LAPORAN TUGAS

**LINEAR REGRESSION**

**Mata Kuliah:**

**MACHINE LEARNING**

**Genap 2019/2020**

**Dosen Pengampu:**

**Dr., INDAH AGUSTIEN SIRADJUDDIN, S.Kom., M.Kom**

**Oleh :**

**[Miftahul Choir](https://new.edmodo.com/user/108671307/)**

**[160411100055](https://new.edmodo.com/user/108671307/)**

**[Dennis Thandy Nur Cahyono](https://new.edmodo.com/user/21129755/)**

**[160411100079](https://new.edmodo.com/user/21129755/)**

**[Nurrachmad Widyanto](https://new.edmodo.com/user/24931618/)**

**[160411100179](https://new.edmodo.com/user/24931618/)**

**[Muhammad Yusqi Alfan Thoriq](https://new.edmodo.com/user/108909389/)**

**[160411100187](https://new.edmodo.com/user/108909389/)**

****

**PROGRAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**BANGKALAN**

**2019**

1. **Deskripsi Dataset**

Dataset yang digunakan pada metode Linear Regression ini adalah data diabetes dari library sklearn dengan printah *sklearn.datasets.load\_diabetes()* yang memiliki jumlah fitur 10 dengan target memprediksi tingkat resiko penyakit diabetes dari pasien yang bertujuan untuk mengetahui seberapa besar resiko diabetes yang bisa terjadi pada pasien.

1. **Penjelasan Kode Program**
2. **Kode Scikit**

|  |
| --- |
| import numpy as np  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import r2\_score  def splitData(dataset, p=100):  if p >= 100:  dataTrain = dataset.data  target = dataset.target  dataTest = dataset.data  targetTest = dataset.target  else:  jumlahDataTrain = int((p \* dataset.data.shape[0]) / 100)  # data train  dataTrain = dataset.data[:jumlahDataTrain]  target = dataset.target[:jumlahDataTrain]  # data test  dataTest = dataset.data[jumlahDataTrain:]  targeTest = dataset.target[jumlahDataTrain:]  return dataTrain, target, dataTest, targeTest  # load data  dataset = datasets.load\_diabetes()  # split Data  dataTrain, target, dataTest, targeTest = splitData(dataset, p=80)  # scikitLearn  reg = LinearRegression().fit(dataTrain, target)  predictSK = reg.predict(dataTest)  # score scikit  print(r2\_score(targeTest, predictSK)) |

1. **Penjelasan Bagian Kode Program**

|  |
| --- |
| def splitData(dataset, p=100):  if p >= 100:  dataTrain = dataset.data  target = dataset.target  dataTest = dataset.data  targetTest = dataset.target  else:  jumlahDataTrain = int((p \* dataset.data.shape[0]) / 100)  # data train  dataTrain = dataset.data[:jumlahDataTrain]  target = dataset.target[:jumlahDataTrain]  # data test  dataTest = dataset.data[jumlahDataTrain:]  targeTest = dataset.target[jumlahDataTrain:]  return dataTrain, target, dataTest, targeTest |

Bagian fungsi diatas bernama *slpitData()* yang yang melakukan proses membagi data training, data target training, data testing, dan data target testing.

|  |
| --- |
| # load data  dataset = datasets.load\_diabetes()  # split Data  dataTrain, target, dataTest, targeTest = splitData(dataset, p=80) |

Bagian kode diatas merupakan perintah untuk mengambil dataset dari library *sklearn.datasets* kemudian dengan memanggil fungsi splitData dataset tersebut dibagi menjadi data training, data target training, data testing, dan data target testing.

|  |
| --- |
| # scikitLearn  reg = LinearRegression().fit(dataTrain, target)  predictSK = reg.predict(dataTest) |

Bagian kode diatas merupakan langkah membuat model dengan fungsi *LinearRegression().fit()* dari library *sklearn* dan melakukan proses testing dengan perintah *predict()*

|  |
| --- |
| # score scikit  print(r2\_score(targeTest, predictSK)) |

Bagian kode diatas merupakan perintah untuk menghitung presentase akurasi dari model yang telah dibuat dengan fungsi *score()* dari library *sklearn.*

1. **Kode Manual Incremental Learning**

|  |
| --- |
| import numpy as np  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import r2\_score  diabetes = datasets.load\_diabetes()  x\_train, x\_test, y\_train, y\_test = train\_test\_split(diabetes.data,  diabetes.target,  test\_size=0.4,  random\_state=0)  def buildMatrix(x):  [numOfData,feature]=x.shape  newX=np.ones((numOfData,feature+1))  newX[:,1:]=x  return newX  def stochasticGD(x,y,epoch,etha):  [numOfData,feature]=x.shape  weight=np.zeros((epoch,feature+1)) #+1 for bias  tempWeight=np.random.randn(1,feature+1)  tempData=buildMatrix(x)  for i in range(epoch):  for j in range(numOfData):  prediction=tempWeight.dot(tempData[j])  error=prediction - y[j]  tempWeight=tempWeight-etha\*error\*tempData[j]  weight[i]=tempWeight  return weight  w=stochasticGD(x\_train,y\_train,10,0.02)  print(w) |

1. **Penjelasan Bagian Kode Program**

|  |
| --- |
| diabetes = datasets.load\_diabetes()  x\_train, x\_test, y\_train, y\_test = train\_test\_split(diabetes.data,  diabetes.target,  test\_size=0.4,  random\_state=0) |

Bagian kode diatas merupakan perintah untuk mengambil dataset yang akan digunakan dengan perintah *sklearn.datasets* kemudian data dibagi menjadi empat bagian yaitu *x\_train, x\_test, y\_train,* dan *y\_test.*

|  |
| --- |
| def buildMatrix(x):  [numOfData,feature]=x.shape  newX=np.ones((numOfData,feature+1))  newX[:,1:]=x  return newX |

Bagian fungsi diatas bernama *buildMatrix()* langkah menambahkan nilai 1 pada matriks data training untuk menyamakan dengan ukuran matriks bobot dengan perintah *np.ones().*

|  |
| --- |
| def stochasticGD(x,y,epoch,etha):  [numOfData,feature]=x.shape  weight=np.zeros((epoch,feature+1)) #+1 for bias  tempWeight=np.random.randn(1,feature+1)  tempData=buildMatrix(x)  for i in range(epoch):  for j in range(numOfData):  prediction=tempWeight.dot(tempData[j])  error=prediction - y[j]  tempWeight=tempWeight-etha\*error\*tempData[j]  weight[i]=tempWeight  return weight |

Bagian fungsi diatas bernama *stochasticGD()* yang yang melakukan proses menghitung bobot dari setiap data berdasarkan *epoch* yang telah ditentukan

|  |
| --- |
| w=stochasticGD(x\_train,y\_train,100,0.02)  print(w) |

Bagian kode diatas merupakan perintah memanggil fungsi *stochasticGD()*  yang kemudian mengeluarkan hasil berupa bobot yang telah dihitung

1. **Kode Manual Direct Equation**

|  |
| --- |
| import numpy as np  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import r2\_score  def linearRegression(x, y):  xt = np.transpose(x)  xinv = np.linalg.inv(np.dot(xt, x))  temp = np.dot(xinv, xt)  w = np.dot(temp, y)  return w  def predict(w, data):  temp = w[1:] \* data  temp1 = np.concatenate((w[0],temp), axis=None)  return sum(temp1)  def predicts(w, datas):  jumlahBaris = datas.shape[0]  result = np.zeros(jumlahBaris)  for i in range(jumlahBaris):  result[i] = predict(w, datas[i])  return result  def R2(target, predict):  sstot = sum((target - np.average(target)) \*\* 2)  ssres = sum((target - predict) \*\* 2)  result = 1 - (ssres/sstot)  return(result)  def splitData(dataset, p=100):  if p >= 100:  dataTrain = dataset.data  target = dataset.target  dataTest = dataset.data  targetTest = dataset.target  else:  jumlahDataTrain = int((p \* dataset.data.shape[0]) / 100)  # data train  dataTrain = dataset.data[:jumlahDataTrain]  target = dataset.target[:jumlahDataTrain]  # data test  dataTest = dataset.data[jumlahDataTrain:]  targeTest = dataset.target[jumlahDataTrain:]  return dataTrain, target, dataTest, targeTest  # load data  dataset = datasets.load\_diabetes()  # split Data  dataTrain, target, dataTest, targeTest = splitData(dataset, p=80)  # menambah matrix ones pada data train  ones = np.ones((dataTrain.shape[0], 1))  dataTrain1 = np.concatenate((ones, dataTrain), axis=1)  # manual  w = linearRegression(dataTrain1, target)  predict = predicts(w, dataTest)  # score manual  print(R2(targeTest, predict)) |

1. **Penjelasan Bagian Kode Program**

|  |
| --- |
| def linearRegression(x, y):  xt = np.transpose(x)  xinv = np.linalg.inv(np.dot(xt, x))  temp = np.dot(xinv, xt)  w = np.dot(temp, y)  return w |

Bagian fungsi diatas bernama *linearRegression()* yang yang melakukan proses menghitung bobot dari data dan target.

|  |
| --- |
| def predict(w, data):  temp = w[1:] \* data  temp1 = np.concatenate((w[0],temp), axis=None)  return sum(temp1) |

Bagian fungsi diatas bernama *predict()* yang melakukan proses menghitung tebakan target dari setiap data

|  |
| --- |
| def predicts(w, datas):  jumlahBaris = datas.shape[0]  result = np.zeros(jumlahBaris)  for i in range(jumlahBaris):  result[i] = predict(w, datas[i])  return result |

Bagian fungsi diatas bernama *predicts()* yang yang melakukan proses menghitung tebakan target dari semua data

|  |
| --- |
| def R2(target, predict):  sstot = sum((target - np.average(target)) \*\* 2)  ssres = sum((target - predict) \*\* 2)  result = 1 - (ssres/sstot)  return(result) |

Bagian fungsi diatas bernama *R2()* yang yang melakukan proses menghitung akurasi dari data target dan tebakan

|  |
| --- |
| def splitData(dataset, p=100):  if p >= 100:  dataTrain = dataset.data  target = dataset.target  dataTest = dataset.data  targetTest = dataset.target  else:  jumlahDataTrain = int((p \* dataset.data.shape[0]) / 100)  # data train  dataTrain = dataset.data[:jumlahDataTrain]  target = dataset.target[:jumlahDataTrain]  # data test  dataTest = dataset.data[jumlahDataTrain:]  targeTest = dataset.target[jumlahDataTrain:]  return dataTrain, target, dataTest, targeTest |

Bagian fungsi diatas bernama *slpitData()* yang yang melakukan proses membagi data training, data target training, data testing, dan data target testing.

|  |
| --- |
| # load data  dataset = datasets.load\_diabetes()  # split Data  dataTrain, target, dataTest, targeTest = splitData(dataset, p=80) |

Bagian kode diatas merupakan perintah untuk mengambil dataset dari library *sklearn.datasets* kemudian dengan memanggil fungsi splitData dataset tersebut dibagi menjadi data training, data target training, data testing, dan data target testing.

|  |
| --- |
| # menambah matrix ones pada data train  ones = np.ones((dataTrain.shape[0], 1))  dataTrain1 = np.concatenate((ones, dataTrain), axis=1) |

Bagian kode diatas merupakan langkah menambahkan nilai 1 pada matriks data training untuk menyamakan dengan ukuran matriks bobot dengan perintah *np.ones()* kemudian *np.concatenate()* untuk menggabungkan nilai 1 dengan matriks data training.

|  |
| --- |
| # manual  w = linearRegression(dataTrain1, target)  predict = predicts(w, dataTest) |

Bagian kode diatas merupakan perintah untuk menghitung bobot dengan fungsi *linearRegression()* dan memprediksi tebakan dengan nilai bobot dan data test dengan fungsi *predicts()*

|  |
| --- |
| # score manual  print(R2(targeTest, predict)) |

Bagian kode diatas merupakan perintah untuk menghitung presentase akurasi dari model yang telah dibuat dengan fungsi *R2().*

1. **Hasil**

Setelah uji coba dengan dataset diabetes menggunakan metode Linear Regression diperoleh akurasi

* 0,54 akurasi menggunakan program Linear Regression Scikit
* 0,522 akurasi menggunakan program Linear Regression Incremental Learning (Stochastic)
* 0,54 akurasi menggunakan program Linear Regression Direct Equation

1. **Perhitungan Manual**

Perhitungan manual dilakukan pade excel dengan menggunakan dataset diabetes yaitu hanya 10 data dan 5 fitur. Dapat dilihat pada tabel berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X= | age | sex | bmi | bp | s1 |
| 1 | 0,038075906 | 0,050680119 | 0,061696207 | 0,021872355 | -0,044223498 |
| 1 | -0,001882017 | -0,044641637 | -0,051474061 | -0,026327835 | -0,008448724 |
| 1 | 0,085298906 | 0,050680119 | 0,044451213 | -0,005670611 | -0,045599451 |
| 1 | -0,089062939 | -0,044641637 | -0,011595015 | -0,036656447 | 0,012190569 |
| 1 | 0,00538306 | -0,044641637 | -0,036384692 | 0,021872355 | 0,003934852 |
| 1 | -0,092695478 | -0,044641637 | -0,04069594 | -0,019442093 | -0,06899065 |
| 1 | -0,045472478 | 0,050680119 | -0,047162813 | -0,015999223 | -0,04009564 |
| 1 | 0,063503676 | 0,050680119 | -0,001894706 | 0,066629674 | 0,090619882 |
| 1 | 0,041708445 | 0,050680119 | 0,061696207 | -0,040099317 | -0,013952536 |
| 1 | -0,070900247 | -0,044641637 | 0,039062153 | -0,033213576 | -0,012576583 |

Table 1. Data Diabetes 10 data dan 5 fitur

|  |
| --- |
| Y= |
| 151 |
| 75 |
| 141 |
| 206 |
| 135 |
| 97 |
| 138 |
| 63 |
| 110 |
| 310 |

Table 2. target

XT =

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0,038075906 | -0,001882017 | 0,085298906 | -0,089062939 | 0,00538306 | -0,092695478 | -0,045472478 | 0,063503676 | 0,041708445 | -0,070900247 |
| 0,050680119 | -0,044641637 | 0,050680119 | -0,044641637 | -0,044641637 | -0,044641637 | 0,050680119 | 0,050680119 | 0,050680119 | -0,044641637 |
| 0,061696207 | -0,051474061 | 0,044451213 | -0,011595015 | -0,036384692 | -0,04069594 | -0,047162813 | -0,001894706 | 0,061696207 | 0,039062153 |
| 0,021872355 | -0,026327835 | -0,005670611 | -0,036656447 | 0,021872355 | -0,019442093 | -0,015999223 | 0,066629674 | -0,040099317 | -0,033213576 |
| -0,044223498 | -0,008448724 | -0,045599451 | 0,012190569 | 0,003934852 | -0,06899065 | -0,04009564 | 0,090619882 | -0,013952536 | -0,012576583 |

Table 3. X di Transpose

XT\*X =

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | -0,066043165 | 0,030192411 | 0,017698552 | -0,067034718 | -0,12714178 |
| -0,066043165 | 0,038149759 | 0,020403066 | 0,012674834 | 0,011224452 | 0,00766073 |
| 0,030192411 | 0,020403066 | 0,022806751 | 0,010532808 | 0,005540764 | 0,000599815 |
| 0,017698552 | 0,012674834 | 0,010532808 | 0,019106556 | -0,000270041 | -0,001430123 |
| -0,067034718 | 0,011224452 | 0,005540764 | -0,000270041 | 0,010810386 | 0,008150938 |
| -0,12714178 | 0,00766073 | 0,000599815 | -0,001430123 | 0,008150938 | 0,019202678 |

Table 4. X yang telah ditranspose di kali dengan X awal

inv(XT\*X) =

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0,111632064 | 0,235422059 | -0,454543134 | 0,035818078 | 0,268341426 | 0,548164474 |
| 0,235422059 | 74,08733888 | -41,82151134 | -27,6056065 | -48,5979159 | -8,119012703 |
| -0,454543134 | -41,82151134 | 99,39069391 | -25,52319887 | -25,75347116 | 19,6008845 |
| 0,035818078 | -27,6056065 | -25,52319887 | 85,28098683 | 44,45091814 | -0,469312764 |
| 0,268341426 | -48,5979159 | -25,75347116 | 44,45091814 | 205,7148098 | -62,04020628 |
| 0,548164474 | -8,119012703 | 19,6008845 | -0,469312764 | -62,04020628 | 84,63142154 |

Table 5. hasil perkalian di-Inverse

inv(XT\*X)\*XT =

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0,08139702 | 0,117940704 | 0,083751506 | 0,107386915 | 0,13991388 | 0,065608061 | 0,049928999 | 0,171032167 | 0,082216099 | 0,100824648 |
| -1,470215441 | 4,732015046 | 3,854171551 | -2,493494459 | 2,410742901 | -2,136741211 | -2,848008043 | -1,100781529 | 1,564833462 | -2,512522277 |
| -0,014596215 | -2,98658422 | -0,867028972 | 0,312158901 | -4,674142841 | -0,827727175 | 7,314188154 | 2,035412747 | 2,022808865 | -2,314489245 |
| 3,945705128 | -4,328920418 | -0,052248815 | 1,009883717 | -1,105910073 | -0,568300344 | -4,716859309 | -0,253119416 | 1,076522411 | 4,993247118 |
| 7,098298088 | -5,670453601 | -1,543828588 | -3,06619666 | 3,794428375 | 4,394521258 | -1,727145503 | 3,877419311 | -7,704743308 | 0,547700628 |
| -3,89621691 | 0,630943228 | -2,679205332 | 3,707572422 | -1,377433753 | -4,187741492 | -0,467890561 | 4,562412374 | 2,480900429 | 1,226659595 |

Table 6. hasil dari invers dikali dengan X transpose

inv(XT\*X)\*XT\*y=W

|  |
| --- |
| 138,2841658 |
| -788,2612501 |
| -352,8367382 |
| 1266,829377 |
| 63,95580899 |
| 128,8460781 |

Table 7. dikali dengan Y akan mendapatkan W (bobot)

f(x)=w0+w1X1+w2X2+w3X3+w4X4+w5X5 Menghitung tebakan prediksi.

|  |
| --- |
| 164,248003 |
| 87,53764002 |
| 103,2386594 |
| 208,7776483 |
| 105,6047747 |
| 165,416206 |
| 90,30990418 |
| 83,88197329 |
| 161,3214601 |
| 255,663731 |

Table 8. hasil prediksi

Menghitung akurasi menggunakan R2

SStot = (Yi - (rata-rata Y))^2

|  |
| --- |
| 70,56 |
| 4569,76 |
| 2,56 |
| 4019,56 |
| 57,76 |
| 2079,36 |
| 21,16 |
| 6336,16 |
| 1062,76 |
| 28022,76 |

Table 9. Y dikurangi Y rata-rata dan dikuadratkan 2

Menghasilkan SStot = 46242,4

SSres = (Yi - f(X))^2

|  |
| --- |
| 175,5095841 |
| 157,1924173 |
| 1425,918845 |
| 7,715330335 |
| 864,0792718 |
| 4680,77724 |
| 2274,345239 |
| 436,0568086 |
| 2633,892269 |
| 2952,430131 |

Table 10. menghitung SSres

Menghasilkan SSres = 15607,91714

Kemudian R2 = 1-SSres/SStot = 0,662476058

Diperoleh Akurasi 0,662476058

1. **Pembagian Tugas**
2. Miftahul Choir 160411100055 (Program Linear Regression Manual Direct Equation, Hitung akurasi manual, generalisasi / testing)
3. Dennis Thandy Nur Cahyono 160411100079 (Laporan, dataset)
4. Nurrachmad Widyanto 160411100179 (Linear Regression Scikit)
5. Muhammad Yusqi Alfan Thoriq 160411100187(Program manual Linear Regression incremental learning stochastic)