**DS620 Machine Learning and Deep Learning**

**HOS03A Regression**

03/26/2021 Developed by Minh Nguyen

04/08/2021 Reviewed by Shanshan Yu

School of Technology & Computing @City University of Seattle (CityU)

**Learning outcome**

* Machine Learning process for Regression problem
* Model selection with Cross validation
* Fine Tuning model
* Evaluate Regression models

**Resources**

* scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation. (n.d.). Scikit-Learn. <https://scikit-learn.org/stable/index.html>
* Massaron, L., Boschetti, A. (2016). Regression Analysis with Python. Packt Publishing.
* Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc.

**Introduction**

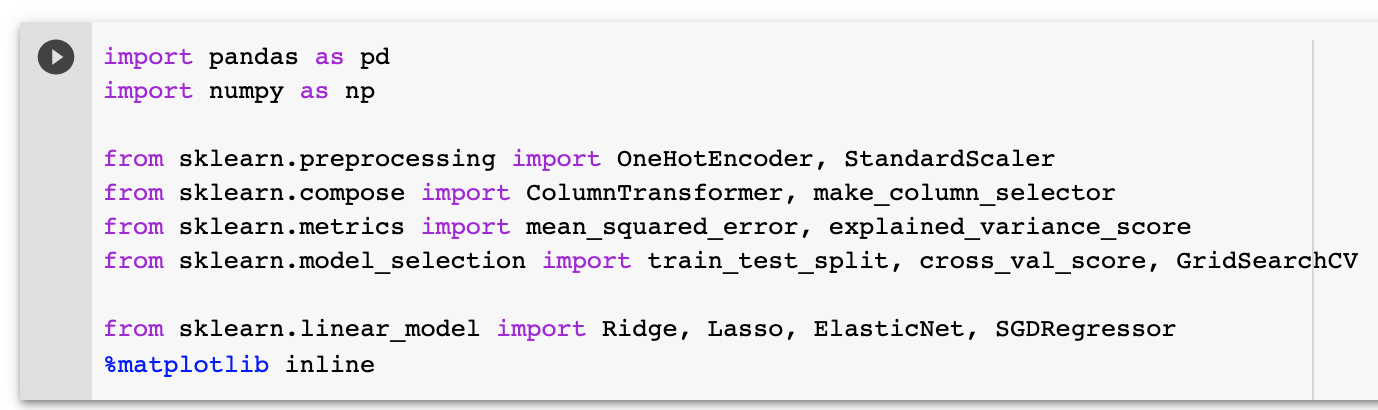
For this HOS, we will go over the same process as the previous chapter. However, we will tackle Regression problem which is about predicting a continuous variable. Unlike classification, where the set of target label is often very limited, evaluating a regression model can often be daunting as the set of target label is significantly greater. With the knowledge we’ve already.

**Overview of a Machine Learning project**

1. Get the data.
2. Discover and visualize the data to gain insights.
3. Prepare the data for Machine Learning algorithms.
4. Select a model and train it.
5. Fine-tune your model.

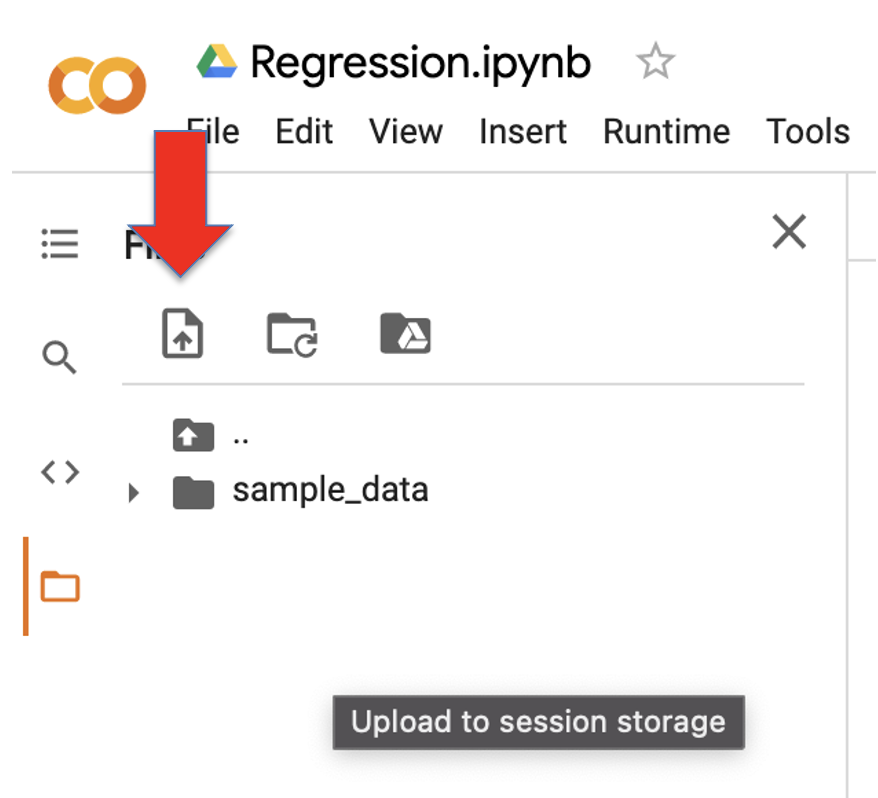
***Preparing development environment***

1. From [Google Colab](https://colab.research.google.com/), create a new notebook, name it “Regression.ipynb”
2. Type the following codes to import libraries.

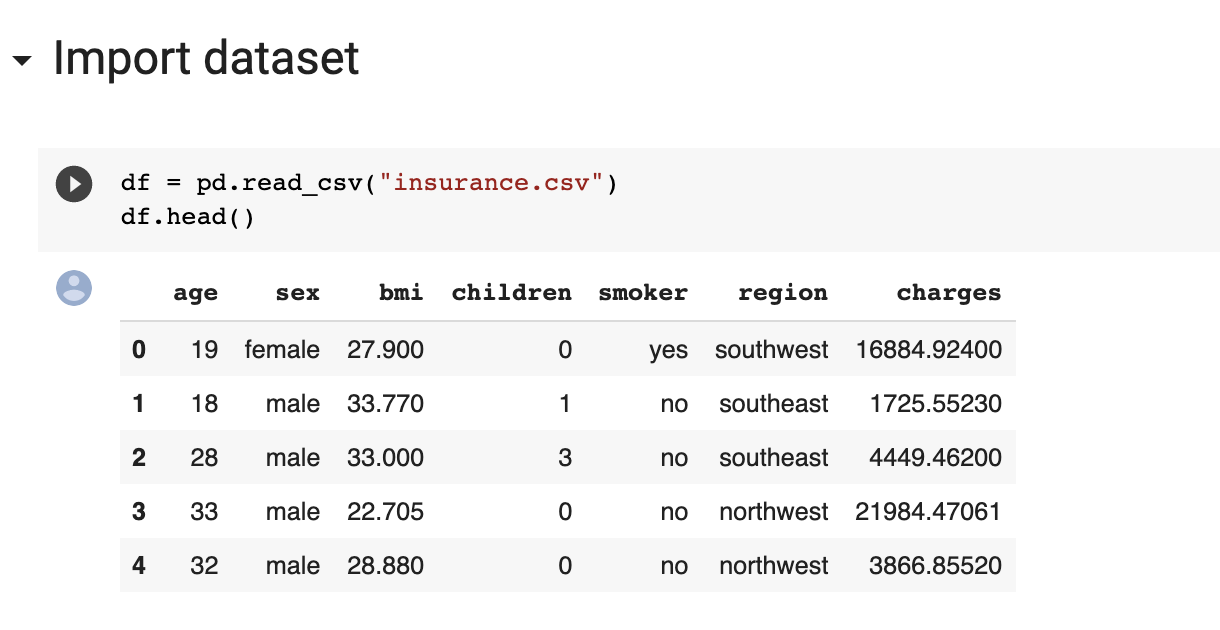


**1. Get the data**

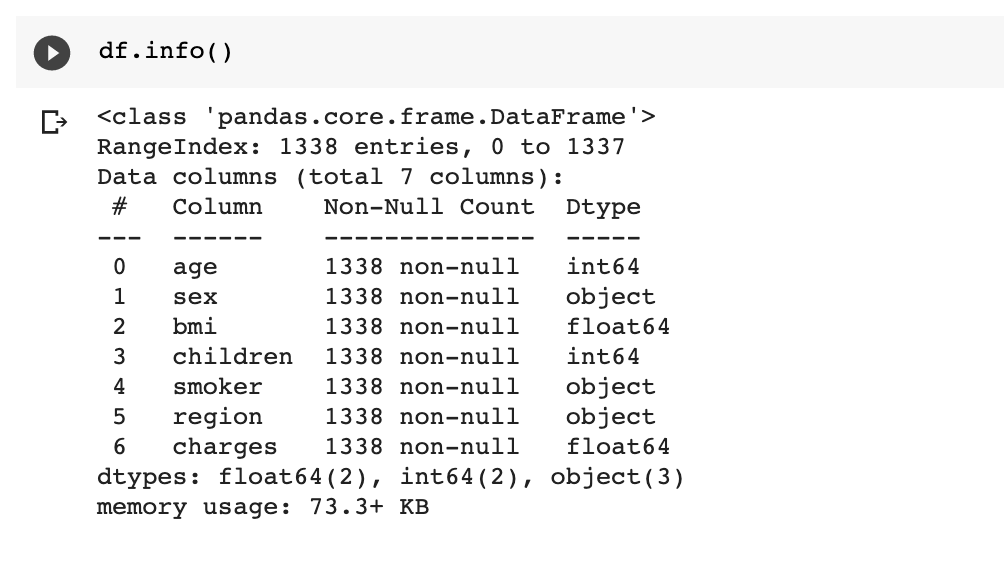
1. Upload the insurance dataset to Google Colab.



1. Import data



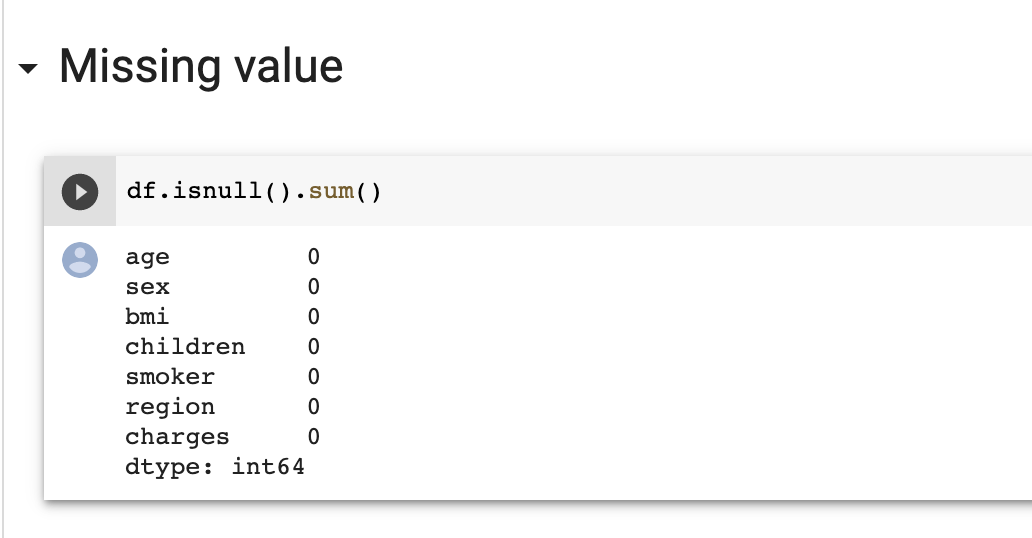
1. Get information about columns of the data.



*We can see that there are only 1338 rows in this dataset, which is quite small. The more data a model is trained on, the higher the quality of the model is. We can ensure this by not dropping null value and maneuvering more data to the training set.*

**2. Missing value**

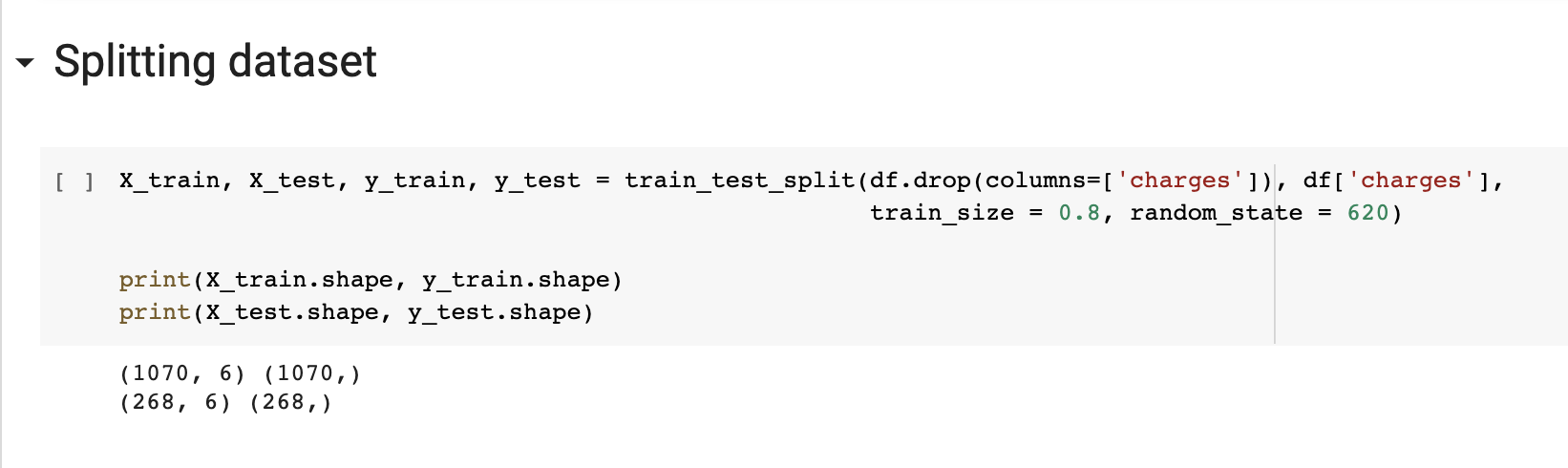
* Check for null value.



*Luckily, there isn’t any null value in this dataset.*

**3. Splitting data set**

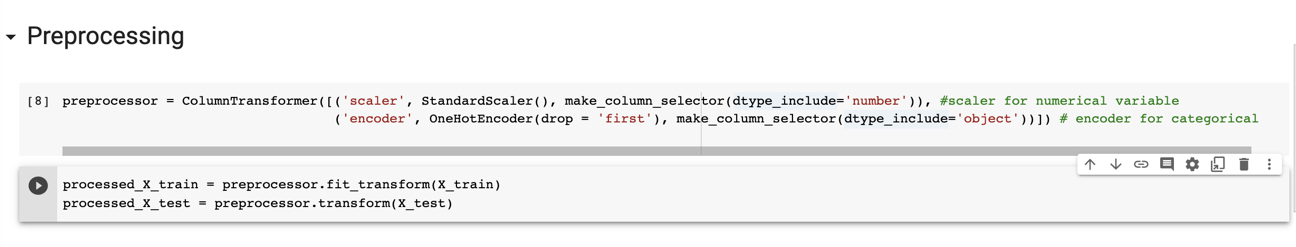
* Splitting data set and setting training proportion to 80%.



*Remember to set the random\_state parameter to make the sampling reproducible.*

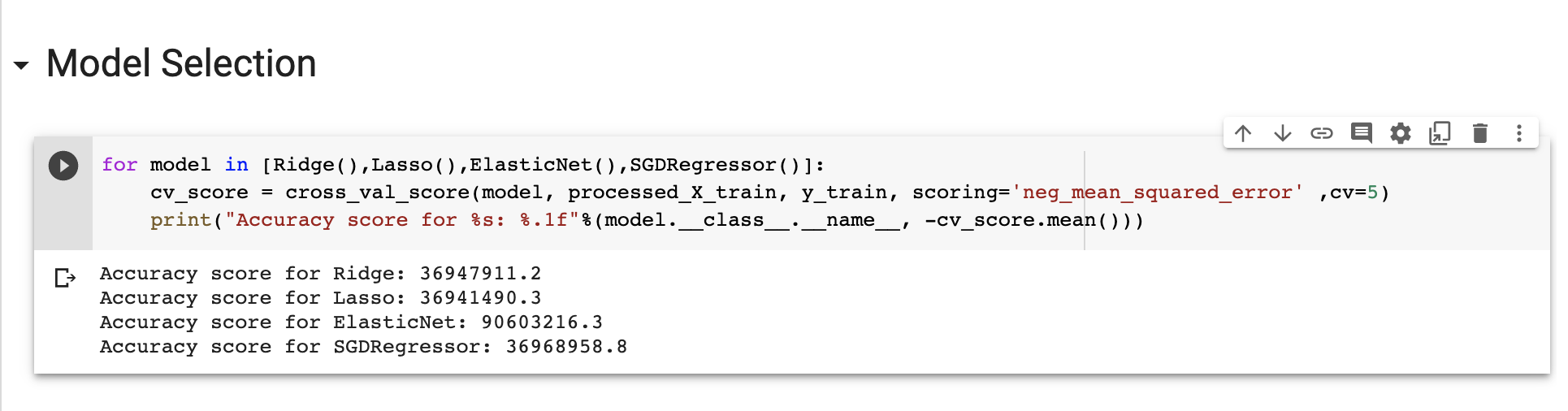
**4. Preprocessing**

* Building a preprocessor object with ColumnTransformer class.



**5. Model Selection with Cross Validation**

* Getting Cross Validation score with 4 Regression model: Ridge, Lasso, Elastic Net, Stochastic Gradient Descent Regressor. Mean squared error is one of the scoring metrics for Regression problem. Notice that we put negative mean squared error in the scoring parameter. In Regression problem, the lower the mean squared error, the better the model performs. However, the scoring functions of sklearn score the model with higher scoring value as better. Therefore, the mean squared error is negated for regression model. More scoring metrics can be found [here](https://scikit-learn.org/stable/modules/model_evaluation.html).

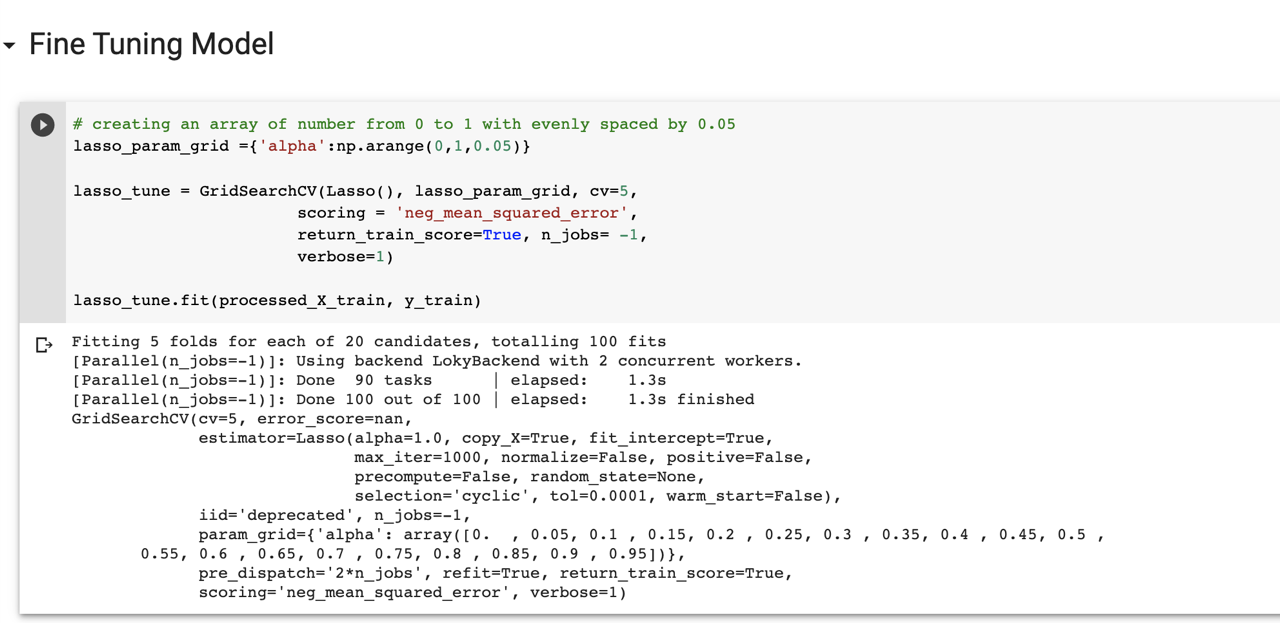


*We can see that Lasso model returns the lowest Mean squared error. Which make it the best baseline model.*

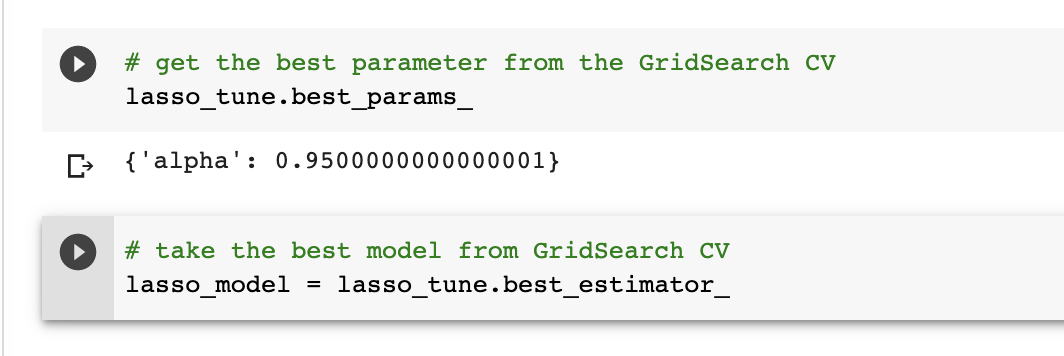
**6. Tuning parameters with Grid Search CV**

The tunning hyperparameter for the Lasso Model is alpha. It’s the constant value that’s multiplied by the regularization term, it can range from 0 to 1. We will try different values of alpha and find the one that optimizes the model performance.

1. Setting parameter grid and GridSearch CV object for Lasso model.

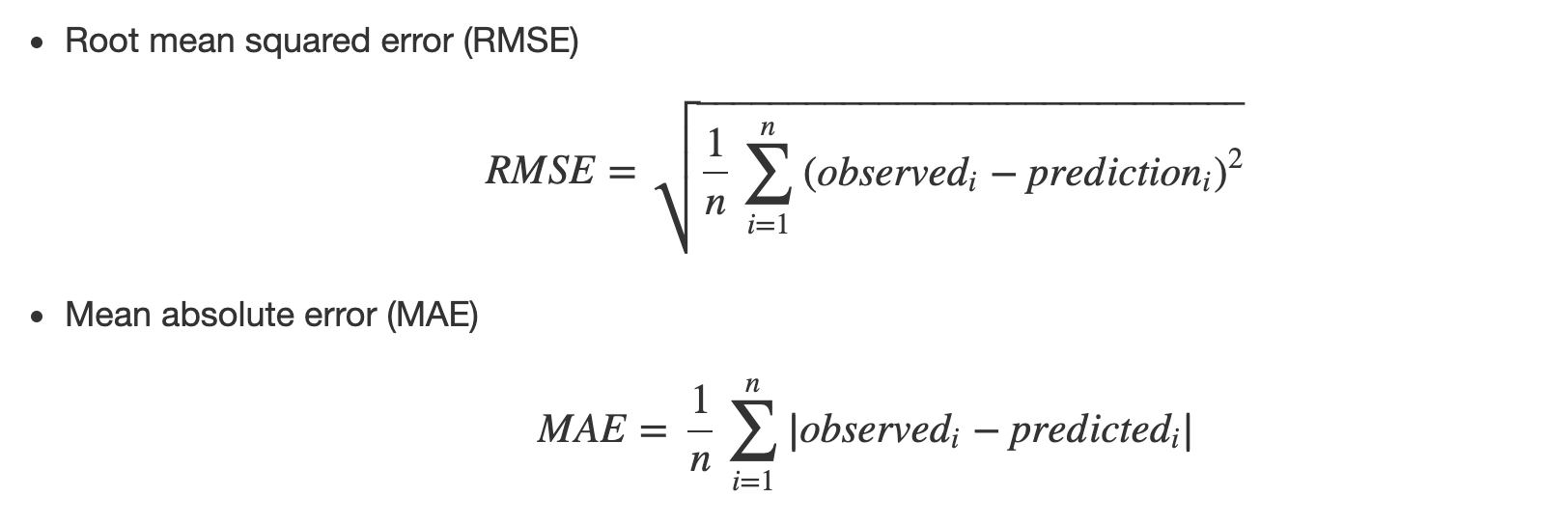


1. View the best value of alpha and get the model from the GridSearch CV.



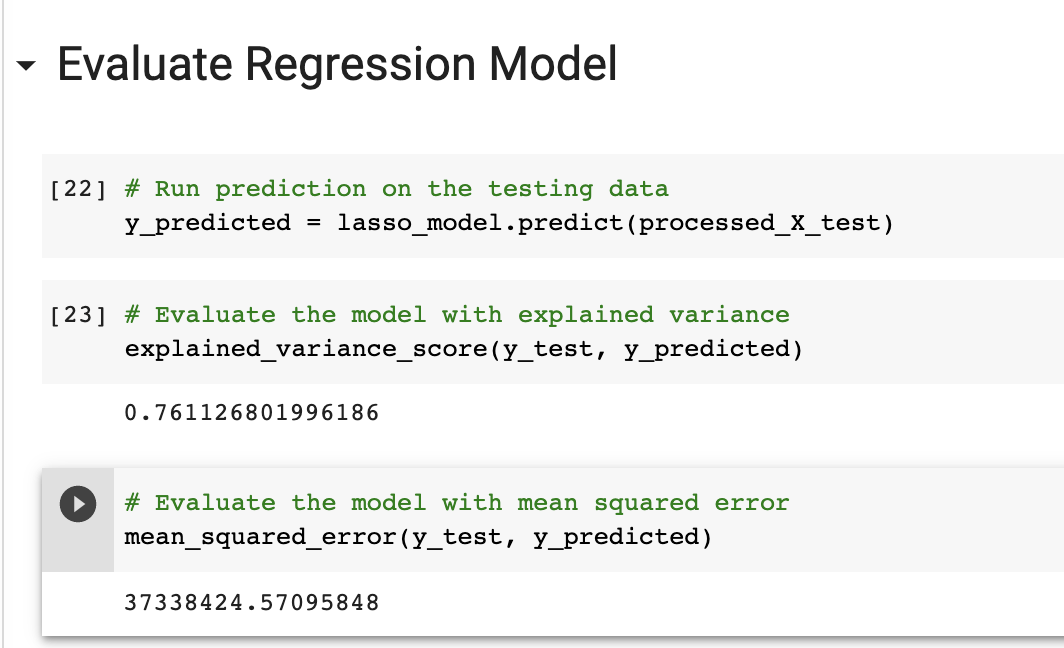
**7. Model Evaluation**

As mentioned above, The Mean Squared Error is one of the ways we can use to evaluate the performance of a regression model. It measures the distance between the true value and the predicted value, the bigger Mean Squared Error is, the less accurate the model is. The goal is to minimize this value.



You might notice that RMSE is an absolute metric, which sometimes can get very large. Another relative metric is called explained variance which as its name implies the percentage of the variance in the true value that is covered by the predicted value.

* Run the following code to use the model to make predictions on test data. Then use explained variance and mean square error to evaluate the performance on the training set.



**Push Your Work to GitHub**

**Download the notebook from Colab**:

File -> Classification.ipynb.ipynb

Move the downloaded file into your **Module3** working folder.

Open terminal and Navigate to the GitHub folder of this week HOS.

**Make sure the assignment files on the subfolder Module3 of hos03a\_YouGithubUserName folder, enter the following command to upload your work**:

>>>> git add .

>>>> git commit -m “Submission for HOS03”

>>>> git push origin master