QF634 APPLIED QUANTITATIVE RESEARCH METHODS
LECTURE 2

Lecturer: Prof Emeritus Lim Kian Guan

Regularizations

Ridge and Lasso Regressions

ML Approach to Prediction

- The data set is divided into a training data set (usually involving 50-60% of the data), a validation data set (usually involving 20-30% of the data), and a test data set (remaining part of the data set). The test data set is also called the hold-out data set.
- Internal Model Parameters – the ML algo must fix optimally, given the hyperparameters, that will minimize the training errors (error between predicted target and actual target in training set) according to error criterion, e.g., MMSE, Min Cross Entropy.
- Hyperparameters - are exogenously specified and they control the estimation or fitting process and affects the parameter estimation and model fitting.
- Different hyperparameter(s) are chosen and each set of the hyperparameter values is used together with the training data set to find the optimized/fitted model. This fitted model (using training data set) is then applied to the validation data set features to obtain predicted targets. These predicted targets are then compared with the actual validation set targets to derive the accuracy measure. Optimal hyperparameters are found by maximizing this prediction accuracy measure.
- If the training set performance and validation set performance are similarly good, then one may proceed to employing the fitted model on the test data set. Validation also acts as a robustness check.

ML Approach to Prediction

- In this lecture, we show the use of validation in a multiple linear regression model with regularization. Regularization is a method to constrain a model to fit out-of-sample (validation set) more accurately. Without regularization, the trained model may overfit and this can lead to underfitting or poor prediction performance out-of-sample.
- In linear regression, regularization takes the form of introducing constraints on the estimated coefficients to ensure they are not too large due to some significant impact of some features that may not occur uniformly or commonly across training and validation sample sets.
- Regularization is also useful in improving out-of-sample prediction when the model is complex and there are too many features against a finite sample size or where there are high feature correlations that may sometimes produce higher prediction errors.
- Hyperparameters are used in regression with regularization.

Regression and Regularization

- Let Y_k be the kth target variable observed and $(X_{1k}, X_{2k}, X_{3k}, \ldots, X_{pk})$ be the set of p features related to the kth target variable. The linear regression is written as

$$Y_k = b_0 + b_1 X_{1k} + b_2 X_{2k} + b_3 X_{3k} + \ldots\ldots + b_p X_{pk} + \varepsilon_k \qquad (2.1)$$

  where ε_k is a residual variable that is unobserved.

- Suppose the regression model or algorithm has hyperparameters. The training data can be used firstly to check if the predictive model or algorithm is feasible, i.e., the in-sample fitting error is not too large.

- If feasible, the model should next be checked for consistent performance as well as improved performance by optimal choice (fine-tuning) of the hyperparameters. This is done by a separate validation data set. The validation data set prediction performance should return consistent performance of similar accuracy as in the training data. An inconsistent performance could be over-fitting or very high prediction accuracy in the training data set but under-fitting or high prediction error in the validation data set. If tuning the hyperparameters cannot eliminate the situation of high prediction error in the validation data set, then the predictive model could be problematic.

Regression and Regularization

- Unlike classical setup in linear regression where the added residual noise or innovation may take specific distributional assumptions to enable testing, the residual noise in the machine learning method takes a low profile.
- The linear regression under supervised machine learning in Eq. (2.1) typically aims to find optimal parameter (estimates) $\hat{b}_0, \hat{b}_1, \hat{b}_2, \ldots, \hat{b}_p$ so that a loss criterion such as sum of squared errors, $\sum_{k=1}^{0.8n}\left(Y_k - \hat{Y}_k\right)^2$ is minimized, where fitted $\hat{Y}_k = \hat{b}_0 + \hat{b}_1 X_{1k} + \hat{b}_2 X_{2k} + \cdots + \hat{b}_p X_{pk}$. 0.8n is 80% sample as training set.
- R^2-score for the training data set is $1 - \sum_{k=1}^{0.8n}\left(Y_k - \hat{Y}_k\right)^2 / \sum_{k=1}^{0.8n}(Y_k - \overline{Y}_k)^2$. The second term is sum of squared errors over total sum of squares in standard least squares regression terminology, and R^2 lies between 0 and +1.

Regression and Regularization

- When the trained or fitted coefficients $\hat{b}_0$, $\hat{b}_1$, $\hat{b}_2$, …., $\hat{b}_p$ are employed in the prediction of the validation data set of 0.2n number of the remaining Y_k's, using the associated validation set data of X_{jk}'s (for j =1,2,…,p), the prediction yields $Y_k^* = \hat{b}_0 + \hat{b}_1 X_{1k} + \hat{b}_2 X_{2k} + \cdots + \hat{b}_p X_{pk}$.
- The validation score is $R^2 = 1 - \sum_{k=1}^{0.2n}(Y_k - Y_k^*)^2 / \sum_{k=1}^{0.2n}(Y_k - \overline{Y}_k)^2$. In this case, the score R^2 **may be sometimes negative** if the actual model of the test data is different so that $\sum_{k=1}^{0.2n}(Y_k - Y_k^*)^2$ is very large.
- The R^2 score in the validation set is a useful measure of accuracy of the machine learning method when coefficients (or model parameters) trained by a training data set is applied to new data, assuming both data sets come from the same data generating model.
- A somewhat similar though not identical metric in measuring prediction performance is the root mean square error (RMSE) which is $\sqrt{\sum_{k=1}^{0.2N}\left(Y_k - Y_k^*\right)^2/(0.2N)}$.

Regression and Regularization

- The validation performance will not be good if the assumption that both data sets come from the same data generating model is incorrect. The use of training and validation data sets is akin to the idea of in-sample and out-of-sample fits. The above comments on the validation data set apply similarly to use of trained or fitted model in prediction in the test data set.
- If $\{Y_k, X_{1k}, X_{2k}, X_{3k}, \ldots, X_{pk}\}_{k=1,2,\ldots.n}$ is not a time series, but a cross-section, then splitting the set into training and testing subsets can be random.
- If it is a time series, then typically the training set data would precede those of the test set. Moreover, in the strict sense of prediction based on available information, the features would have time stamps prior to that of the target variable.
- If a training set R^2 or accuracy of fit by the model is high and if a validation set R^2 score is also high, then the fitted model with its fine-tuned hyperparameters is used to check out the test data set. If the test R^2 score is also high or having accurate prediction based on the trained or fitted coefficients, then the model with its trained coefficients may be usable for future prediction of the target variable when a new set of features arrives.

Problem of Overfitting and Underfitting

Regressions based on available features can run into overfitting problem **when too many features (or explanatory variables) are utilized**. This may produce good fit, e.g., high R^2 in the training data set, but the fitted coefficients may in turn produce inaccurate predictions using the validation and the test data sets.

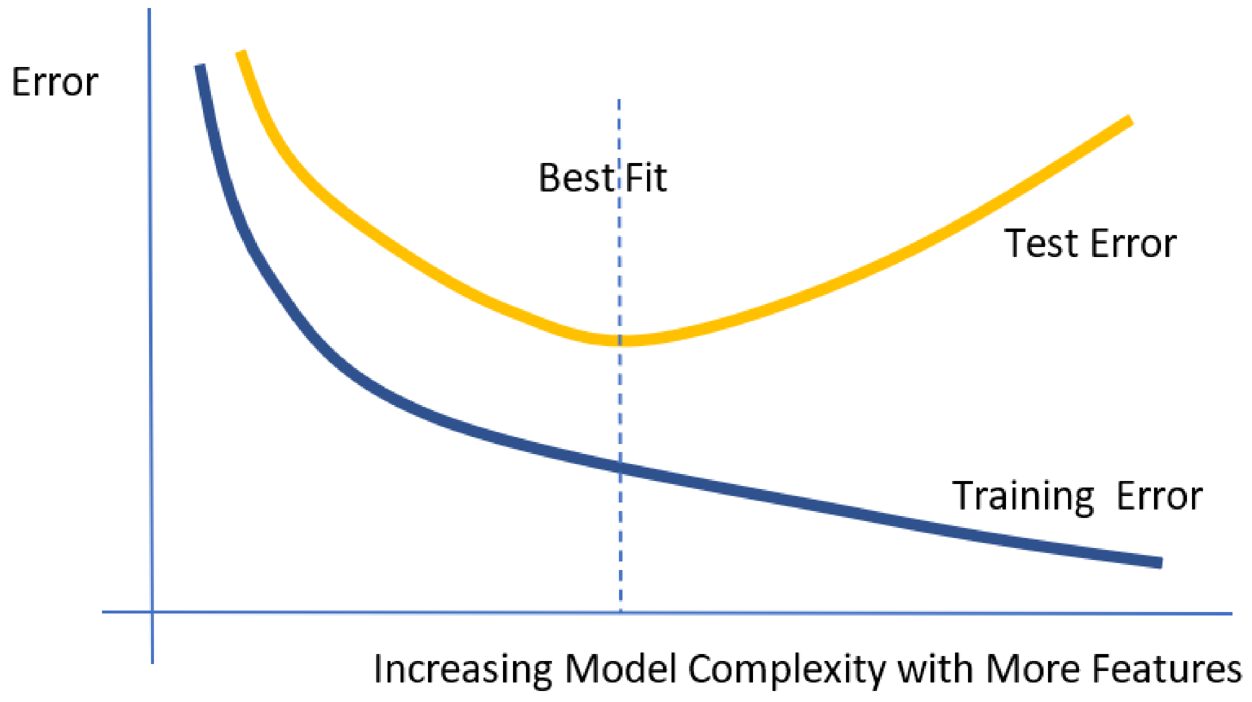
- Consider the multivariate regression (2.1) where we re-write in short-form its predicted value as $\hat{Y}_k = \sum_j \hat{b}_j X_{jk}$. The sum of squared prediction errors is then $\Sigma_k \left(Y_k - \Sigma_j \hat{b}_j X_{jk}\right)^2$. Suppose an extra feature Z_k for instance or sample point k is added. Then the sum of squared prediction errors is now $\Sigma_k \left(Y_k - \Sigma_j \hat{b}_j X_{jk} - \hat{b}_{p+1} Z_k\right)^2$. Clearly this latter SSE is smaller, i.e., $\Sigma_k \left(Y_k - \Sigma_j \hat{b}_j X_{jk} - \hat{b}_{p+1} Z_k\right)^2 \leq \Sigma_k \left(Y_k - \Sigma_j \hat{b}_j X_{jk}\right)^2$. Hence unnecessarily increasing the number of features that do not actually affect the target can produce overfitting (higher training data set R^2). When a validation or test set is used however, there will be underfitting as the additional irrelevant features can distort the prediction and lead to low validation or test score.

Problem of Overfitting and Underfitting

- The training data set may contain some outlier instances or sample points that may not occur uniformly or commonly across training and validation sample sets. These kinds of outliers may not occur in the validation or test data sets. By trying to fit with MMSE on training data set with outliers (that are not removed or not examined), coefficients associated with these outlier features may be fitted with values of larger magnitudes than if the outliers did not occur. This creates overfitting in the training data set (higher R^2) relative to the validation or test set that will see underfitting (lower R^2) when the fitted coefficients are used for prediction.
- Overfitting could also occur when some features are highly multi-collinear, i.e., having high correlations. Suppose a correctly specified regression is $Y_k = b_0+b_1X_k+\varepsilon_k$ where Y=(5,5.1,4.6,4.9). X=(9,7,8.5, 7.5). A simple regression yields R^2 of 0.18. Consider another feature Z that is highly correlated with X. Z=(1,3,0.5,2.5). Their correlation is –0.92. If we add Z to the regression, i.e., $Y_k = b_0+b_1X_k+b_2Z_k$+residual error, the R^2 becomes 0.87. When X and Z are added in the regression, their negative correlation smooths out their combined fluctuations, producing aggregated $0.36X_k+0.38Z_k$ that correlated highly with Y and hence yields a lower $R^{2.}$. The high multi-collinearity could produce overfitting in the training data set but in a different sample such as validation or test data set, the realizations of X and Z in small sample, it is more likely that since the correct specification is without Z, the out-of-sample prediction would be poorer with underfitting.

Overfitting and Underfitting in Machine Learning

Underfitting occurs when the fitted R^2's or the prediction R^2 are low. There is possibly an ideal balance between overfitting and underfitting for the training data as seen in the graph. At the ideal balance, it is likely that the validation and the test data fits will be best (least error). This is seen in the graph below.



Overfitting and Underfitting in Machine Learning

- To reduce overfitting and enable good validation and test data predictions, besides feature selections, some constraints can be added to the regression.
- A penalty function is added to reduce the tendencies to yield large-fitted coefficients – this technique is called regularization.
- There are three common regularized linear regression methods – Ridge regression, Lasso (Least absolute shrinkage and selection operator) regression, and Elastic Net regression.
- Regularization could shrink some fitted coefficients or even reduce them toward zero, effectively reducing the dimension space of the features since those affected features become impactless if their fitted coefficients are close to zero.
- They are also called shrinkage estimators. They reduce the ill effects of excessive number of features, outsized outliers and also high multi-collinearity.

Ridge Regression

- For Ridge regression using the module sklearn.linear_model.Ridge, see documentation in https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

$$\text{Min } \sum_{k=1}^{n}\left(Y_k - \sum_{j=0}^{p} b_j X_{jk}\right)^2 + \alpha\left(\sum_{j=0}^{p} b_j^2\right) \tag{2.2}$$

  where $X_{0k} = 1$. The second term is in L^2-norm, so Eq. (2.2) is also called a L^2 regularized regression. The L2 norm calculates the distance of the vector coordinate from the origin of the vector space. It is also known as the Euclidean norm as it is calculated as the Euclidean distance from the origin.

- The second term is the penalty term for the regression. When we use the computed value of sklearn linear_model. Ridge ( ), the default parameters within ( ) includes setting hyperparameter alpha, e.g., α =0.1.

Lasso Regression

- For Ridge regression using the module sklearn.linear_model.Lasso, see documentation in https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html.

$$\mathrm{Min}\ \sum_{k=1}^{n}\left(Y_k - \sum_{j=0}^{p} b_j X_{jk}\right)^2 + \alpha \sum_{j=0}^{p}\left|b_j\right| \qquad (2.3)$$

  where $X_{0k} = 1$. The second term is in L^1-norm, so Eq. (2.3) is also called a L^1 regularized regression. The L1 norm calculates the sum of absolute vector values, where the absolute value of a scalar is |.|.

- The second term is the penalty term for the regression. When we use the computed value of sklearn linear_model.Lasso ( ), the default parameters within ( ) includes setting hyperparameter alpha, e.g., α =0.1.

Elastic Net Regression

- Another possible regularized regression is the ‘elastic net’ regression that is the minimization of objective function as follows, where there are two hyperparameters α_1 and α_2.

$$\sum_{k=1}^{n}\left(Y_k - \sum_{j=0}^{p} b_j X_{jk}\right)^2 + \alpha_1\left(\sum_{j=1}^{p} b_j^2\right) + \alpha_2 \sum_{j=1}^{p}\left|b_j\right| \quad (2.4)$$

- This is a combination of regularizations from the Ridge and also the Lasso methods.

Worked Example – Data

Please upload Chapter2-1.ipynb and follow the computing steps in Jupyter Notebook

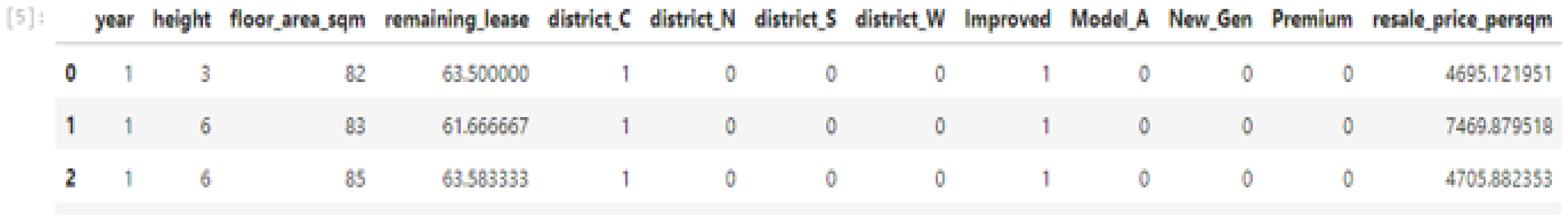
We explain the regularized regressions based on a study to predict Singapore Housing Development Board (HDB) resale flat (house) prices.

See the publicly accessible data sets at https://data.gov.sg/collections/189/view.

This method and other related methods in machine learning can be applied to predicting housing prices anywhere as long as there are adequate data pertaining to the house prices and features of each house.

See demonstration file Chapter2-1.ipynb.

We use a data set that comprises only 13 features (characteristics). 62,359 samples (sample points) are available in the data set HDBflat_value.csv, each of which is a flat in Singapore. The data cover the period 2017 through to 2023. The target variable is 'resale_price_persqm' – the transacted resale price of a 4-room flat in terms of Singapore dollars per square metre of the flat floor area.



[5]:

	year	height	floor_area_sqm	remaining_lease	district_C	district_N	district_S	district_W	Improved	Model_A	New_Gen	Premium	resale_price_persqm
0	1	3	82	63.500000	1	0	0	0	1	0	0	0	4695.121951
1	1	6	83	61.666667	1	0	0	0	1	0	0	0	7469.879518
2	1	6	85	63.583333	1	0	0	0	1	0	0	0	4705.882353

The features, after some pre-processing, are:

(1) 'year' – number of years after 2016 in which the resale took place, with resale in 2017 indicated as '1', resale in 2018 indicated as '2', and so on.

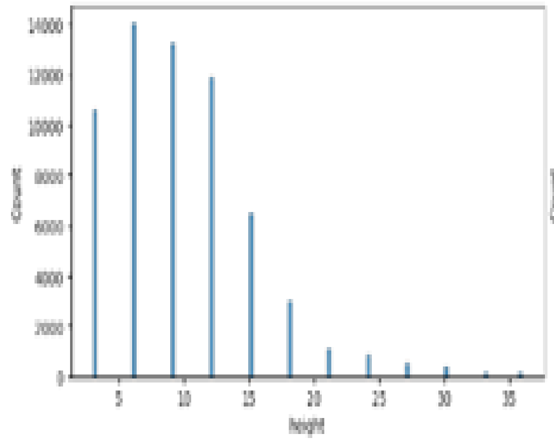
(2) 'height' – the level of the flat. For example, '9' indicates that the flat was in the 9th storey.

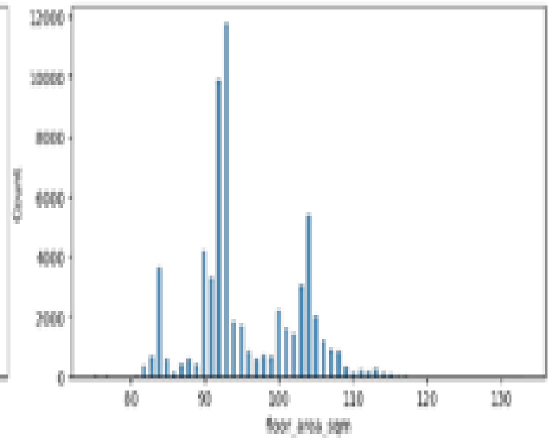
(3) 'floor_area_sqm' – the floor area of the flat

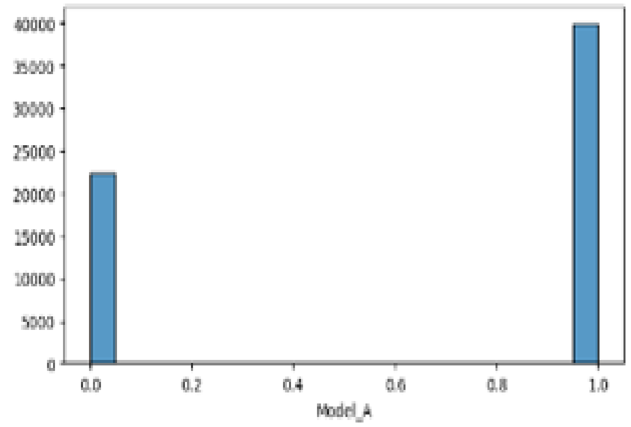
(4) 'remaining_lease' – the number of years left in the lease. All Singapore government flats have a 99 years old lease at the start. By the end of the lease, the flat reverts back to the government.

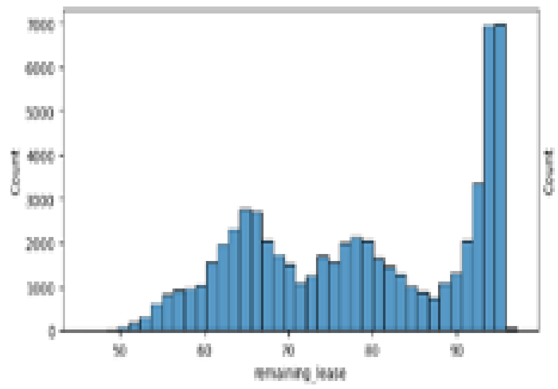
…………………………………………………………………….

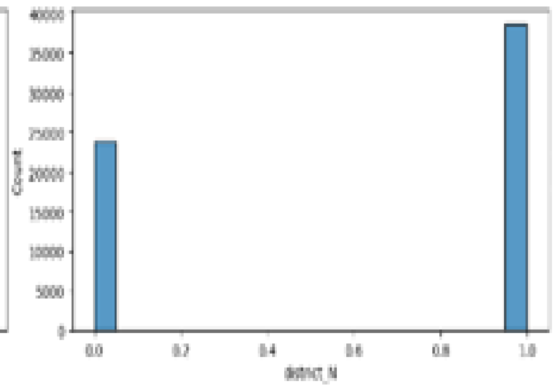
Code line [8] explores the empirical distributions of each feature. Such preliminary explorations provide some ideas of which features may be more important in the prediction.

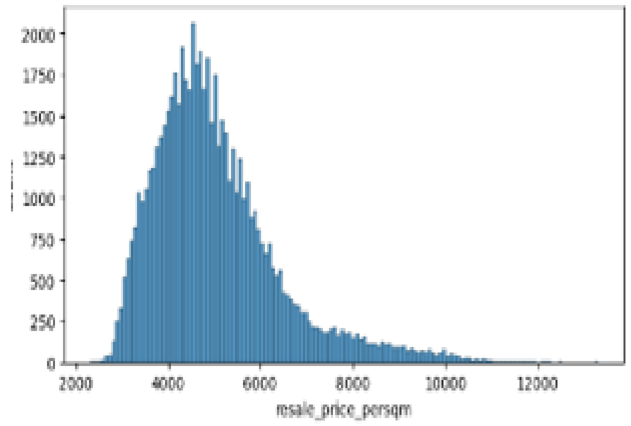










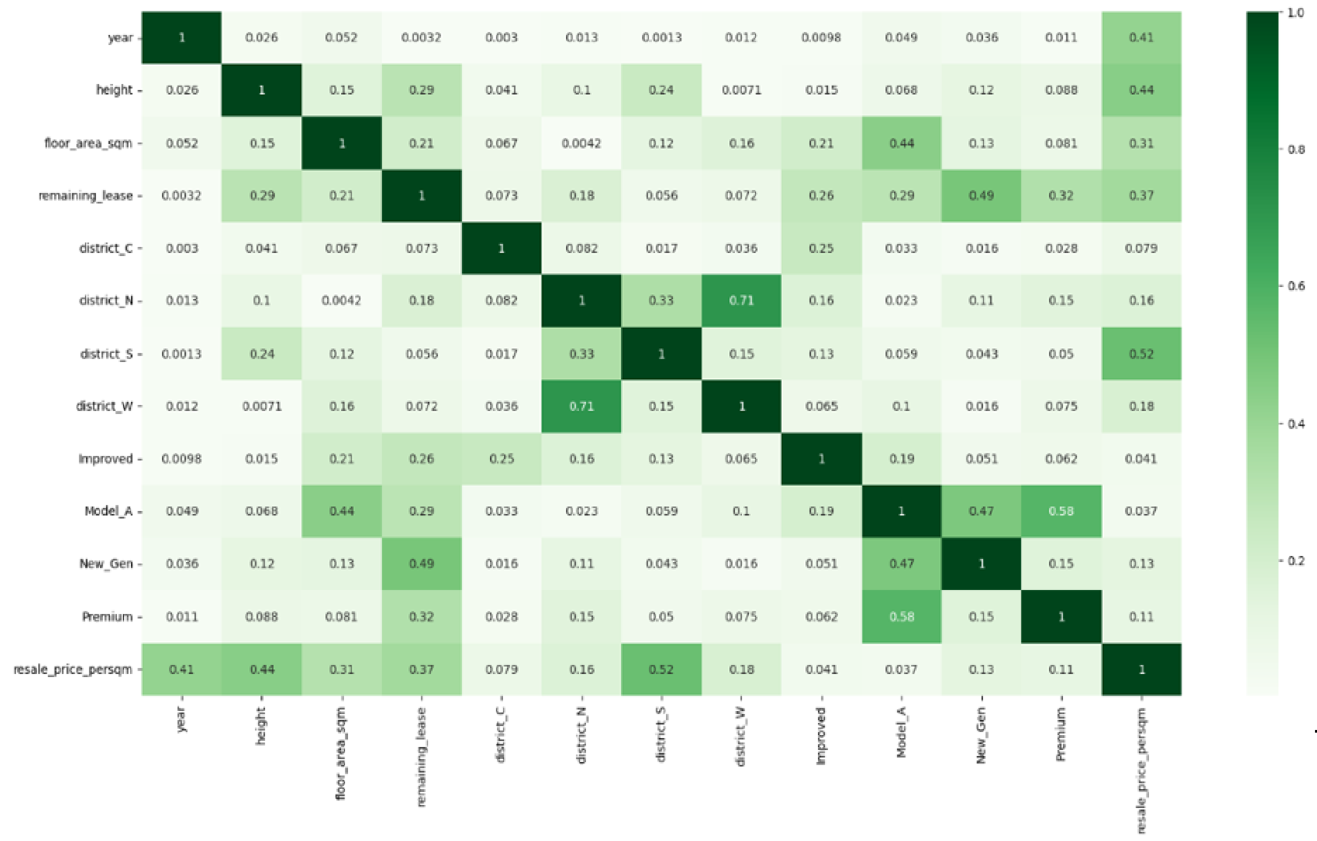


Pairwise Correlations of all Features

Code line [9] shows the correlation heat map amongst the features, including the target variable.

In the last row, it is seen that ‘year’, ‘height’, ‘remaining_lease’, and location ‘district_S’ all have high positive correlations with the target variable ‘resale_price_persqm’.

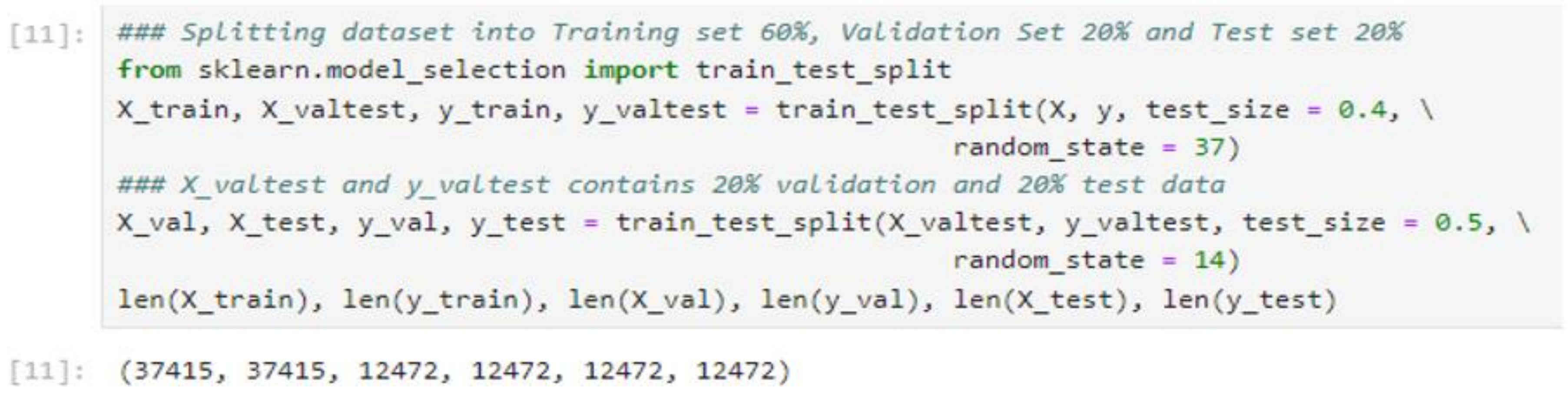
```
[9]:  <Axes: >
```



	year	height	floor_area_sqm	remaining_lease	district_C	district_N	district_S	district_W	Improved	Model_A	New_Gen	Premium	resale_price_persqm
year	1	0.026	0.052	0.0032	0.003	0.013	0.0013	0.012	0.0098	0.049	0.036	0.011	0.41
height	0.026	1	0.15	0.29	0.041	0.1	0.24	0.0071	0.015	0.068	0.12	0.088	0.44
floor_area_sqm	0.052	0.15	1	0.21	0.067	0.0042	0.12	0.16	0.21	0.44	0.13	0.081	0.31
remaining_lease	0.0032	0.29	0.21	1	0.073	0.18	0.056	0.072	0.26	0.29	0.49	0.32	0.37
district_C	0.003	0.041	0.067	0.073	1	0.082	0.017	0.036	0.25	0.033	0.016	0.028	0.079
district_N	0.013	0.1	0.0042	0.18	0.082	1	0.33	0.71	0.16	0.023	0.11	0.15	0.16
district_S	0.0013	0.24	0.12	0.056	0.017	0.33	1	0.15	0.13	0.059	0.043	0.05	0.52
district_W	0.012	0.0071	0.16	0.072	0.036	0.71	0.15	1	0.065	0.1	0.016	0.075	0.18
Improved	0.0098	0.015	0.21	0.26	0.25	0.16	0.13	0.065	1	0.19	0.051	0.062	0.041
Model_A	0.049	0.068	0.44	0.29	0.033	0.023	0.059	0.1	0.19	1	0.47	0.58	0.037
New_Gen	0.036	0.12	0.13	0.49	0.016	0.11	0.043	0.016	0.051	0.47	1	0.15	0.13
Premium	0.011	0.088	0.081	0.32	0.028	0.15	0.05	0.075	0.062	0.58	0.15	1	0.11
resale_price_persqm	0.41	0.44	0.31	0.37	0.079	0.16	0.52	0.18	0.041	0.037	0.13	0.11	1

Splitting Data Set

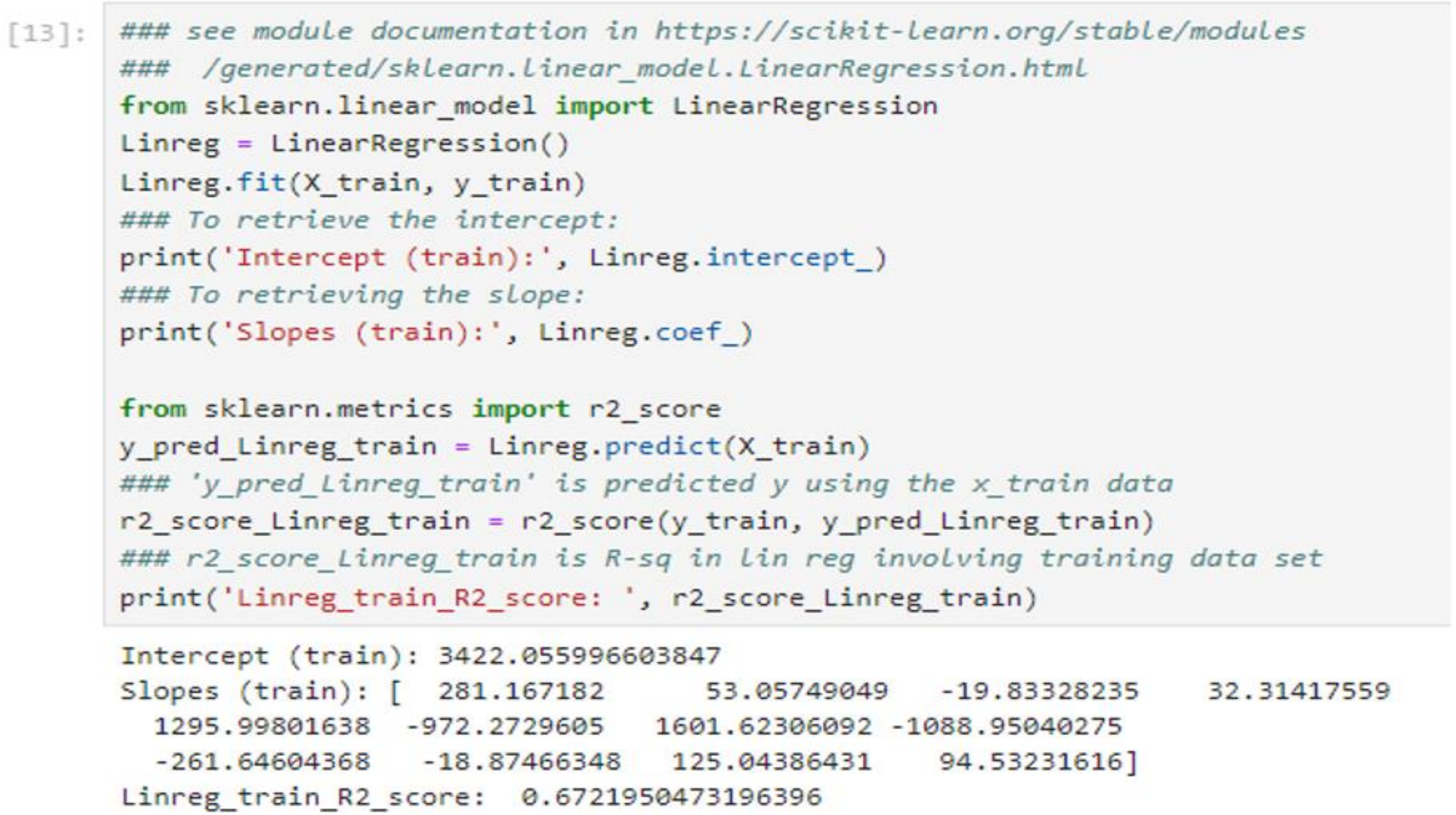
Next we define the target variable y and also the features X. The vector y and matrix X are then divided into (60%) rows forming corresponding X_train, y_train in training set, (20%) rows forming corresponding X_val, y_val in the validation set, and the remaining (20%) rows forming X_test, y_test in the test set. The rows are randomly assigned to the separate data sets via a random_state. When formed, the training, validation, and test data sets have correspondingly 37,415, 12,472, and 12, 472 sample sizes.



```
[11]: ### Splitting dataset into Training set 60%, Validation Set 20% and Test set 20%
      from sklearn.model_selection import train_test_split
      X_train, X_valtest, y_train, y_valtest = train_test_split(X, y, test_size = 0.4, \
                                                                random_state = 37)
      ### X_valtest and y_valtest contains 20% validation and 20% test data
      X_val, X_test, y_val, y_test = train_test_split(X_valtest, y_valtest, test_size = 0.5, \
                                                                random_state = 14)
      len(X_train), len(y_train), len(X_val), len(y_val), len(X_test), len(y_test)

[11]: (37415, 37415, 12472, 12472, 12472, 12472)
```

Linear Regression Prediction

A linear regression model is used as a base comparison model to predict the flat resale prices (per square metre). For comparison with the regularized models with hyperparameters, we employ the training set (60% data) for the initial fitting. This is done in code lines [13] and [14].

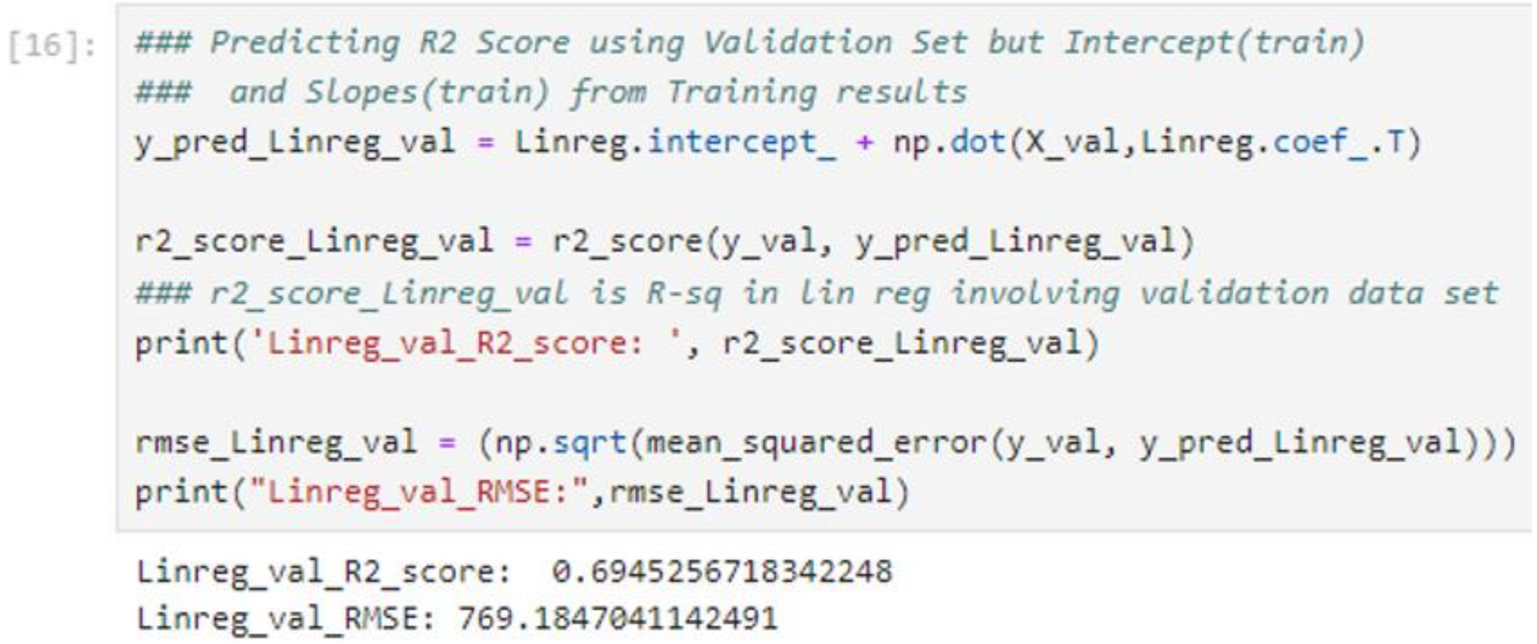


```
[13]: ### see module documentation in https://scikit-learn.org/stable/modules
      ###  /generated/sklearn.linear_model.LinearRegression.html
      from sklearn.linear_model import LinearRegression
      Linreg = LinearRegression()
      Linreg.fit(X_train, y_train)
      ### To retrieve the intercept:
      print('Intercept (train):', Linreg.intercept_)
      ### To retrieving the slope:
      print('Slopes (train):', Linreg.coef_)

      from sklearn.metrics import r2_score
      y_pred_Linreg_train = Linreg.predict(X_train)
      ### 'y_pred_Linreg_train' is predicted y using the x_train data
      r2_score_Linreg_train = r2_score(y_train, y_pred_Linreg_train)
      ### r2_score_Linreg_train is R-sq in lin reg involving training data set
      print('Linreg_train_R2_score: ', r2_score_Linreg_train)
```

```
Intercept (train): 3422.055996603847
Slopes (train): [  281.167182      53.05749049   -19.83328235    32.31417559
  1295.99801638  -972.2729605   1601.62306092 -1088.95040275
  -261.64604368   -18.87466348   125.04386431    94.53231616]
Linreg_train_R2_score:  0.6721950473196396
```

Linear Regression Prediction

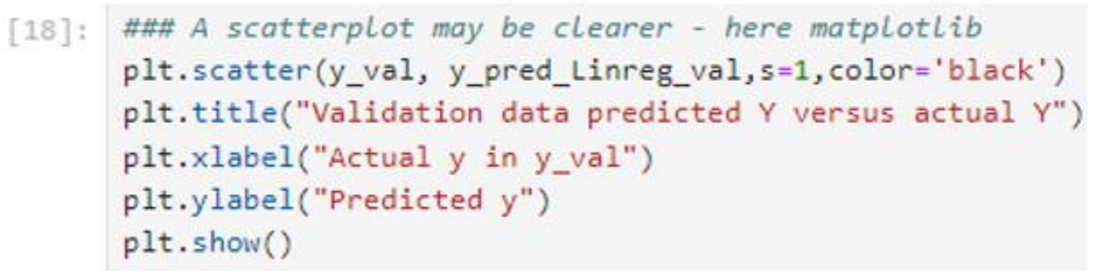
In code line [16], the fitted model is then used to predict targets in the validation data set employing features from the validation data set. The linear regression validation R^2 score is 0.69452567 with RMSE 769.1847. It is possible, as in this case, that the prediction outcomes are better than in the fitted model.



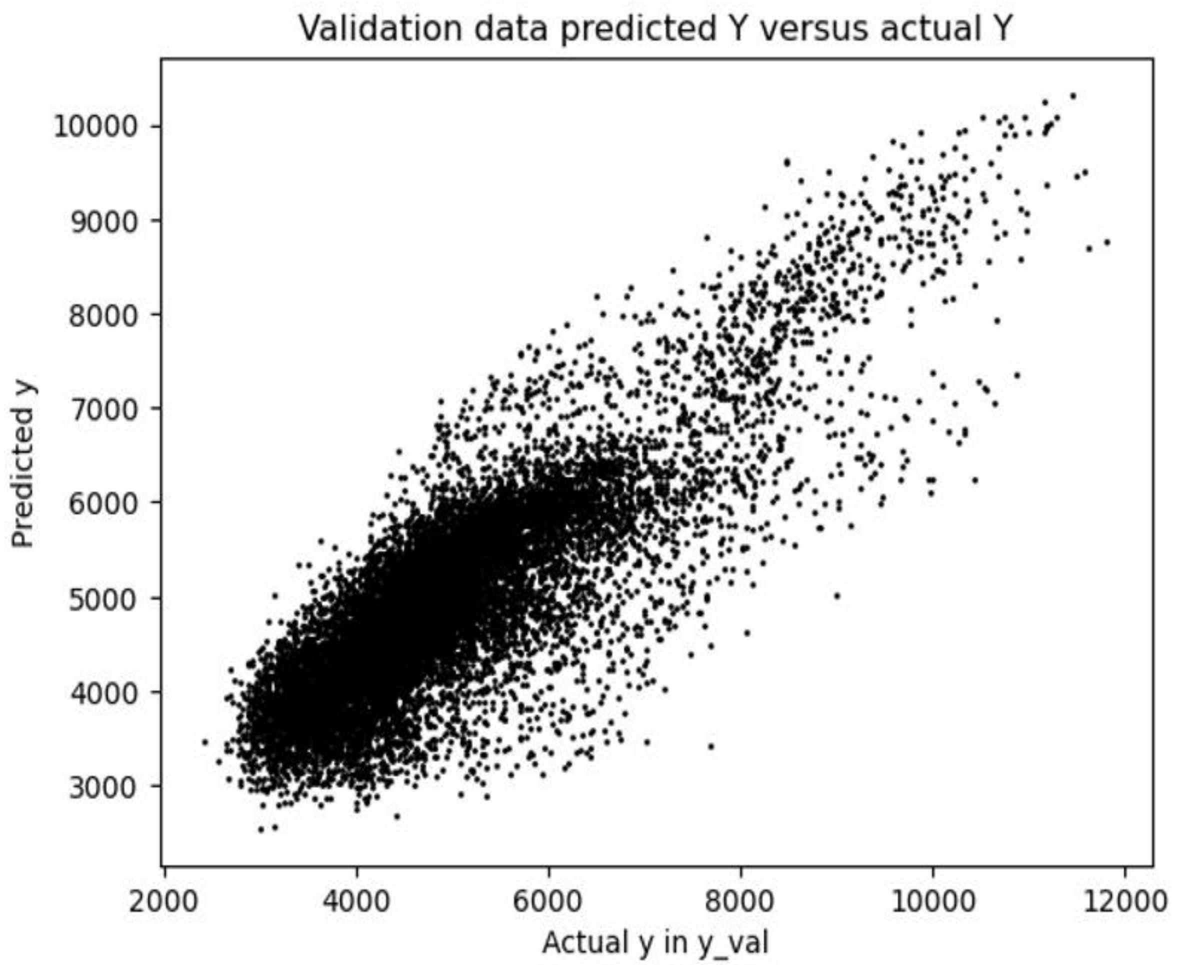
```
[16]: ### Predicting R2 Score using Validation Set but Intercept(train)
      ###  and Slopes(train) from Training results
      y_pred_Linreg_val = Linreg.intercept_ + np.dot(X_val,Linreg.coef_.T)

      r2_score_Linreg_val = r2_score(y_val, y_pred_Linreg_val)
      ### r2_score_Linreg_val is R-sq in lin reg involving validation data set
      print('Linreg_val_R2_score: ', r2_score_Linreg_val)

      rmse_Linreg_val = (np.sqrt(mean_squared_error(y_val, y_pred_Linreg_val)))
      print("Linreg_val_RMSE:",rmse_Linreg_val)

      Linreg_val_R2_score:  0.6945256718342248
      Linreg_val_RMSE: 769.1847041142491
```

Linear Regression Prediction

The predicted target in the validation set is plotted against the actual target values and this is shown in code line [18].



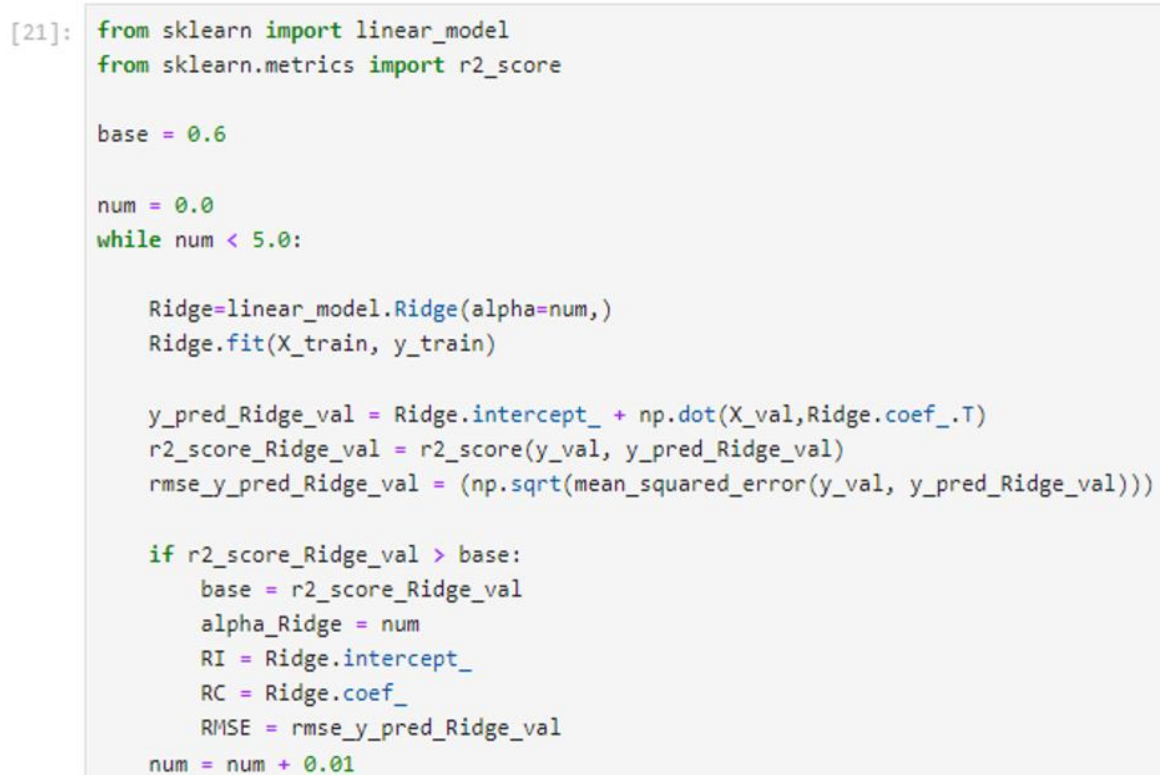
```
[18]: ### A scatterplot may be clearer - here matplotlib
      plt.scatter(y_val, y_pred_Linreg_val,s=1,color='black')
      plt.title("Validation data predicted Y versus actual Y")
      plt.xlabel("Actual y in y_val")
      plt.ylabel("Predicted y")
      plt.show()
```



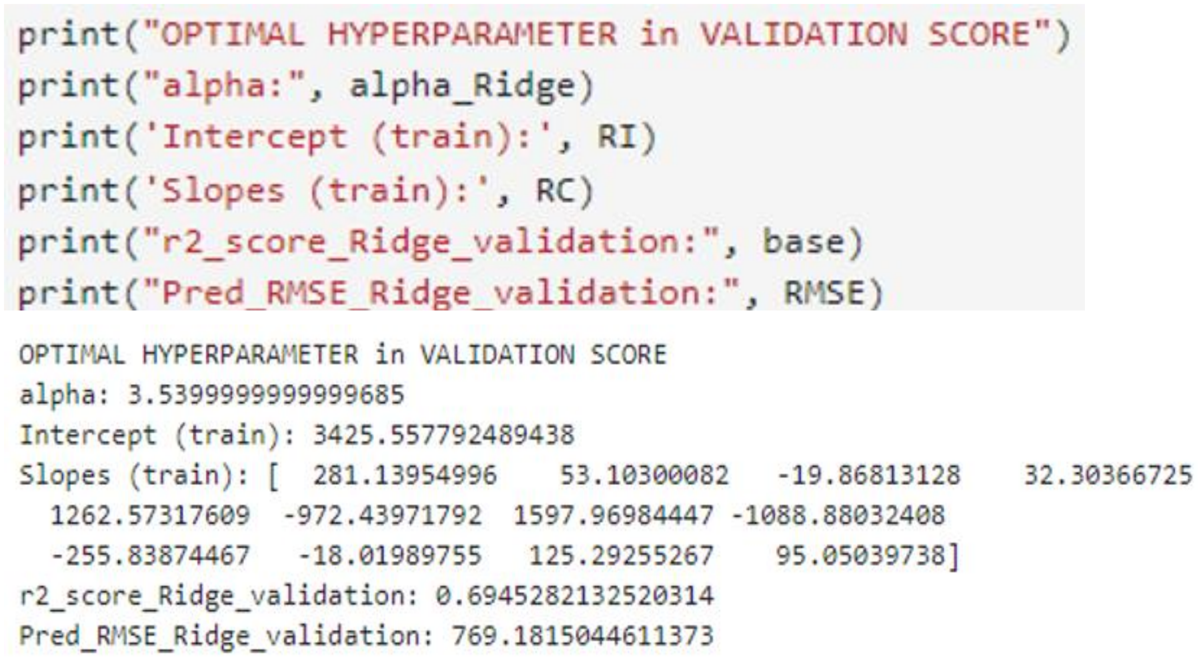
Fine-Tuning Hyperparameters

- The Ridge, Lasso, and Elastic Net regressions with constraints as in Eqs. (2.2), (2.3), and (2.4) are then used to fit the data in the training set.
- For each model in (2.2), (2.3), and (2.4), different hyperparameter value(s) each time is employed to fit the model using the training set data.
- The fitted model is then used to perform prediction based on features from the validation set.
- The validation prediction R^2 score is noted, and the hyperparameter values are iterated till a maximum validation R^2 score is obtained. The optimal (best or "fine-tuned") hyperparameter value(s) and the associated maximum validation R^2 score are noted for each model.

Code line [21] shows the outcome for the Ridge regression model.



```
[21]: from sklearn import linear_model
      from sklearn.metrics import r2_score

      base = 0.6

      num = 0.0
      while num < 5.0:

          Ridge=linear_model.Ridge(alpha=num,)
          Ridge.fit(X_train, y_train)

          y_pred_Ridge_val = Ridge.intercept_ + np.dot(X_val,Ridge.coef_.T)
          r2_score_Ridge_val = r2_score(y_val, y_pred_Ridge_val)
          rmse_y_pred_Ridge_val = (np.sqrt(mean_squared_error(y_val, y_pred_Ridge_val)))

          if r2_score_Ridge_val > base:
              base = r2_score_Ridge_val
              alpha_Ridge = num
              RI = Ridge.intercept_
              RC = Ridge.coef_
              RMSE = rmse_y_pred_Ridge_val
          num = num + 0.01
```



```
print("OPTIMAL HYPERPARAMETER in VALIDATION SCORE")
print("alpha:", alpha_Ridge)
print('Intercept (train):', RI)
print('Slopes (train):', RC)
print("r2_score_Ridge_validation:", base)
print("Pred_RMSE_Ridge_validation:", RMSE)
```

```
OPTIMAL HYPERPARAMETER in VALIDATION SCORE
alpha: 3.5399999999999685
Intercept (train): 3425.557792489438
Slopes (train): [  281.13954996    53.10300082   -19.86813128    32.30366725
  1262.57317609  -972.43971792  1597.96984447 -1088.88032408
  -255.83874467   -18.01989755   125.29255267    95.05039738]
r2_score_Ridge_validation: 0.6945282132520314
Pred_RMSE_Ridge_validation: 769.1815044611373
```

Optimal alpha (with highest R^2 score) is 3.54. Compared with the base regression model with no constraint, the validation R^2 score is marginally higher at 0.69452821 while the RMSE is marginally lower at 769.1815. The estimated coefficients for features ‘district_C’, ‘district_S’, and ‘Improved’ also have smaller magnitudes due to the Ridge regression constraint. Hence the constraints lead to better prediction.

The validation R^2 scores for the linear regression with no constraints, the Ridge, Lasso, and Elastic Net regressions are tabulated below with the optimal hyperparameters shown in the brackets.

	Validation R^2
Linear Regression (no constraints)	0.69452567
Ridge Regression (sklearn $\alpha = 3.54$)	0.69452821
Lasso Regression (sklearn $\alpha = 0.043$)	0.69452610
Elastic Net Regression (sklearn $\alpha = 0.001$, l1_ratio = 0.899)	0.69452818

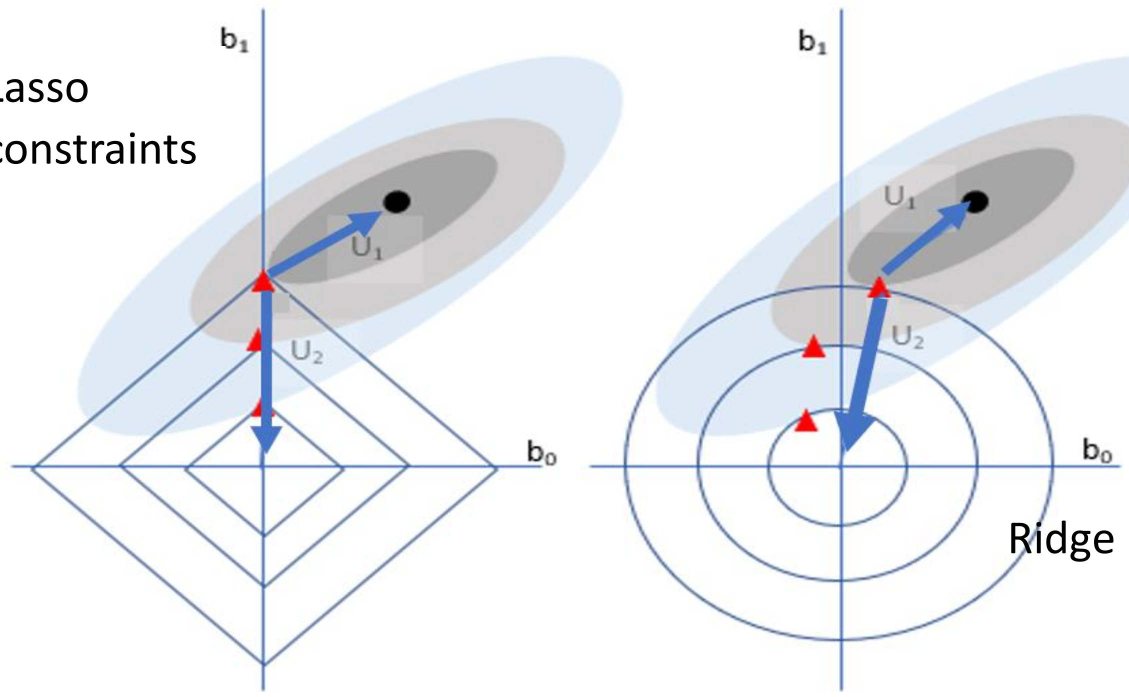
In the data studied, the various regressions produce close solutions. In general, however, there are some differences between the Ridge and the Lasso regression due to the difference in their constraints.

The difference between Ridge and Lasso regressions can be illustrated geometrically as follows using a two-dimensional case of minimizing the sum of squared errors

$$\sum_{k=1}^{n} (Y_k - b_0 - b_1 X_{1k})^2$$

that is geometrically an ellipse centred at (b_0, b_1), taking the values (Y_k, X_k)'s as given. Under regularization, its minimization is subject to the Lasso type constraints (LHS below) and the Ridge type constraints (RHS below).

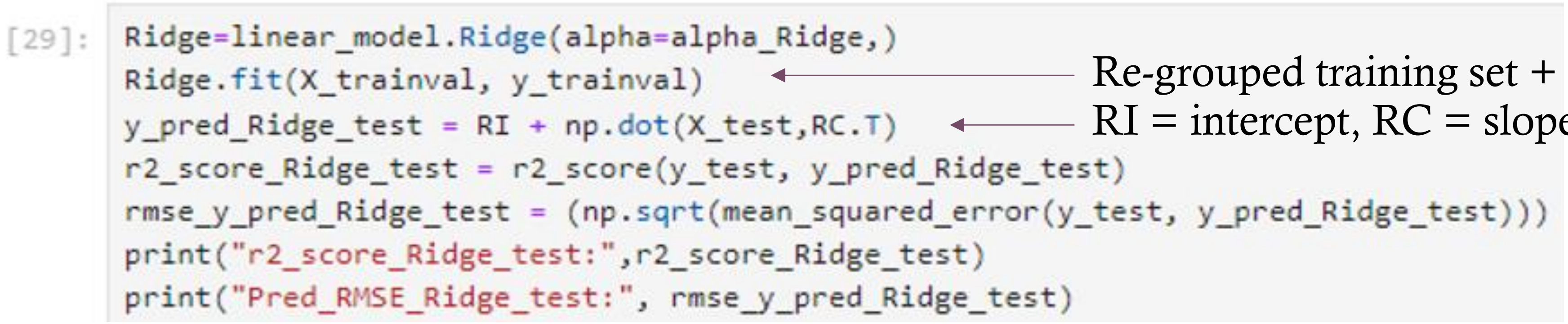
In general, for problems with a much larger set of features, Lasso models are adept at feature selection – i.e., selecting a reduced/smaller set of features from a large set.



The contoured rings, contoured squares, and contoured circles represent equal levels.

Prediction Using Hold-Out Test Set

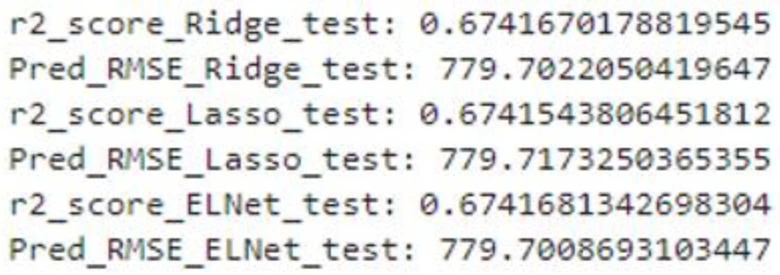
We perform the optional step of combining the training data set and the validation data set for re-training the model parameters, but using the fine-tuned hyperparameters from the validation phase. The re-trained model is then applied to the prediction of targets from the test data set using the test data features. Code line [29] shows how the step is done for the Ridge regression case.



```
[29]: Ridge=linear_model.Ridge(alpha=alpha_Ridge,)
      Ridge.fit(X_trainval, y_trainval)
      y_pred_Ridge_test = RI + np.dot(X_test,RC.T)
      r2_score_Ridge_test = r2_score(y_test, y_pred_Ridge_test)
      rmse_y_pred_Ridge_test = (np.sqrt(mean_squared_error(y_test, y_pred_Ridge_test)))
      print("r2_score_Ridge_test:",r2_score_Ridge_test)
      print("Pred_RMSE_Ridge_test:", rmse_y_pred_Ridge_test)
```

Re-grouped training set + validation set

RI = intercept, RC = slope from Ridge regression



```
r2_score_Ridge_test: 0.6741670178819545
Pred_RMSE_Ridge_test: 779.7022050419647
r2_score_Lasso_test: 0.6741543806451812
Pred_RMSE_Lasso_test: 779.7173250365355
r2_score_ELNet_test: 0.6741681342698304
Pred_RMSE_ELNet_test: 779.7008693103447
```

In [29], we find the test data prediction scores for all 3 models since they perform with marginal differences in the validation phase. As seen in the output, the Elastic Net regression model is now marginally better than that of the Ridge regression. The performance at R^2 of over 67% is close to the validation R^2 performance of just over 69%.

Cross-Validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. K-fold cross validation refers to the number of groups that a given data sample less the hold-out test sample is split into. This method allows validation to be conducted within the training data set itself and thus avoids use of another separate set of data for validation.
- k approximately equal subsets of the data sample after leaving out the hold-out set are drawn randomly. Pick a subset as the validation sample. Pick the rest of subsets as training sample. Fit model on training sample and evaluate prediction R^2 score on the validation sample. Pick the next subset as the validation sample, and the rest as training sample, and so on. There will be k number of such fittings and k number of validation R^2 scores using this k-fold cross-validation approach. The mean and variance of the k number of R^2 scores can be computed.
- A different set of hyperparameter value(s) is used for each cross-validation mean R^2 score. The optimal hyperparameter value(s) is/are that which yield(s) the maximum mean R^2 score in the cross-validation. This optimal hyperparameter value(s) is/are then used to check the prediction performance of the test data set. The cross-validation procedure essentially replaces 0.6N, 0.2N, 0.2N slicing of training, validation, and test data sets with 0.8N, 0.2N simpler slicing into training and test data sets. If k=5, then the training data set is divided into five times of 0.64N training and 0.16N validation.

Cross-Validation

- Fine-tune hyperparameters
- Select optimal model

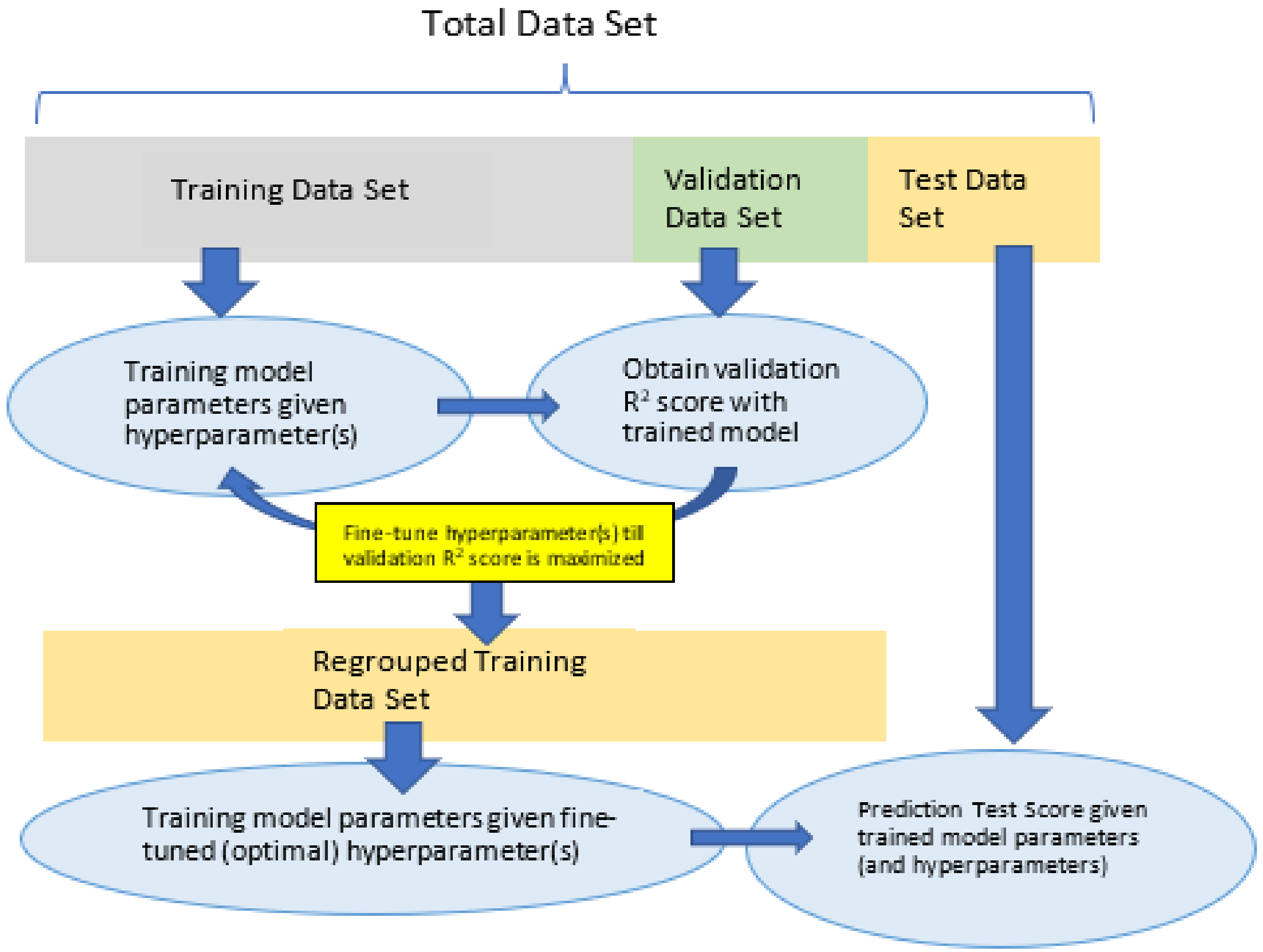
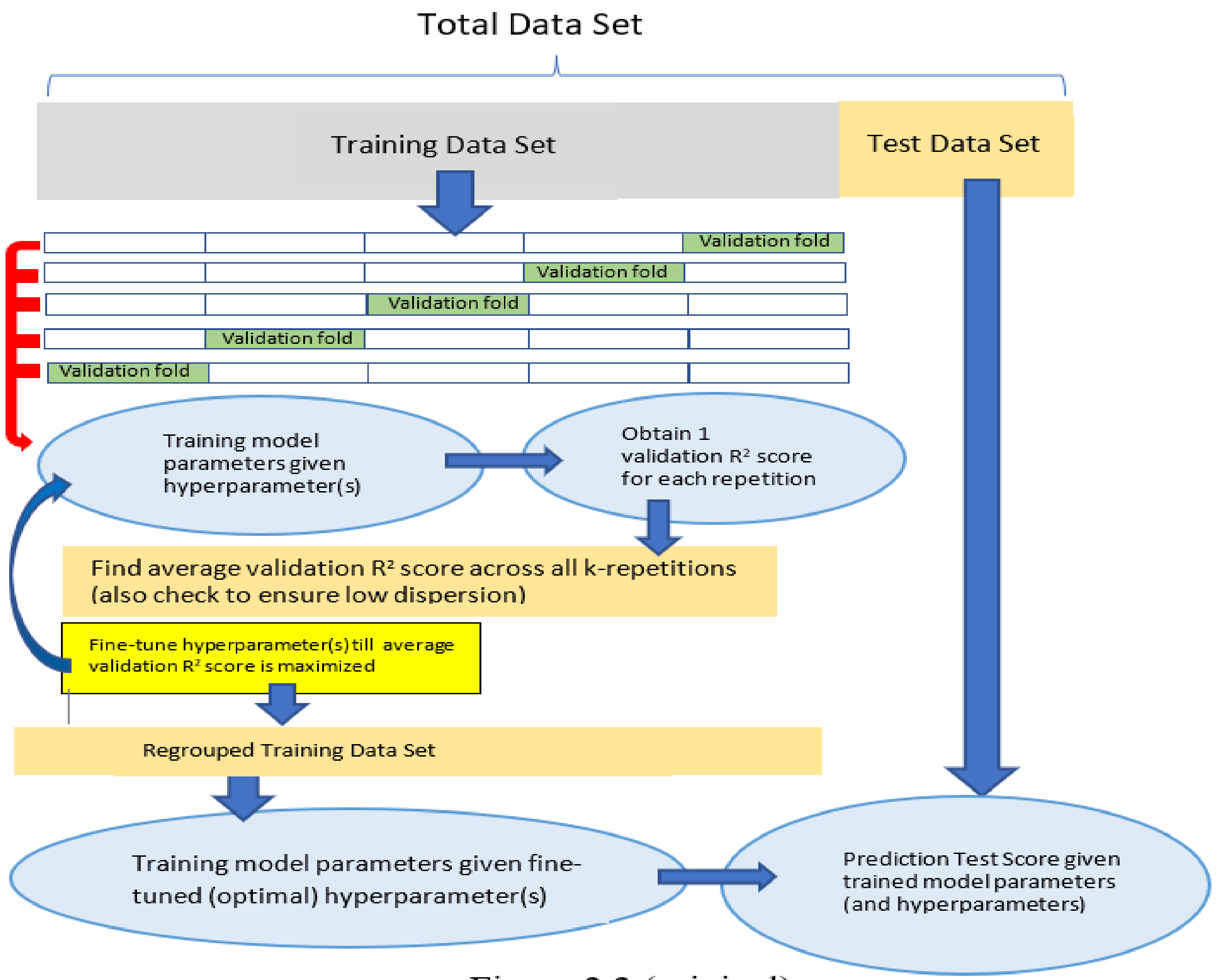

Total Data Set
Training Data Set
Validation Data Set
Test Data Set
Training model parameters given hyperparameter(s)
Obtain validation R² score with trained model
Fine-tune hyperparameter(s) till validation R² score is maximized
Regrouped Training Data Set
Training model parameters given fine-tuned (optimal) hyperparameter(s)
Prediction Test Score given trained model parameters (and hyperparameters)

Figure 2.1 (original)

k-Fold Cross-Validation

- The k-fold cross-validation is a common approach or strategy to improve validation effectiveness and thus come up with an optimal model (with optimized model parameters as well as hyperparameters) .
- This is done particularly with smaller sample size (smaller number of sample points) in the total data set.
- Usually not applicable when the data are in time series form as time ordering cannot be randomly re-arranged.
- Provides k **validation** R^2 scores. Computes mean and standard deviation of scores.

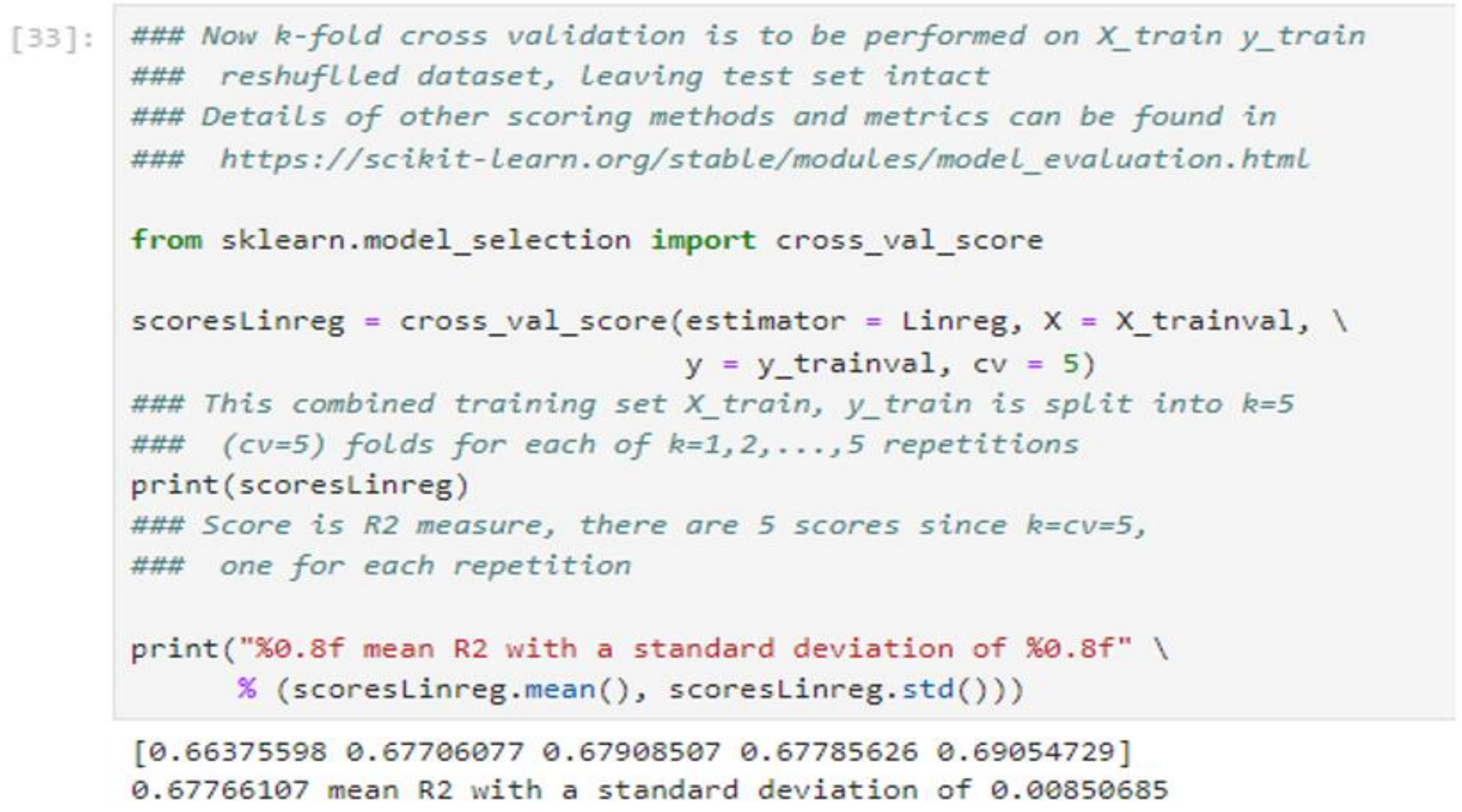


Total Data Set
Training Data Set
Test Data Set
Validation fold
Validation fold
Validation fold
Validation fold
Validation fold
Training model parameters given hyperparameter(s)
Obtain 1 validation R² score for each repetition
Find average validation R² score across all k-repetitions (also check to ensure low dispersion)
Fine-tune hyperparameter(s) till average validation R² score is maximized
Regrouped Training Data Set
Training model parameters given fine-tuned (optimal) hyperparameter(s)
Prediction Test Score given trained model parameters (and hyperparameters)

Figure 2.2 (original)

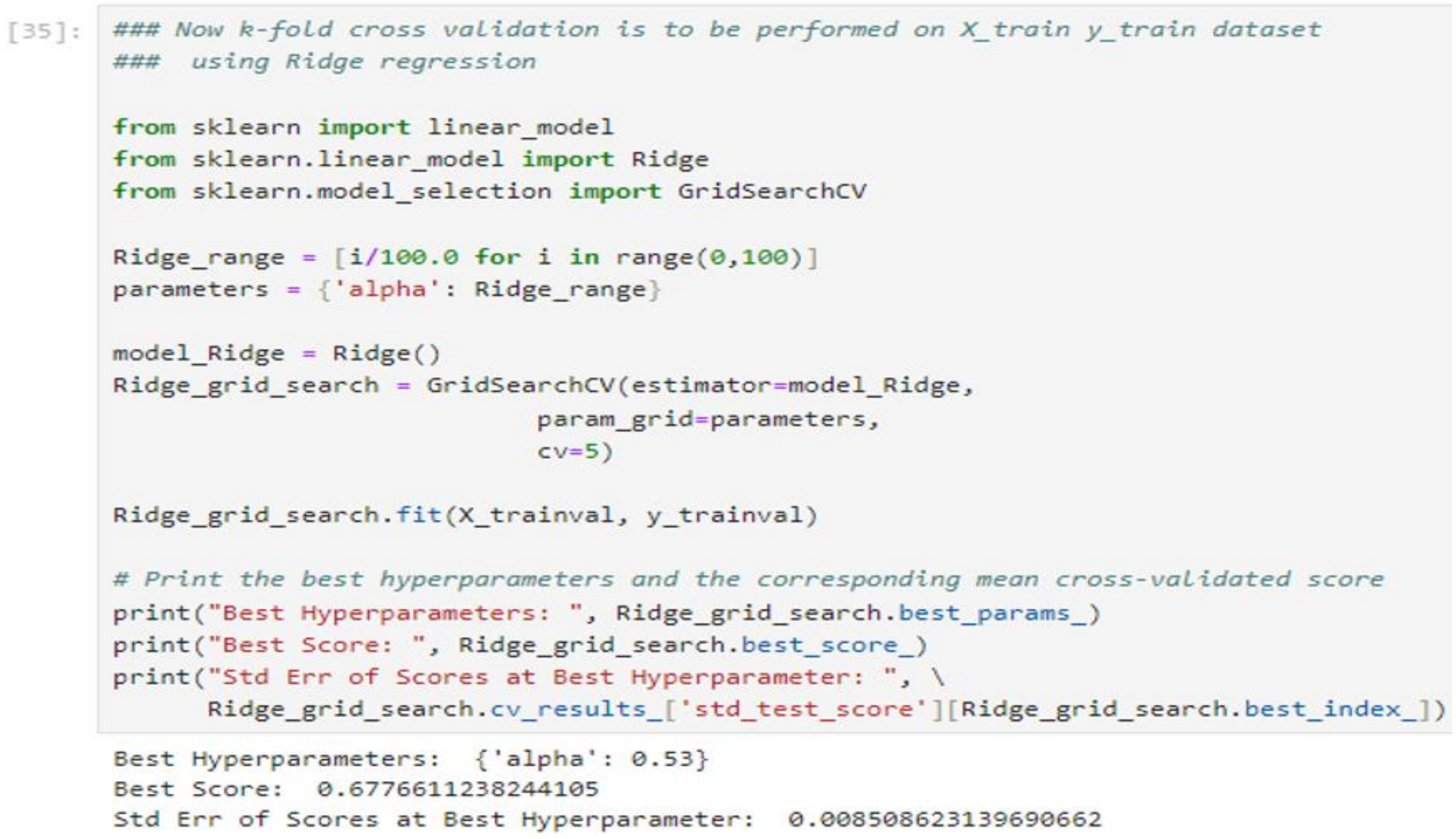
k-Fold Cross-Validation

Code line [30] shows that the data set could be reshuffled anew with a fresh X_train and y_train training set of 49,887 sample points. Code line [33] provides the cross-validation results for the **base case** of linear regression without constraint.



```
[33]: ### Now k-fold cross validation is to be performed on X_train y_train
      ###  reshuflled dataset, leaving test set intact
      ### Details of other scoring methods and metrics can be found in
      ###  https://scikit-learn.org/stable/modules/model_evaluation.html

      from sklearn.model_selection import cross_val_score

      scoresLinreg = cross_val_score(estimator = Linreg, X = X_trainval, \
                                     y = y_trainval, cv = 5)
      ### This combined training set X_train, y_train is split into k=5
      ###  (cv=5) folds for each of k=1,2,...,5 repetitions
      print(scoresLinreg)
      ### Score is R2 measure, there are 5 scores since k=cv=5,
      ###  one for each repetition

      print("%0.8f mean R2 with a standard deviation of %0.8f" \
            % (scoresLinreg.mean(), scoresLinreg.std()))

      [0.66375598 0.67706077 0.67908507 0.67785626 0.69054729]
      0.67766107 mean R2 with a standard deviation of 0.00850685
```

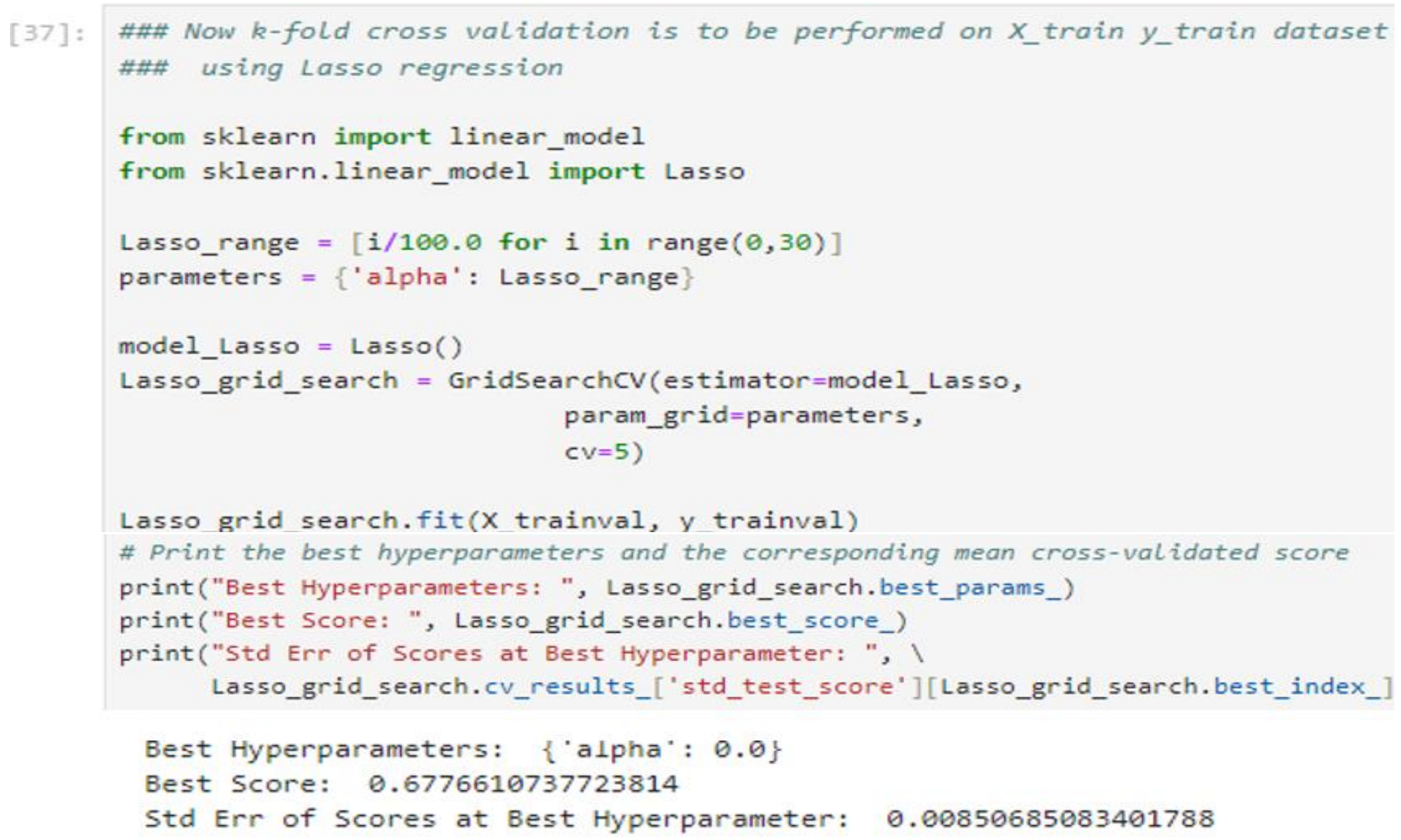
k-Fold Cross-Validation - Ridge Regression with GridSearch



```
[35]: ### Now k-fold cross validation is to be performed on X_train y_train dataset
      ###  using Ridge regression

      from sklearn import linear_model
      from sklearn.linear_model import Ridge
      from sklearn.model_selection import GridSearchCV

      Ridge_range = [i/100.0 for i in range(0,100)]
      parameters = {'alpha': Ridge_range}

      model_Ridge = Ridge()
      Ridge_grid_search = GridSearchCV(estimator=model_Ridge,
                                       param_grid=parameters,
                                       cv=5)

      Ridge_grid_search.fit(X_trainval, y_trainval)

      # Print the best hyperparameters and the corresponding mean cross-validated score
      print("Best Hyperparameters: ", Ridge_grid_search.best_params_)
      print("Best Score: ", Ridge_grid_search.best_score_)
      print("Std Err of Scores at Best Hyperparameter: ", \
            Ridge_grid_search.cv_results_['std_test_score'][Ridge_grid_search.best_index_])
```

```
Best Hyperparameters:  {'alpha': 0.53}
Best Score:  0.6776611238244105
Std Err of Scores at Best Hyperparameter:  0.008508623139690662
```

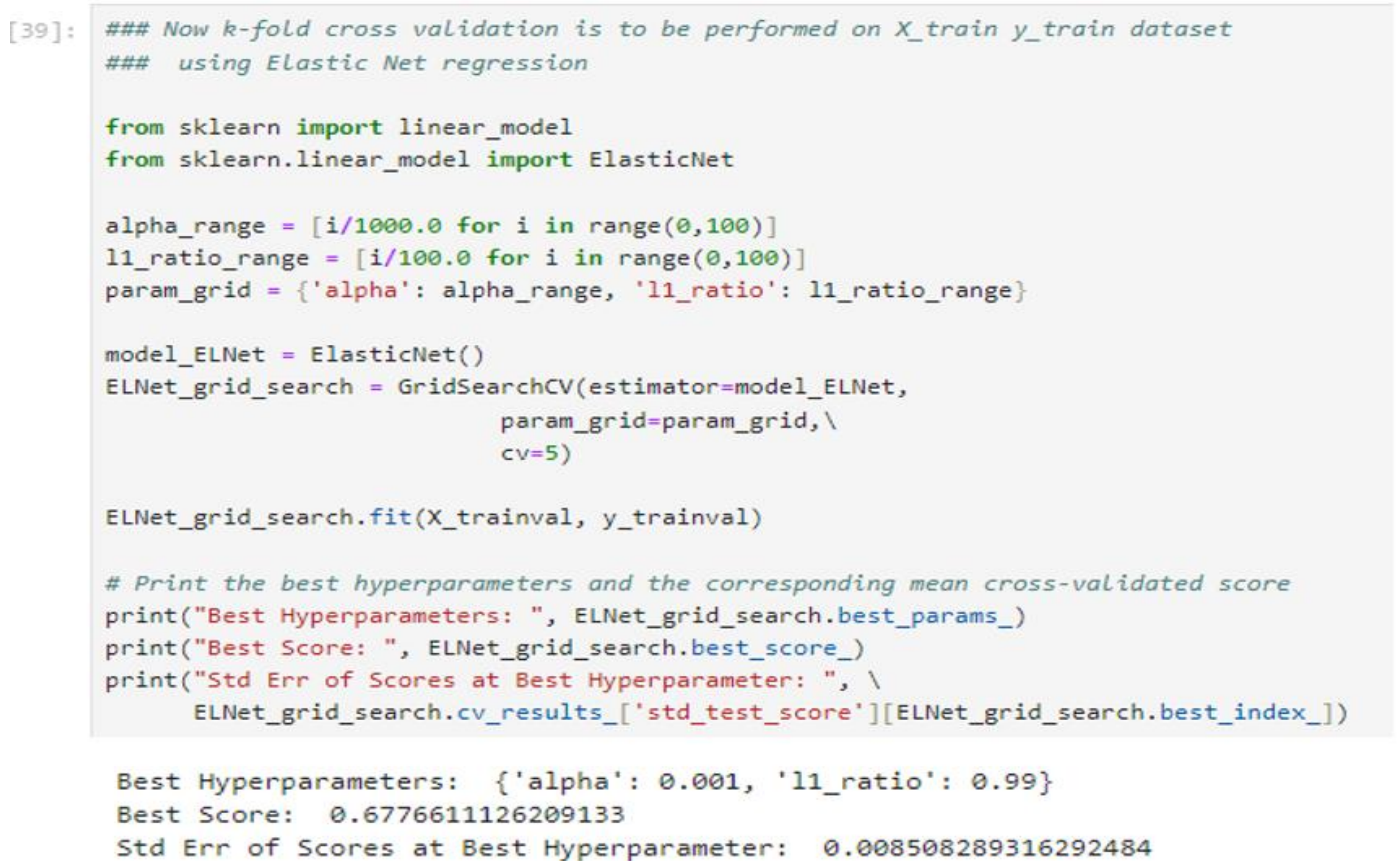
k-Fold Cross-Validation - Lasso Regression with GridSearch



```
[37]: ### Now k-fold cross validation is to be performed on X_train y_train dataset
      ###  using Lasso regression

      from sklearn import linear_model
      from sklearn.linear_model import Lasso

      Lasso_range = [i/100.0 for i in range(0,30)]
      parameters = {'alpha': Lasso_range}

      model_Lasso = Lasso()
      Lasso_grid_search = GridSearchCV(estimator=model_Lasso,
                                   param_grid=parameters,
                                   cv=5)

      Lasso_grid_search.fit(X_trainval, y_trainval)
      # Print the best hyperparameters and the corresponding mean cross-validated score
      print("Best Hyperparameters: ", Lasso_grid_search.best_params_)
      print("Best Score: ", Lasso_grid_search.best_score_)
      print("Std Err of Scores at Best Hyperparameter: ", \
           Lasso_grid_search.cv_results_['std_test_score'][Lasso_grid_search.best_index_]
```

```
Best Hyperparameters:  {'alpha': 0.0}
Best Score:  0.6776610737723814
Std Err of Scores at Best Hyperparameter:  0.008506850834O1788
```

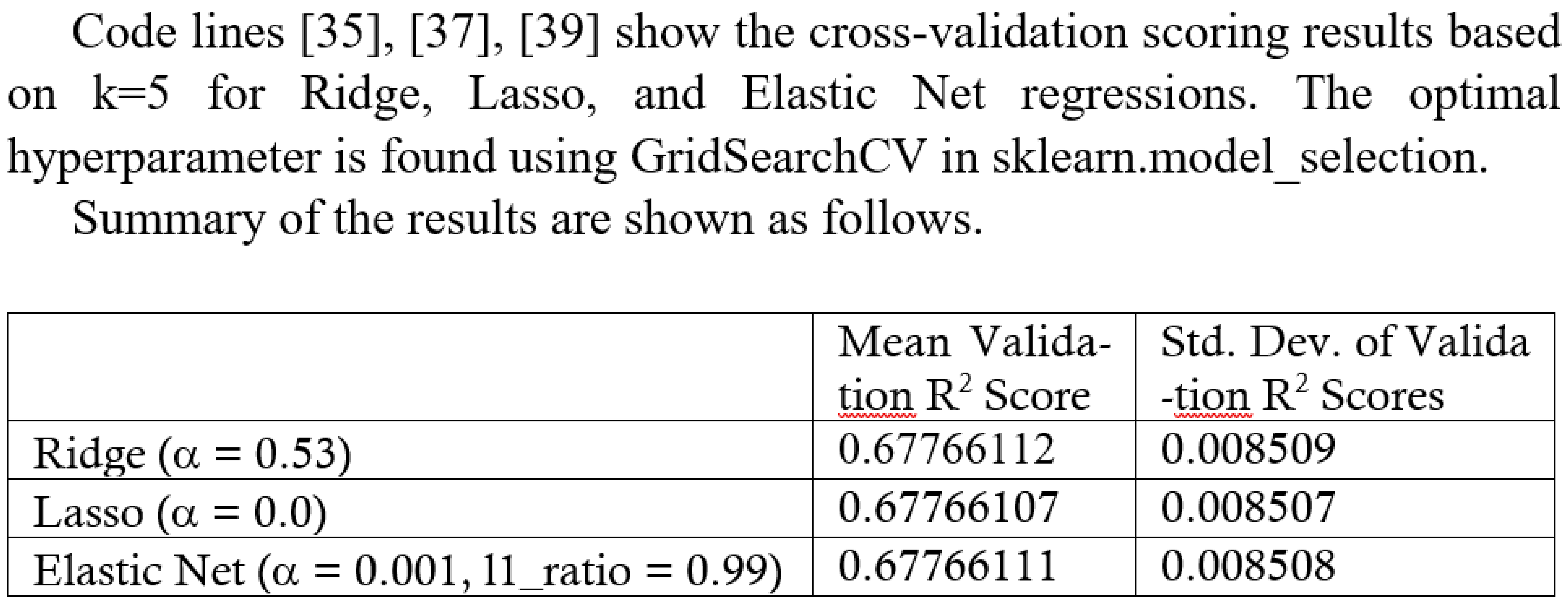
k-Fold Cross-Validation – Elastic Net Regression with GridSearch



```
[39]: ### Now k-fold cross validation is to be performed on X_train y_train dataset
      ###  using Elastic Net regression

      from sklearn import linear_model
      from sklearn.linear_model import ElasticNet

      alpha_range = [i/1000.0 for i in range(0,100)]
      l1_ratio_range = [i/100.0 for i in range(0,100)]
      param_grid = {'alpha': alpha_range, 'l1_ratio': l1_ratio_range}

      model_ELNet = ElasticNet()
      ELNet_grid_search = GridSearchCV(estimator=model_ELNet,
                                 param_grid=param_grid,\
                                 cv=5)

      ELNet_grid_search.fit(X_trainval, y_trainval)

      # Print the best hyperparameters and the corresponding mean cross-validated score
      print("Best Hyperparameters: ", ELNet_grid_search.best_params_)
      print("Best Score: ", ELNet_grid_search.best_score_)
      print("Std Err of Scores at Best Hyperparameter: ", \
            ELNet_grid_search.cv_results_['std_test_score'][ELNet_grid_search.best_index_])
```

```
Best Hyperparameters:  {'alpha': 0.001, 'l1_ratio': 0.99}
Best Score:  0.6776611126209133
Std Err of Scores at Best Hyperparameter:  0.008508289316292484
```

k-fold Cross-Validation Scores



Code lines [35], [37], [39] show the cross-validation scoring results based on k=5 for Ridge, Lasso, and Elastic Net regressions. The optimal hyperparameter is found using GridSearchCV in sklearn.model_selection.

Summary of the results are shown as follows.

	Mean Validation R^2 Score	Std. Dev. of Validation R^2 Scores
Ridge ($\alpha = 0.53$)	0.67766112	0.008509
Lasso ($\alpha = 0.0$)	0.67766107	0.008507
Elastic Net ($\alpha = 0.001$, l1_ratio = 0.99)	0.67766111	0.008508

As seen in the cross-validation results using R^2 as the metric or measurement of the accuracy of prediction performance, Ridge regression performs marginally better than regression without constraint and also better than Lasso and Elastic Net, in terms of mean R^2.

Concluding Thoughts

- There are advantages in the use of regularized regressions as a comparison to the linear regression model without constraints when the model may have too many features and can be overfitted using the training data.
- When the data sample size is small, a cross-validation method can be used to accompany regularized regressions. The advantage of cross-validation is that its results are likely to be more robust, The advantage of cross-validation is that its results are likely to be more robust, e.g., in selecting optimal hyperparameter(s) by considering average of the validation R^2 scores. However, cross-validation may take up more computing time with the number of repetitions, especially with larger data sets.
- When the prediction problem does not have too many features, then using regularizations may not significantly improve prediction performance. Often, data features that may have large differences in their values require scaling of the features to improve the performance of the method.
- When the prediction problem does not have too many features, then using regularizations, particularly Lasso, may yield under-fitting when coefficients are forced to be less impactful. Thus, regularizations should be done only when it improves the fitting and testing accuracies.
- By today's data standards, for predicting real estate values professionally, a lot more features can and should be added to improve the prediction.

Homework 1 Graded Exercise:

Chapter2-2.ipynb (this file is provided only after grading)

Use data set 'cruise_ship_info.csv' to predict the number of crews required to man a cruise ship given certain features/attributes of the ship. Use the features Age, TonnageGTx1000 (30.0 means 30,000 gross tons), passengersx100 (6.94 means 694 passengers), lengthx100ft (5.94 means 594 feet length of ship), cabinsx100 (3.55 means a total of 355 cabins on the ship), and spaceratio (42.64 means number of gross tonnage divided by number of passengers on ship). Target variable is Crewx100 (3.55 means 355 crews on the ship).

Use StandardScaler to standardize all the features (subtracting by mean and dividing by standard deviation). Do not standardize the target variable.

Split the 158 rows of sample into 70% training and 30% test. In train_test_split, use random_state = 0. (Ignore use of validation set in this exercise.)

Homework 1 Graded Exercise:

Q1. Using LinearRegression in sklearn.linear.model, find R^2 for training data set, and R^2 score for the test data set.

Q2. Now use a Ridge Regression with alpha = 0.05 and report the training and test R^2 scores.

Q3. Now use a LASSO Regression with alpha = 0.01 and report the training and test R^2 scores.

Q4. Now use 4-fold cross validation (without reshuffling), on the X_train and y_train. Use Linreg and report the mean R^2 scores and standard deviation of R^2 scores in the 4 cross-validation cases.

Q5. Now use 4-fold cross validation (without reshuffling), on the X_train and y_train. Use Ridge regression (alpha = 0.05) and report the mean R^2 scores and standard deviation of R^2 scores in the 4 cross-validation cases.

End of Class