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:

- Optimal value of alpha for lasso regression = 0.0001
- Optimal value of alpha for ridge regression = 2.0

Post the change the optimal alpha value we see that the minor change in R2 values across train and test sets. **Bias** seems to have been introduced in the model.

Also with the increase in the alpha values the coefficients values have gone down (minimise the penalty term). Coefficients (not all) Pre and Post change in alpha has been indicated in the snapshot below.

Ridge with alpha 2.0		Alpha = 4.0	
GrLivArea	**0.147682**	GrLivArea	0.146753
MSZoning_RL	**0.146303**	MSZoning_RL	0.102913
MSZoning_FV	**0.111906**	OverallQual	0.085164
MSZoning_RH	**0.107059**	Functional_Typ	0.077964
LandContour_HLS	**0.089186**	LandContour_HLS	0.077056
OverallQual	**0.082438**	MSZoning_FV	0.069427
Functional_Typ	**0.081865**	Neighborhood_Crawfor	0.066506
LandContour_Low	**0.077334**	LandContour_Low	0.062207
LandContour_Lvl	**0.069962**	MSZoning_RH	0.060779
SaleCondition_Others	0.069751	LandContour_Lvl	0.060176
Neighborhood_Crawfor	0.065177	SaleCondition_Others	0.060009
Condition1_Norm	0.057439	Condition1_Norm	0.057482
SaleCondition_Normal	0.055608	BsmtExposure_Gd	0.052927
OverallCond	0.052608	SaleCondition_Normal	0.052913
BsmtExposure_Gd	0.052207	OverallCond	0.052534
LotArea	0.049149	LotArea	0.049176
MSZoning_RM	**0.048239**	GarageCars	0.048141
ExterQual_Others	0.047982	BsmtFinSF1	0.046996
GarageCars	0.047320	SaleCondition_Partial	0.045816
SaleCondition_Partial	0.047293	ExterQual_Others	0.037961
BsmtFinSF1	0.046570	HouseStyle_Others	0.028296
HouseStyle_Others	0.033130	BsmtFinType2_Unf	0.015713
BsmtFinType2_Unf	0.016234	BsmtFinSF2	0.007622
BsmtFinSF2	0.007843	MSZoning_RM	0.007390
BsmtFinType1_Others	-0.018687	BsmtExposure_Others	-0.018105

Lasso - Alpha .0001		Alpha .0002	
MSZoning_RL	**2.442987e-01**	GrLivArea	0.147614
MSZoning_FV	**2.120457e-01**	MSZoning_RL	0.110788
MSZoning_RH	**2.097999e-01**	OverallQual	0.084032
GrLivArea	**1.485291e-01**	LandContour_HLS	0.081300
MSZoning_RM	**1.446369e-01**	Functional_Typ	0.080287
LandContour_HLS	**9.624016e-02**	MSZoning_FV	0.077810
LandContour_Low	**8.793387e-02**	MSZoning_RH	0.069328
Functional_Typ	**8.452235e-02**	LandContour_Low	0.066644
OverallQual	**8.113511e-02**	Neighborhood_Crawfor	0.064388
LandContour_Lvl	**7.442868e-02**	SaleCondition_Others	0.061112
SaleCondition_Others	7.019293e-02	LandContour_Lvl	0.060917
Neighborhood_Crawfor	6.213792e-02	Condition1_Norm	0.056773
Condition1_Norm	5.713134e-02	BsmtExposure_Gd	0.052938
SaleCondition_Normal	5.324238e-02	OverallCond	0.052533
ExterQual_Others	5.232562e-02	SaleCondition_Normal	0.051399
OverallCond	5.229005e-02	LotArea	0.048974
BsmtExposure_Gd	5.128783e-02	GarageCars	0.047431
LotArea	4.900915e-02	BsmtFinSF1	0.046705
GarageCars	4.665842e-02	SaleCondition_Partial	0.042757
BsmtFinSF1	4.650181e-02	ExterQual_Others	0.029258
SaleCondition_Partial	4.455439e-02	HouseStyle_Others	0.026814
HouseStyle_Others	3.476000e-02	MSZoning_RM	0.013594
BsmtFinSF2	2.529254e-03	BsmtFinSF2	0.002538
BsmtFinType2_Unf	0.000000e+00	BsmtFinType1_Others	-0.000000
BsmtFinType1_Others	-1.777331e-17	HeatingQC_Others	-0.000000

- **Post change to alpha it seems GrLivArea to be the top predictor emerging from both Lasso and Ridge regression models followed by**
- **MSZoning_***
- **LandContours_***

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?

Ans : I will go with Lasso regression, as with lasso the penalty has pushed many of the coefficients towards 0. The model explanation with lesser variables is relatively easy, also the model complexity is reduced with lesser number of predictors.

```
lasso = Lasso(alpha=alpha)

lasso.fit(X_train_lasso, y_train_lasso)

lasso.coef_
```

```
Out[99]: array([ 0.04897407,  0.08403186,  0.05253312,  0.04670542,  0.00253812,
                 0.14761427,  0.04743112, -0.10236951,  0.07780969,  0.06932805,
                 0.11078849,  0.01359386, -0.13554365,  0.08130014,  0.06664393,
                 0.06091669, -0.04648949,  0.06438756, -0.10090442, -0.08491732,
                -0.07735024, -0.04718375, -0.06934033, -0.03307014, -0.05661515,
                 0.05677319, -0.08075   , -0.06474999,  0.02681397, -0.04917613,
                -0.08803867, -0.07195137, -0.11065449, -0.08281804, -0.08442949,
                -0.08300998,  0.02925759, -0.07973906, -0.0842672 , -0.09779472,
                 0.05293785, -0.04007503, -0.        ,  0.        , -0.04878297,
                -0.        ,  0.08028704,  0.05139893,  0.06111229,  0.04275651])
```

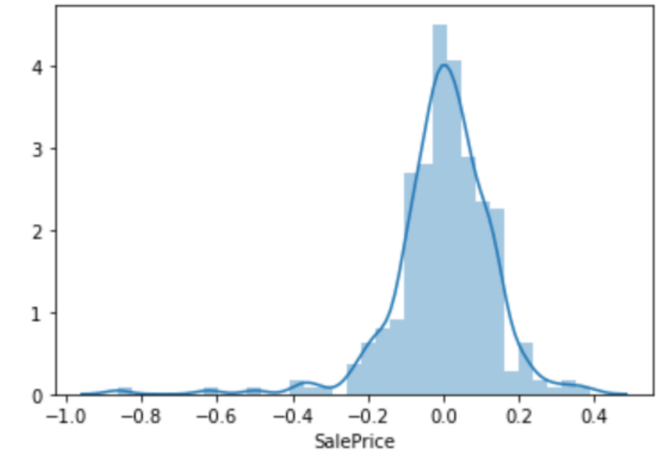
```
In [101]: # Lets calculate revised metrics e.g. R2 score, RSS and RMSE with new aplha
```

Identifying and Applying the Lambda (alpha) value is with trial and error which results in optimal regression matrices specially RMSE and **which helps ensure all the regression assumptions are held intact or met i**.e. Normality of residual teams, Homoscedasticity etc..

In this case we have chosen the value of alpha = 0.0001. Although the value is low however we could see the model performance is at 90%+ across both train and test sets.

The residual are normally distributed and they are randomly distributed. Variance in the error terms / residuals do not exhibit any patterns. The variance between the error terms is Homoscedastic.

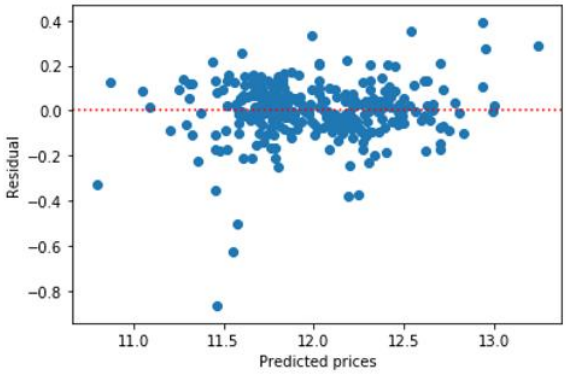
```
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x23232c
```



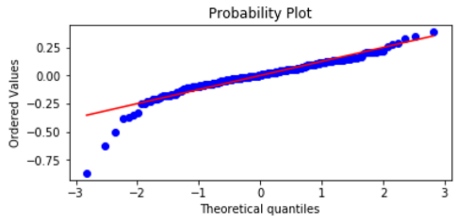
Error terms are normally distributed

Homoscedasticity

```
In [74]: plt.scatter(y_pred_test , residual)
         plt.axhline(y=0, color='r', linestyle=':')
         plt.xlabel("Predicted prices")
         plt.ylabel("Residual")
         plt.show()
```




```
In [72]: import scipy as sp
         fig, ax = plt.subplots(figsize=(6,2.5))
         _, (__, ___, r) = sp.stats.probplot(residual, plot=ax, fit=True)
```



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?

Ans:

Top Predictors with original dataset include following predictors:

MSZoning: Identifies the general zoning classification of the sale.

```
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
```

GrLivArea: Above grade (ground) living area square feet

1	Lasso - Alpha .0001	
2	MSZoning_RL	2.442987e-01
3	MSZoning_FV	2.120457e-01
4	MSZoning_RH	2.097999e-01
5	GrLivArea	1.485291e-01
6	MSZoning_RM	1.446369e-01

We will drop these two fields from the predictor variable list and then rebuild the model. With the new model the top 4-5 predictors are

- **OverallQual**
- **Neighborhood_Crawfor**
- **LotArea**
- **GarageCars**

Snapshot of 20+ top predictors in order is given below.

```
In [122]: betas_['Lasso'].sort_values(ascending=False)

Out[122]: OverallQual              1.655414e-01
          Neighborhood_Crawfor     1.375706e-01
          LotArea                  1.045905e-01
          GarageCars               7.431506e-02
          SaleCondition_Others     5.479561e-02
          SaleCondition_Normal     5.024926e-02
          BsmtFinSF1               4.396939e-02
          LandContour_Lvl          4.101181e-02
          OverallCond              3.914258e-02
          SaleCondition_Partial    3.869027e-02
          Condition1_Norm          3.611998e-02
          BsmtExposure_Gd          2.861821e-02
          LandContour_HLS          1.155251e-02
          ExterQual_Others         8.851509e-03
          LandContour_Low          5.551632e-03
          BsmtFinSF2               3.530747e-03
          BldgType_Duplex          2.115485e-03
          Functional_Typ           0.000000e+00
          BsmtFinType2_Unf         0.000000e+00
          BsmtFinType1_Others     -1.523426e-18
          BsmtExposure_Others     -1.062618e-02
          BldgType_Others         -1.484154e-02
          HouseStyle_Others       -2.858513e-02
          Neighborhood_NWAmes     -4.372754e-02
          Neighborhood_Gilbert    -5.167627e-02
          Neighborhood_NAmes      -5.837714e-02
          Neighborhood_SawyerW    -6.248242e-02
          Neighborhood_Sawyer     -6.458313e-02
          Neighborhood_CollgCr    -6.733718e-02
          Exterior1st_CemntBd     -7.103719e-02
          Age_Build               -7.417587e-02
          BsmtQual_Gd             -8.404548e-02
```

Question 4

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

Model should be is robust and generalizable so that it is not impacted by the variance in the data wrt outliers and ranges in the training data. It’s important because the test accuracy can’t be drastically compromised as against the train accuracy. This leads to models being overfitting on Train set but under fitting on the test set. Overall model is over fitted and can't be reliably used for unseen data. We adopt multiple strategies to make the model robust and generalizable:

- Outlier treatment :
    - The outlier analysis needs to be done and only those which are relevant to the dataset need to be retained i.e which make business sense. Those outliers which it does not make sense to keep must be removed from the dataset to ensure data stability.
- Data Transformation :
    - Transform the data (e.g. Log, inverse, sq , sqrt etc to ensure that the data can be transformed to follow a pattern.
- Scaling and standardisation: Same unit across all numeric variables helps model perform better and also standardisation helps make the variable more Gaussian which is what the most of the model expect and perform better on.
- Data imbalance – this is one issues which we cannot remediate easily (unless we manufacture data!) hence splitting into test and train can be avoided and we can use GRIDCV approach to use the dataset randomly multi-fold to as training data.
- Predictor Variability: Remove those variables which have near zero variability and doesn't have much implications on decision making.

Too much weightage should not give to the predictors with outliers so that the accuracy predicted by the model is high

This has to be noted that we use to train data to fit once and only do transform on the train data (we don't fit again)