110086 1393	3.860682808510 375	4.293997673934 792	0.096919496637 63424	0.102213907590 65437

Total No. of features selected earlier : 75
Total No of features selected now : 51

Top 20 Features :

Old Features	New Features
GrLivArea	GrLivArea
OverallQual	OverallQual
SaleType_New	SaleType_New
Neighborhood_Crawfor	YearBuilt
YearBuilt	OverallCond
Functional_Typ	Functional_Typ
MSZoning_FV	TotalBsmtSF
SaleCondition_Normal	BsmtFinSF1
OverallCond	Foundation_PConc
MSZoning_RL	Condition1_Norm
BsmtExposure_No	KitchenQual_TA
KitchenQual_TA	LotConfig_Inside
KitchenAbvGr	SaleType_WD

KitchenQual_Gd	MSSubClass
Fence_GdWo	LotShape_Reg
Neighborhood_IDOTRR	BldgType_Twnhs
SaleType_WD	KitchenAbvGr
HeatingQC_Fa	BsmtExposure_No
HeatingQC_TA	HeatingQC_TA
BldgType_Twnhs	MSZoning_RM

Thus, the most important predictor variables are:

- BldgType_Twnhs : Type of dwelling, Townhouse
- Condition1_Norm : Proximity to various conditions, Normal
- Functional_Typ : Home functionality, Typical Functionality
- GrLivArea : Above grade (ground) living area square feet
- HeatingQC_TA : Heating quality and condition,Average/Typical
- KitchenQual_Gd : Kitchen quality
- OverallCond : Overall condition of the house
- OverallQual : Overall material and finish of the house
- SaleType_WD : Type of sale, Warranty Deed - Conventional
- TotalBsmtSF : Total square feet of basement area
- YearBuilt : Original construction date

In general,

- If alpha is higher, coefficient values will be lower.
- High alpha means high bias and low variance
- In Ridge, for higher alpha, model coefficients will reduce.
- In Lasso, for higher alpha, some of the coefficient values may change to 0 i.e. No of features might decrease.

Question 2:

You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

Answer:

Although, the r2 score for both the models are comparable, I will choose Lasso as the data has a lot of features and Lasso helps in reducing the no of features making the model more robust.

Question 3

After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

Answer:

Five most important predictor variables in Lasso, earlier, were:

- GrLivArea
- OverallQual
- SaleType_New
- Neighborhood_Crawfor
- YearBuilt

After dropping the above features, the 5 most important predictor variables now are :

- 1stFlrSF
- 2ndFlrSF
- Functional_Typ
- Neighborhood_StoneBr
- MSZoning_FV

Below are few of the code snippets from the Jupyter notebook for dropping the 5 features and rebuilding the model.

```
In [201]: ## Top 5 features to be removed
          remove_features = ['GrLivArea', 'OverallQual', 'SaleType_New', 'Neighborhood_Crawfor', 'YearBuilt']
```

```
In [206]: X_train_new = X_train.drop(remove_features, axis=1)
          X_test_new = X_test.drop(remove_features, axis=1)
          X_train_new.columns

Out[206]: Index(['MSSubClass', 'LotFrontage', 'LotArea', 'Street', 'OverallCond',
                 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
                 ...
                 'SaleType_ConLD', 'SaleType_ConLI', 'SaleType_ConLw', 'SaleType_Oth',
                 'SaleType_WD', 'SaleCondition_AdjLand', 'SaleCondition_Alloca',
                 'SaleCondition_Family', 'SaleCondition_Normal',
                 'SaleCondition_Partial'],
                dtype='object', length=243)
```

```
In [207]: params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
           0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
           4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}
```

```
In [208]: lasso = Lasso()
```

In [209]:
```
# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
```

In [210]:
```
model_cv.fit(X_train_new, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    3.5s finished
```

Out[210]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [212]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)


print(model_cv.best_score_)
```

```
{'alpha': 0.001}
-0.09068262727414145
```

In [213]:
```
alpha = 0.001

lasso = Lasso(alpha=alpha)
lasso.fit(X_train_new, y_train)
```

Out[213]:
```
Lasso(alpha=0.001)
```

Top 5 features selected by the new Lasso Model are :

In [223]:
```
main_coef.sort_values(ascending=False)[:5]
```

Out[223]:
```
1stFlrSF                0.095407
2ndFlrSF                0.094301
Functional_Typ          0.090831
Neighborhood_StoneBr    0.084677
MSZoning_FV             0.073358
dtype: float64
```

Question 4

How can you make sure that a model is robust and generalisable? What are the implications of the same for the accuracy of the model and why?

Answer:

- A generalised model would ideally mean that there isn't a huge difference in its performance between training set and the testing set.
- Simpler models are more robust and generalisable.
- Complex models tend to change significantly with slight changes in the training data. Complex models lead to overfitting.
- Simpler models have low variance, high bias, whereas, complex models have high variance and low bias.

- If a model is not robust, it can't predict well on unseen data.
- To make a model robust and generalisable, regularization can be used.
    - We need to make sure that the model doesn't overfit. An overfit model will tend to memorize the training data but fail to pick up patterns on an unseen data set.
    - It aims to reduce the model complexity while ensuring that the model doesn't become too simple.
    - It involves adding a penalty term (regularization) to the cost by adding the absolute values (Lasso) or the squares of the parameters of the model (Ridge).
    - It essentially shrinks the model coefficients, thus discouraging the model to overfit /become too complex.
    - It compromises bias for low variance.

- **Implications on Accuracy**
    - A highly complex model will have high accuracy on the training set.
    - A robust model will have low variance and some bias. Addition of bias causes the accuracy to decrease

<u>Bias – Variance trade off:</u>