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?

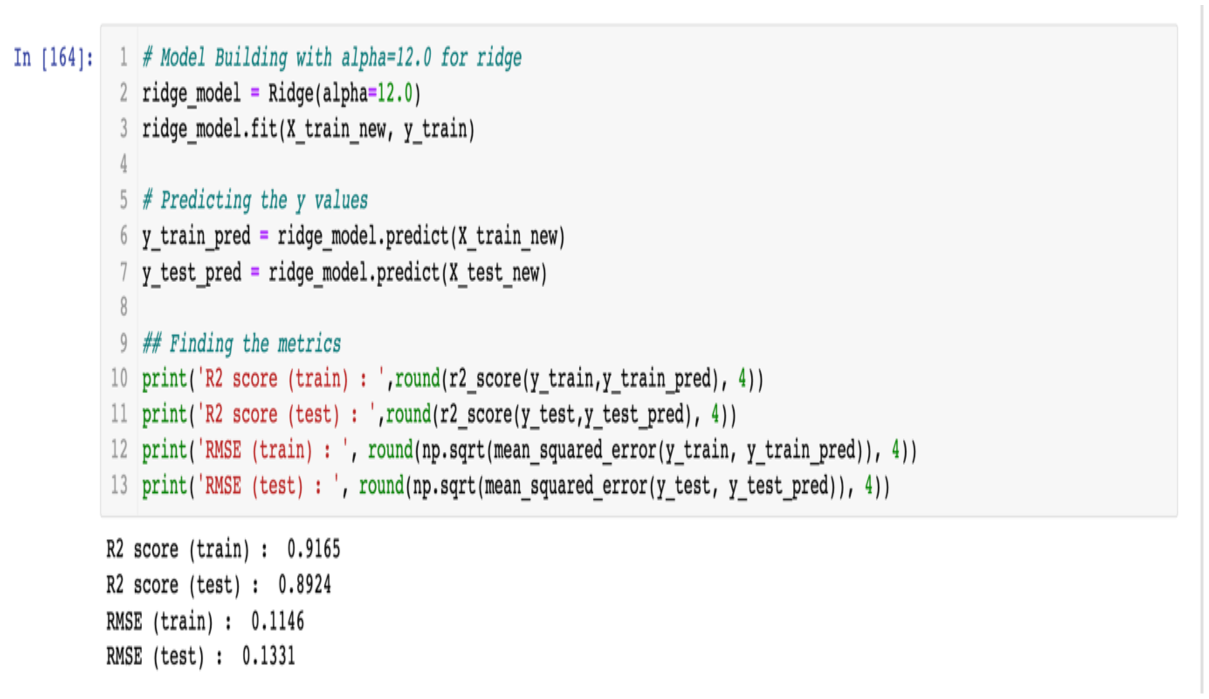
Answer

After doing our analysis on the provided dataset we found the following optimal value of alpha for ridge and lasso regression is as follows

- Ridge :- 6.0
- Lasso :- 0.0002

Once we double the value of alpha for both ridge and lasso regression i.e

- Ridge :- 12.0

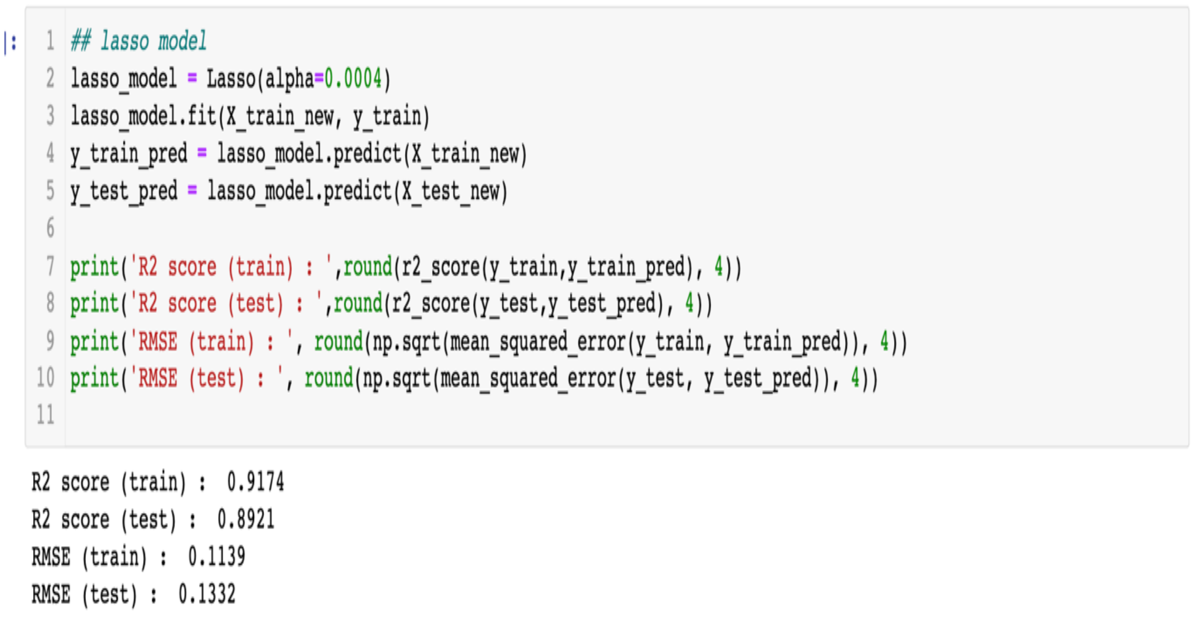


```
In [164]:
# Model Building with alpha=12.0 for ridge
ridge_model = Ridge(alpha=12.0)
ridge_model.fit(X_train_new, y_train)

# Predicting the y values
y_train_pred = ridge_model.predict(X_train_new)
y_test_pred = ridge_model.predict(X_test_new)

## Finding the metrics
print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))
```

```
R2 score (train) :  0.9165
R2 score (test) :  0.8924
RMSE (train) :  0.1146
RMSE (test) :  0.1331
```

We can see that the value of R2 for train is high and test is low, which clearly shows a case of overfitting

- Lasso :- 0.0004

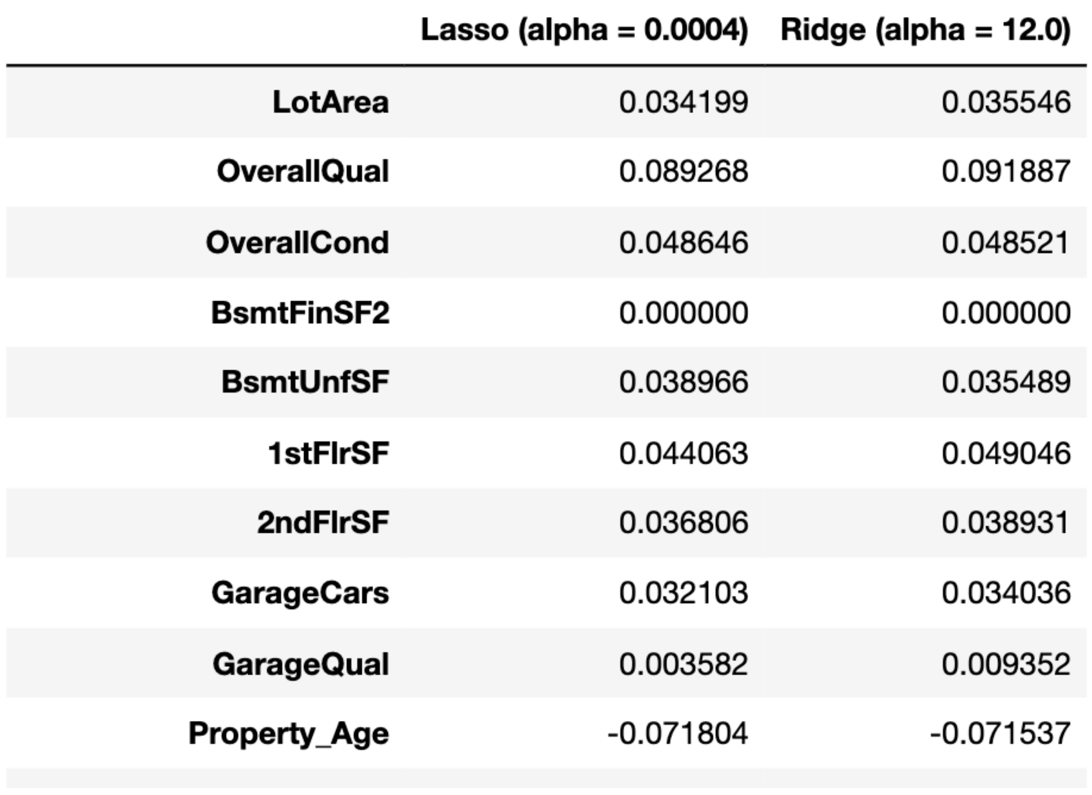


```python
## lasso model
lasso_model = Lasso(alpha=0.0004)
lasso_model.fit(X_train_new, y_train)
y_train_pred = lasso_model.predict(X_train_new)
y_test_pred = lasso_model.predict(X_test_new)

print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))

```

```
R2 score (train) :  0.9174
R2 score (test) :  0.8921
RMSE (train) :  0.1139
RMSE (test) :  0.1332
```

We also see the similar behaviour of overfitting with lasso regression.

After we make the change, the top 10 predictor variables are as follows



	Lasso (alpha = 0.0004)	Ridge (alpha = 12.0)
LotArea	0.034199	0.035546
OverallQual	0.089268	0.091887
OverallCond	0.048646	0.048521
BsmtFinSF2	0.000000	0.000000
BsmtUnfSF	0.038966	0.035489
1stFlrSF	0.044063	0.049046
2ndFlrSF	0.036806	0.038931
GarageCars	0.032103	0.034036
GarageQual	0.003582	0.009352
Property_Age	-0.071804	-0.071537

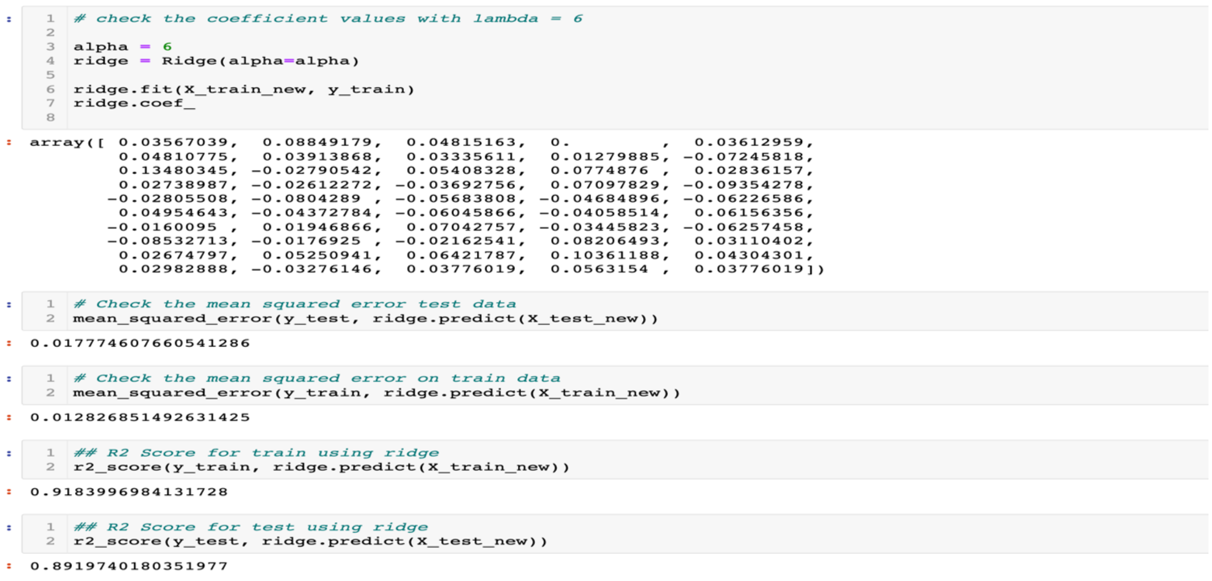
Question 2

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 :-

For alpha=6.0 in ridge regression

- R2_Score on train set (0.8919)
- Mean_Squared_error on train set (0.0177)



```
# check the coefficient values with lambda = 6

alpha = 6
ridge = Ridge(alpha=alpha)

ridge.fit(X_train_new, y_train)
ridge.coef_

```

```
array([ 0.03567039,  0.08849179,  0.04815163,  0.        ,  0.03612959,
        0.04810775,  0.03913868,  0.03335611,  0.01279885, -0.07245818,
        0.13480345, -0.02790542,  0.05408328,  0.0774876 ,  0.02836157,
        0.02738987, -0.02612272, -0.03692756,  0.07097829, -0.09354278,
       -0.02805508, -0.0804289 , -0.05683808, -0.04684896, -0.06226586,
        0.04954643, -0.04372784, -0.06045866, -0.04058514,  0.06156356,
       -0.0160095 ,  0.01946866,  0.07042757, -0.03445823, -0.06257458,
       -0.08532713, -0.0176925 , -0.02162541,  0.08206493,  0.03110402,
        0.02674797,  0.05250941,  0.06421787,  0.10361188,  0.04304301,
        0.02982888, -0.03276146,  0.03776019,  0.0563154 ,  0.03776019])
```

```
# Check the mean squared error test data
mean_squared_error(y_test, ridge.predict(X_test_new))
```

```
0.017774607660541286
```

```
# Check the mean squared error on train data
mean_squared_error(y_train, ridge.predict(X_train_new))
```

```
0.012826851492631425
```

```
## R2 Score for train using ridge
r2_score(y_train, ridge.predict(X_train_new))
```

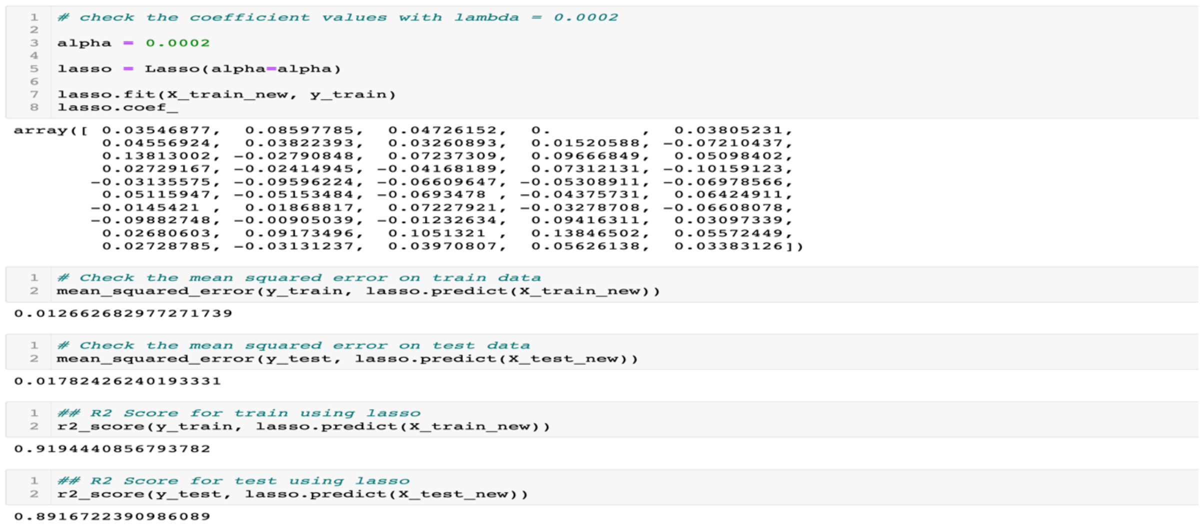
```
0.9183996984131728
```

```
## R2 Score for test using ridge
r2_score(y_test, ridge.predict(X_test_new))
```

```
0.8919740180351977
```

For lasso=0.0002 in lasso regression

- R2_Score on train set (0.8916)
- Mean_Squared_error on train set (0.0178)



```
# check the coefficient values with lambda = 0.0002

alpha = 0.0002

lasso = Lasso(alpha=alpha)

lasso.fit(X_train_new, y_train)
lasso.coef_
```

```
array([ 0.03546877,  0.08597785,  0.04726152,  0.        ,  0.03805231,
        0.04556924,  0.03822393,  0.03260893,  0.01520588, -0.07210437,
        0.13813002, -0.02790848,  0.07237309,  0.09666849,  0.05098402,
        0.02729167, -0.02414945, -0.04168189,  0.07312131, -0.10159123,
       -0.03135575, -0.09596224, -0.06609647, -0.05308911, -0.06978566,
        0.05115947, -0.05153484, -0.0693478 , -0.04375731,  0.06424911,
       -0.0145421 ,  0.01868817,  0.07227921, -0.03278708, -0.06608078,
       -0.09882748, -0.00905039, -0.01232634,  0.09416311,  0.03097339,
        0.02680603,  0.09173496,  0.1051321 ,  0.13846502,  0.05572449,
        0.02728785, -0.03131237,  0.03970807,  0.05626138,  0.03383126])
```

```
# Check the mean squared error on train data
mean_squared_error(y_train, lasso.predict(X_train_new))
```

```
0.012662682977271739
```

```
# Check the mean squared error on test data
mean_squared_error(y_test, lasso.predict(X_test_new))
```

```
0.01782426240193331
```

```
## R2 Score for train using lasso
r2_score(y_train, lasso.predict(X_train_new))
```

```
0.9194440856793782
```

```
## R2 Score for test using lasso
r2_score(y_test, lasso.predict(X_test_new))
```

```
0.8916722390986089
```

Hence, I would go ahead with using alpha for ridge regression as we clearly see that the R2_Score is better in Ridge regression over the test data set and the mean squared error is also less on the test data set.

Question 3

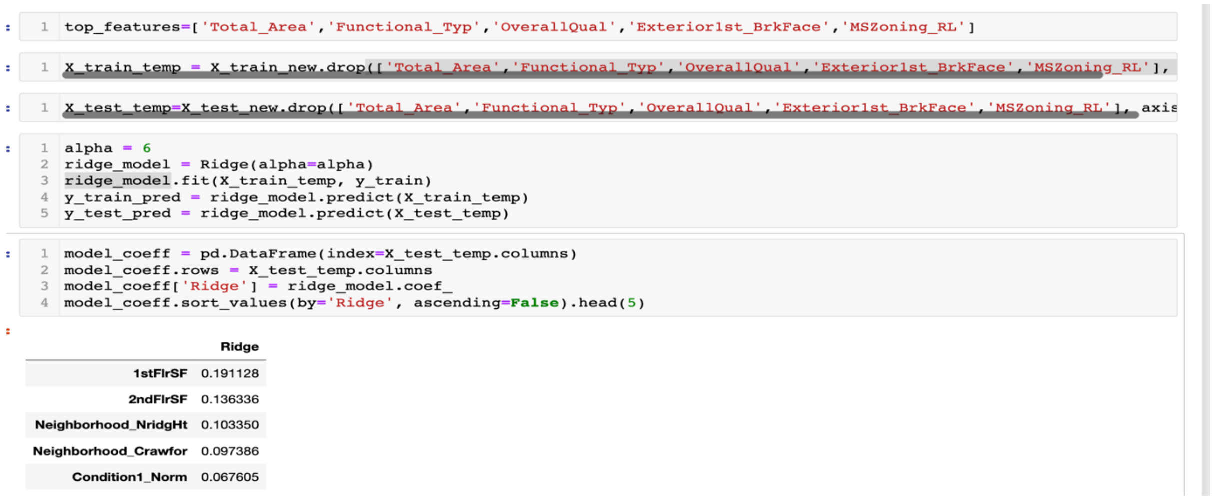
After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

Answer :-

Since we are using Ridge regression as our final model and looking at the coefficient we see that the top 5 features are as follows

- Total_Area
- Functional_Typ
- OverallQual
- Exterior1st_BrkFace
- MSZoning_RL

Since, the question says that this is not available anymore, hence we will create a new dataframe by dropping these features, once we drop these features and will do another round of model building with ridge regression using alpha=6.0



```
top_features=['Total_Area','Functional_Typ','OverallQual','Exterior1st_BrkFace','MSZoning_RL']
```

```
X_train_temp = X_train_new.drop(['Total_Area','Functional_Typ','OverallQual','Exterior1st_BrkFace','MSZoning_RL'],
```

```
X_test_temp=X_test_new.drop(['Total_Area','Functional_Typ','OverallQual','Exterior1st_BrkFace','MSZoning_RL'], axis
```

```
alpha = 6
ridge_model = Ridge(alpha=alpha)
ridge_model.fit(X_train_temp, y_train)
y_train_pred = ridge_model.predict(X_train_temp)
y_test_pred = ridge_model.predict(X_test_temp)
```

```
model_coeff = pd.DataFrame(index=X_test_temp.columns)
model_coeff.rows = X_test_temp.columns
model_coeff['Ridge'] = ridge_model.coef_
model_coeff.sort_values(by='Ridge', ascending=False).head(5)
```

	Ridge
1stFlrSF	0.191128
2ndFlrSF	0.136336
Neighborhood_NridgHt	0.103350
Neighborhood_Crawfor	0.097386
Condition1_Norm	0.067605

Now, the top features are as follows

- 1stFlrSF
- 2ndFlrSF
- Neighborhood_NridgHt
- Neighborhood_Crawfor
- Condition1_Norm

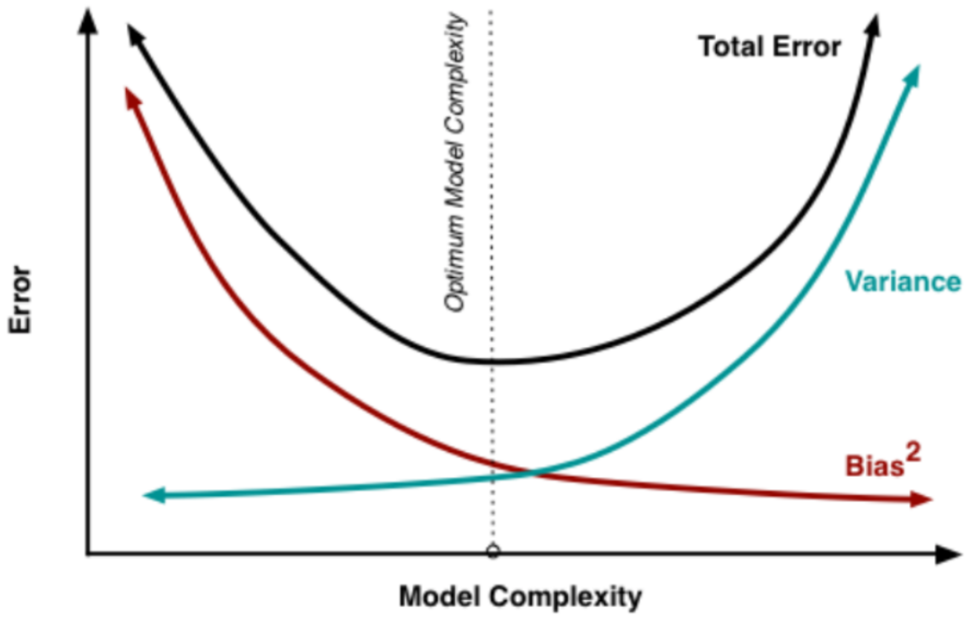
Question 4

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 model to be generalisable has to be simple, and it will be robust if a slight change in the underlying data does not changes the model a lot.

A simple and generalisable model may not have the best accuracy compared to a complex model. But the drawback is that a complex model may lead to a situation of overfitting i.e (high train accuracy and low test accuracy)



As you can see that as the model complexity increases the bias goes down but the variance becomes high (small change in data can cause model to behave weird)

A robust generalisable model will have similar sort of accuracy on both training and test dataset. While a non generalisable model works only with a certain kind of data (accuracy is high for certain dataset but not for all)

A more robust and generalisable model is one that has low bias and low variance. However making model simpler leads to tradeoff between Bias and Variance, that is when Bias increases Variance decreases and vice versa.

Bias is error in model, when bias is low that means the model is not able to learn efficiently from data (underfitting). And when variance is high that means for a slight change in the underlying data model does not predicts correctly.

Regularization is a technique that can be used to make simpler models. Regularization technique like lasso/ridge penalise complex models thus ensuring that the models are simpler and robust.