INTRODUCTION TO MACHINE LEARNING

Assignment 1-2021AMA2090(Rahman Alam)

Contents

Non-AIP701-PART 1A 2

1. Scikit Learn Library 2
2. Design Matrix 2
3. Curve Fitting (Moore-Penrose) 2
4. Plots 6
5. Coefficient for Best and worst model and Noise variance 10
6. Conclusion-1A 11
7. Gradient Descent Algorithm 12

Non-AIP701-PART 1B 15

8. Design Matrix 15
9. Curve Fitting (Moore-Penrose) 15
10. Plots 17
11. Coefficient for best and worst model and Noise Variance 19
12. Conclusion 20
13. Gradient Descent Algorithm 1B 21

Non-AIP701-PART 2(Time series) 24

14. Cross Validation- 26
15. Package Used 27

`

NON-AIP701-PART 1A

1. Scikit Learn Library

This is a scientific library which contains various model class within itself. Least Square Regression is one part of the various class containing this library. This library helps us apply various model on classification, regression, clustering, dimensionality reduction,

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
```

These three subclasses can be used for simple polynomial regression fit and can be used to calculate mean squared error for the train-test data.

More information about the library can be access here at its *documentation.*

2. Design Matrix

Design matrix are the set of linear equations whose solution will give the polynomial coefficient being fit.
I will be using inbuilt class presented in the scikit learn library for the design matrix.

Ex-

```
polynomial_features = PolynomialFeatures(degree = i)
X_poly = polynomial_features.fit_transform(x_train)
design_matrix_20.append(X_poly)
```

Design Matrix for M=1 is shown here for 20 data points

```
.      1.24561],[ 1.       -1.016511],[1.       0.305252],[ 1.       -1.104524],[ 1.       -
0.785528],[1.       0.486322],[ 1.       -1.720929],[ 1.       -0.150237],[1.       1.065645],[ 1.       -0.272616],[
1.       -1.405794],[ 1.       -1.972946],[ 1.       -0.326951],[1.       1.006105],[1.       0.067061],[ 1.       -
1.298048],[ 1.       -1.634338],[ 1.       -0.727636],[1.       0.514267],[ 1.       -0.379221]
```

Design Matrix for M=2 is shown here for 20 data points

```
1.      1.24561],[ 1.       -1.016511],[1.       0.305252],[ 1.       -1.104524],[ 1.       -
0.785528],[1.       0.486322],[ 1.       -1.720929],[ 1.       -0.150237],[1.       1.065645],[ 1.       -0.272616],[
1.       -1.405794],[ 1.       -1.972946],[ 1.       -0.326951],[1.       1.006105],[1.       0.067061],[ 1.       -
1.298048],[ 1.       -1.634338],[ 1.       -0.727636],[1.       0.514267],[ 1.       -0.379221]
```

Same way we can make design matrix for N number of data points.

Design matrix can be generated using the command shown away which is the part of scikit learn library .Then the metric used will be pass to fit_transform to fit the required degree of polynomial using Moore Penrose methodology. Then trained model can be used to predict for unknown X(input features) using model.predict command. These all things are used in the code. For loop is used to iterate for M degree of polynomial to visualize the fitting that we will see in later part of report.

```
polynomial_features = PolynomialFeatures(degree = i)
X_poly = polynomial_features.fit_transform(x_train)
design_matrix_20.append(X_poly)
polynomial_reg_model_20=LinearRegression()
polynomial_reg_model_20.fit(X_poly,y_train)
Coefficeint_20.append(polynomial_reg_model_20.coef_)
y_predicted = polynomial_reg_model_20.predict(X_poly)
```

3. Curve Fitting (Moore-Penrose)

> *After Creating the Design matrix, we can solve the equation and get the coefficients to build the polynomial regression model. Started from 0/1 polynomial degree to M degree.*

The First 10 iterations RMSE are shown here for **20 data points** taken for training.

Root Mean Square Error for M=0:247.0541643191767

Root Mean Square Error for M=1:211.12447248939546

Root Mean Square Error for M=2:154.8615544456756

Root Mean Square Error for M=3:96.24902022347398

Root Mean Square Error for M=4:46.02383183717053

Root Mean Square Error for M=5:22.935950053185138

Root Mean Square Error for M=6:18.149825822102095

Root Mean Square Error for M=7:18.145050251430355

Root Mean Square Error for M=8:17.97025665371971

Root Mean Square Error for M=9:16.88613992844558

Root Mean Square Error for M=10:15.031923356337714

The Plot for 25-degree polynomials is shown here for the **20 data points** taken for the training. Conclusions are very interesting to see.

For the testing we have used the file namely testing.csv which contains 30 other point for the prediction on unknown input features. Plot for RMSE and M is shown below.
The First 20 iterations RMSE are shown here for **30 test data points** taken for testing.

Root Mean Square Error for test data for M=0:169.56904267380926
Root Mean Square Error for test data for M=1:244.69143622755874
Root Mean Square Error for test data for M=2:300.8214329305542
Root Mean Square Error for test data for M=3:347.8199608928004
Root Mean Square Error for test data for M=4:331.0007758364925
Root Mean Square Error for test data for M=5:319.31714487953866
Root Mean Square Error for test data for M=6:281.91381171982033
Root Mean Square Error for test data for M=7:280.6497395751493
Root Mean Square Error for test data for M=8:338.304453542702
Root Mean Square Error for test data for M=9:1839.2216328766644
Root Mean Square Error for test data for M=10:3072.433988599605
Root Mean Square Error for test data for M=11:1933.817448331964
Root Mean Square Error for test data for M=12:10444.067774794421
Root Mean Square Error for test data for M=13:118868.33901603591
Root Mean Square Error for test data for M=14:413027.2991993005
Root Mean Square Error for test data for M=15:566570.2986125293
Root Mean Square Error for test data for M=16:2520892.6938975854
Root Mean Square Error for test data for M=17:20809287.058145054
Root Mean Square Error for test data for M=18:61402409.48038547
Root Mean Square Error for test data for M=23:65289855207.476265
Root Mean Square Error for test data for M=24:2982004229.0930424

So we can see after the nearly M =6 the testing error is getting increased. So for now we can say M=6 is a good degree to go ahead with.

Error vs M for 30 Test Data Points

The First 10 iterations RMSE are shown here for **70 data** points taken for training.

Root Mean Square Error for M=0:298.8398296781834

Root Mean Square Error for M=1:237.58781279272927

Root Mean Square Error for M=2:231.6178705136389

Root Mean Square Error for M=3:98.3836525965776

Root Mean Square Error for M=4:95.48674442022723

Root Mean Square Error for M=5:25.514474057246286

Root Mean Square Error for M=6:25.51339401238804

Root Mean Square Error for M=7:19.080837936561306

Root Mean Square Error for M=8:18.896599506145428

Root Mean Square Error for M=9:18.44095661320482

Root Mean Square Error for M=10:18.426066987822978

The Plot for 25-degree polynomials is shown here for the **70 data points** taken for the training. Conclusions are very interesting to see.

Error vs M for 70 data points

For the testing we have used the file namely testing.csv which contains 30 other point for the prediction on unknown input features. Plot for RMSE and M is shown below.
The First 20 iterations RMSE are shown here for **30 test data points** taken for testing. Now the model is being trained on full data.
Root Mean Square Error for test data for M=0:107.45554812663964
Root Mean Square Error for test data for M=1:260.90570442811844
Root Mean Square Error for test data for M=2:237.8068094902698
Root Mean Square Error for test data for M=3:343.23616869111675
Root Mean Square Error for test data for M=4:337.6748943746974
Root Mean Square Error for test data for M=5:310.82291184352806
Root Mean Square Error for test data for M=6:310.884926432513
Root Mean Square Error for test data for M=7:296.2304731460998
Root Mean Square Error for test data for M=8:297.58153464321464
Root Mean Square Error for test data for M=9:301.1836247890146
Root Mean Square Error for test data for M=10:300.7678700671116
Root Mean Square Error for test data for M=11:300.36914553966596
Root Mean Square Error for test data for M=12:299.70138744206554
Root Mean Square Error for test data for M=13:298.9148612344063
Root Mean Square Error for test data for M=14:298.79834969510875
Root Mean Square Error for test data for M=15:294.10494864685194
Root Mean Square Error for test data for M=16:292.17619169917407
Root Mean Square Error for test data for M=17:291.649017251848
Root Mean Square Error for test data for M=18:279.1772117746284
Root Mean Square Error for test data for M=19:277.9642043954586

We can see that model trained with more data at least is not overshooting in error as we have seen with model on 20 data points. Keeping in mind for computational power and error we can say that 6th degree is a good choice to go-ahead with.

4. Plots

Plots for 20 data points iterations for the trainning

We can see till M=6 model is giving a good fit. Higher M may also give better result, but the computational power needed will be more. So, choosing M=6 can be good choice. We can see for M=19 model for 20 data points is getting totally overfit. For M=0,1,2,3 we can clearly see the model is showing the underfitting.

Plots for 20 data points iterations for the testing data

We can see there seems to be an outlier which is resulting in giving high value od RMSE.Removing that may help make better the model.

Plots for 70 data points iterations

Plots for 70 data points iterations for the testing data on trained model for corresponding M value.

We can clearly see there is no improvement after 6 degree polynomial which leads us to choose 6 degree polynomial as optimal model.

5. Coefficient for Best and Worst model and Noise variance

> *Both the case (20 data points and 70 data points) are giving good results for the polynomial degree 6 however less data points are getting overfit early compared to more data points.*

The Coefficients for the 6th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5….X^0, constant is given below

[19.48598671 ,-32.4384702 ,-36.82352355 ,60.83358937, 9.32876641 ,-28.66575686 ,1.3944773]

The Coefficients for the 6th degree polynomial are given here for 70 data point case. Corresponding to X^6, X^5….X^0, constant is given below

[1.14116218 ,10.62512673, 0.25552779, -2.87813669, -21.87741547,1.21642023,15.54965251]

So, we can see and compare the coefficients for less data are high.

The Coefficients for the 19th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5….X^0, constant is given below. And the noise variance for this case is [7.5767451e-08].

```
1.89838006e+03  1.89860233e+03  5.29050852e+04  8.23409178e+04

 -9.07753437e+05 -2.28226954e+06  4.74268294e+06  1.69995519e+07

 -4.82488169e+06 -5.23357147e+07 -2.42826626e+07  6.84456009e+07

  6.48568625e+07 -3.21481783e+07 -5.91758988e+07 -5.86983714e+06

  2.17355273e+07  8.89447424e+06 -1.90631899e+06 -1.79301770e+06

 -2.95161580e+05
```

The Coefficients for the 19th degree polynomial are given here for 70 data point case. Corresponding to X^6, X^5….X^0, constant is given below

```
 0.00000000e+00 -1.46777661e+01  1.42125328e+02 -1.37632939e+02

 -5.24993917e+02  9.71663726e+02  4.24941255e+00 -2.16817196e+03

  2.47790028e+03  2.52024805e+03 -4.61082676e+03 -1.78189342e+03

  3.98741644e+03  8.00475188e+02 -1.91655946e+03 -2.21190935e+02

  5.24313345e+02  3.39278727e+01 -7.63274641e+01 -2.19158723e+00

  4.58815342e+00
```

So, we can again see that less data points in the overfitting case have overshooted constants.

6. Conclusion-1A

Plotting the Graph together gives a lot of insights

1. We can see for both fewer data (20) and Full data (70) the optimal M value is 6.

2. The Maximum error for more data is more which supposed to be high for more data as then the data will have more chances of being error at early stages.

3. We can observe that fewer data got over fitted at just for M=19 but the more data has saved us at that point.

Even more data is not overfitted (considering error close to 0) at M=25.

So, we can say that more data can save us from overfitting a model.

7. Gradient Descent Algorithm

- Stochastic gradient descent

Here we are updating the weights for every sample, so the convergence is noisy but computationally effective. We are using MSE as our loss function and a function defined to give a Mth order polynomial. No regularization is used. I have shown plots for 1000, 2000,…10000 epochs and we can see that training loss is almost **constant after 4000** epochs so we can just **stop till 4000 epochs**. Learning rate I have taken as 0.001.test data with 30 points have been used to test the models after every 1000 epochs. M=6 has chosen here for SGD. So more epochs are helping reducing error but upto 4000 epochs.
Plots are shown below.

Model
Training loss
RMSE loss
Model_test prediction

- Mini Batch Gradient Descent

  Here we will be updating the weights after few samples are passed or you can say after a handful of samples called mini batches is passed. It will help us smoothly enter towards the optimal points/weights.

  Model is trained for 6th order polynomial

So as above graph suggest to pick the learning rate at 0.001 and a optimal batch size to be 4 as our data set is onjly having 70 data points for training.

Weights for the 6th order polynomial for 0.001 learning rate,4 batch size and 4000 epoch with 70 data points are as below.

2.25212651 ,-10.81834511 , 10.53234425 , -4.54626229 , 7.38617364 , 5.95717802 , -1.94808232

NON-AIP701-PART 1B

8. Design Matrix

Design Matrix for M=1 is shown here for 20 data points

```
[          0.246119    0.06057456],[        0.87242     0.76111666],[         0.2192      0.04804864],[1.809242    3.27335661],[
         -0.923474     0.85280423],[        1.006063    1.01216276],[          -1.41848      2.01208551],[-
1.24184      1.54216659],[       -1.278768    1.6352476],[ -
0.32844      0.10787283],[     0.599934   0.3599208],[       1.248036    1.55759386],[  1.645741    2.70846344],[       0.990873
 0.9818293],[1.          1.681223    2.82651078],[1.          0.242633    0.05887077],[       1.38482     1.91772643],[
-1.858806     3.45515975],[         -0.559941     0.31353392],[-1.318246     1.73777252]
```

Design Matrix for M=2 is shown here for 20 data points

```
[         0.246119    0.06057456 0.01490855],[        0.87242     0.76111666 0.66401339],[          0.2192      0.04804864
0.01053226],[          1.809242    3.27335661 5.92229427],[          -0.923474     0.85280423 -
0.78754253],[          1.006063    1.01216276 1.0182995 ],[           -1.41848      2.01208551 -2.85410305],[          -
1.24184      1.54216659 -1.91512415],[          -1.278768    1.6352476 -2.0911023],[          -0.32844      0.10787283 -
0.03542975],[          0.599934    0.3599208  0.21592873],[          1.248036    1.55759386
1.94393321],[          1.645741    2.70846344
4.45742933],[          0.990873    0.9818293  0.97286815],[          1.681223    2.82651078
4.75199493],[          0.242633    0.05887077 0.01428399],[          1.38482     1.91772643 2.65570592],[          -
1.858806     3.45515975 -6.42247167],[          -0.559941     0.31353392 -0.1755605 ],[          -1.318246     1.73777252 -
2.29081167]
```

9. Curve Fitting (Moore-Penrose)

The First 10 iterations RMSE are shown here for 20 data points taken for training.

Root Mean Square Error for M=0:55.62287520856701

Root Mean Square Error for M=1:53.77283866902993

Root Mean Square Error for M=2:28.491239299489813

Root Mean Square Error for M=3:20.993855995451245

Root Mean Square Error for M=4:6.025820376031074

Root Mean Square Error for M=5:5.2297372513875064

Root Mean Square Error for M=6:5.089281088633635

Root Mean Square Error for M=7:4.698322039383037

Root Mean Square Error for M=8:4.676583516582949

Root Mean Square Error for M=9:4.669799870651431

Root Mean Square Error for M=10:4.616500904873091

The Plot for 25-degree polynomials is shown here for the 20 data points taken for the training. Conclusions are very interesting to see.

Error vs M for 20 Data Points

The First 10 iterations RMSE are shown here for 70 data points taken for training.

Root Mean Square Error for M=0:70.09721538978584

Root Mean Square Error for M=1:67.22070491028734

Root Mean Square Error for M=2:33.696388497087085

Root Mean Square Error for M=3:25.181823338879145

Root Mean Square Error for M=4:8.752391253783841

Root Mean Square Error for M=5:8.455902961555978

Root Mean Square Error for M=6:7.842903432512381

Root Mean Square Error for M=7:7.812639922861118

Root Mean Square Error for M=8:7.638369766057252

Root Mean Square Error for M=9:7.636777019339523

Root Mean Square Error for M=10:7.62929718421274

The Plot for 25-degree polynomials is shown here for the 70 data points taken for the training. Conclusions are very interesting to see

10. Plots

Plots for 20 data points iterations

We can see till M=6 model is giving a good fit. Higher M may also give better result, but the computational power needed will be more. So, choosing M=6 can be good choice. We can see for M=19 model for 20 data points is getting totally overfit. For M=0,1,2,3 we can clearly see the model is showing the underfitting.so for the in 1A and 1B model is getting overfit at M=19 for fewer (20) data points.

Plots for 70 data points iterations

We can see till M=6 model is giving a good fit. Higher M may also give better result, but the computational power needed will be more. So, choosing M=6 can be good choice. We can see for M=19 model for 70 data points is not getting overfit.

11. Coefficient for best and worst model and Noise Variance

> *Both the case (20 data points and 70 data points) is giving good results for the polynomial degree 6 however less data points are getting overfit early compared to more data points.*

The Coefficients for the 6th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5….X^0, constant is given below

-4.45736087 -13.12884113 26.66689349 17.32522203 -21.60795539 0.36253482 3.23142133

The Coefficients for the 6th degree polynomial are given here for 70 data point case. Corresponding to X^6, X^5….X^0, constant is given below

13.53666303 1.38512963 -14.09892717 4.92232874 1.92625672 2.70603127 -0.58152941

So, we can see and compare the coefficients for less data are high. Resulting early overfit for fewer data.

The Coefficients for the 19th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5….X^0, constant is given below. And the noise variance for this case is [3.21873423e-11].

```
                0          23445.74407486    21340.3024207   -378379.81275263
 241346.22046543  1817390.01978246 -1969987.81349591 -3693079.60571943
5147336.39235743  3590483.98839817 -6643345.86394168 -1478845.35574118
4758062.17249966  -115855.53950881 -1922688.52994446   330160.90240019
 408914.49765016  -109282.22189523   -35369.66056063    11746.49232729
```

The Coefficients for the 19th degree polynomial are given here for 70 data point case. Corresponding to X^6, X^5….X^0, constant is given below

```
0.00000000e+00  2.82510067e+01 -3.06776908e+01 -2.13878112e+02
4.13728110e+02  9.13341806e+02 -1.16099300e+03 -1.90363058e+03
1.49306462e+03  2.12229516e+03 -1.03810150e+03 -1.36974022e+03
 4.17399713e+02  5.27792990e+02 -9.70140376e+01 -1.19701269e+0
1.20982340e+01  1.47340123e+01 -6.26388966e-01 -7.58955625e-01
```

So, we can again see that less data points in the overfitting case have overshooted constants. Resulting early overfitting.

12. Conclusion

Plotting the Graph together gives a lot of insights

1. We can see for both fewer data (20) and Full data (70) the optimal M value is 6. Observed in both 1A and 1B.

2. The Maximum error for more data is more which supposed to be high for more data as then the data will have more chances of being error at early stages. Observed in both 1 A and 1B

3. We can observe that fewer data got over fitted at just for M=19 but the more data has saved us at that point. Observed in both 1A and 1B.

Even more data is not overfitted (considering error close to 0) at M=25.

So, we can say that more data can save us from overfitting a model.

13. Gradient Descent Algorithm 1B

- Stochastic gradient descent

Here we are updating the weights for every sample, so the convergence is noisy but computationally effective. We are using MSE as our loss function and a function defined to give a Mth order polynomial. No regularization is used. I have shown plots for 1000, 2000…10000 epochs and we can see that training loss is almost **constant after 400** epochs so we can just **stop till 400 epochs**. Learning rate, I have taken as 0. 001.Train data with 56 points are used to train the model. Test data with 14 points have been used to test the models after every 1000 epochs. M=6 has chosen here for SGD. So, more epochs are helping reducing error but up to 400epochs we will be getting the maximum error reduction so we can choose **400 as stopping criteria.** Data points are in blue and predicted points are in red. The plots for various epochs till 10000 are shown below. In problem 1A model converged at around 4000 epochs comparably 10 times taken more epochs to converge t

- Mini Batch Gradient Descent

  Here we will be updating the weights after few samples are passed or you can say after a handful of samples called mini batches is passed. It will help us smoothly enter towards the optimal points/weights.

  Model is trained for 5th order polynomial

So as above graph suggest picking the learning rate at 0.001 and a optimal batch size to be 4 as our data set is only having 70 data points for training.

Weights for the 5th order polynomial for 0.001 learning rate,4 batch size and 400 epoch with 70 data points are as below.

[[10.55692608], [2.6134792], [9.2856239], [0.94688129], [11.65410628], [-2.47706711]]

Noise is distributed about 0 with very low variance as 71.35 for 6th degree polynomial.

NON-AIP701-PART 2(TIME SERIES)

The data for this problem is containing two parts as training and testing. Training part is having a column name id which contains the data in date format, so we need to first convert that in workable format for that I have done feature engineering to convert that in reasonable number to be work as an input value.

Initial Training Data looks like a typical time series with so many ups and downs as shown below.

Process done to convert date time to input number

```
data_test['id'].dtype
data_test['datetime_converted'] = pd.to_datetime(data_test['id'], format="%m/%d/%y")
data_test['day'] = data_test['datetime_converted'].dt.day
data_test['month'] = data_test['datetime_converted'].dt.month
data_test['year'] = data_test['datetime_converted'].dt.year
data_test['year_converted'] = data_test['year']-2000

data_test['Y'] = data_test['year']+data_test['month']*12-2000
final_data_test
=data_test.drop(['id','datetime_converted','day','month','year','year_converted',],axis=1)
```

So, we have now converted our input a reasonable input for the training of the model. Same it has done for test sata as well.

So, our input now converted to this type of input.

We have fitted a regression model with 2 degrees of polynomial. Which is reasonably giving low RMSE.

Root Mean Square Error for M=2:5.28

Prediction for the test data is as below.

id	Prediction
05-01-2010	20.03105
04-01-2009	18.29508
09-01-2013	12.51772
01-01-2006	4.412854
02-01-2007	10.48598
08-01-2012	16.56464
06-01-2014	20.3213
03-01-2008	15.11339
12-01-2004	-8.29733
07-01-2011	19.16583

14. Cross Validation-

Cross Validation is used as K-fold. For which the K value is taken to be 10,along with degree and scoring hyperparameter as till 20 and negative mean square error.

With the grid search CV the out degree of the linear regression basis function come to be 8. Whose constants and predicted values are given below as.

Optimal Polynomial Degree: {'deg': 6}

Coefficients: [-8.26573505e-04 2.70928709e-02 -2.89954858e-01 9.45745201e-01

7.83805091e-01 -8.79547673e-01 2.07623188e+00]

Predicted output after grid search CV

id	Prediction_CV
05-01-2010	26.02
04-01-2009	21.75563
09-01-2013	5.23483
01-01-2006	2.662546
02-01-2007	7.193112
08-01-2012	12.86773
06-01-2014	25.62495
03-01-2008	14.52161
12-01-2004	-0.42863
07-01-2011	20.63917

id	Prediction	Prediction_CV
05-01-2010	20.03105	26.02
04-01-2009	18.29508	21.75563
09-01-2013	12.51772	5.23483
01-01-2006	4.412854	2.662546
02-01-2007	10.48598	7.193112
08-01-2012	16.56464	12.86773
06-01-2014	20.3213	25.62495
03-01-2008	15.11339	14.52161
12-01-2004	-8.29733	-0.42863
07-01-2011	19.16583	20.63917

Real time Leader board rating for Normal linear regression is=**4.44585**

Real time Leader board rating for CV linear regression is=**2.31564**

15. Package Used

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from scipy.optimize import approx_fprime as gradient
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.base import BaseEstimator
import math
import csv
etc
```

16. References

- python - How to extract only datein ddmmyy format from a timestamp - Stack Overflow
- How to Concatenate Two Pandas DataFrames (With Examples) - Statology
- Polynomial Regression with Cross-Validation – Ka'imi Kahihikolo (kaimi.dev)
- Polynomial Regression with K-fold Cross-Validation | by Aamir Ahmad Ansari | Medium
- sklearn.metrics.r2_score — scikit-learn 1.1.2 documentation