8]:		scribe()						6	AV7 AS6 AS6	Z Z	12.	
9]:	count mean std min 25% 50% 75% max	201 201 201 201 201 201	67.0 1067. 4.0 3. 0.0 1 4.0 1. 4.0 2. 4.0 3. 4.0 4. 4.0 8.		CYLINDERS FULL 067.000000 5.794752 1.797447 3.000000 4.000000 6.000000 8.000000 12.000000	ELCONSU	MPTION_CITY 1067.000000 13.296533 4.101253 4.600000 10.250000 12.600000 15.550000 30.200000	) 2 3 3 ) )	1067.0000 9.4740 2.794 4.9000 7.5000 8.8000 10.8500	500 502 510 500 500 500 500	1067.000000 11.580881 3.485595 4.700000 9.000000 10.900000 13.350000 25.800000	
9]:	cdf = cdf.h  ENG  1 2 3 4	df [ ['EN ead (10)   2.0   2.4   1.5   3.5   3.5	IGINESIZE	RS FUELC 4 4 4 6 6	PONSUMPTION_C	8.5 9.6 5.9 11.1 10.6	196 22' 136 255 244		SIONS']]			
0]:	viz =	cdf[['C			ures by plott. NESIZE','CO2E				I_COMB']]			
	viz.h plt.s			200 100 0 FUE	ENGINESIZE ELCONSUMPTION							
1]:	#ploting these features against the Emission, to see their linear relationship  plt.scatter(cdf.FUELCONSUMPTION_COMB, cdf.CO2EMISSIONS, color='blue')  plt.xlabel("FUELCONSUMPTION_COMB")  plt.ylabel("Emission")  plt.show()											
	500 - 450 - 400 - 350 - 250 - 200 - 150 - 100 - 1		10 FUEL	15 .CONSUMPTIO	ZO ON COMB	25						
2]:	plt.x plt.y plt.s 500 - 450 - 400 - 350 -	label("E label("E		NESIZE, o	edf.CO2EMISSI	ONS, co	lor='blue'	)				
3]:	200 - 150 - 100 - plt.s plt.x	label("C label("E	df.CYLINCYLINDERS	S")	é 7 ze df.CO2EMISSIO	8 NS, col	or='blue')					
	450 - 400 - 350 - 150 - 100 -	4	6	8 CYLINDER		12						
4]: 5]:	msk = train test : #For #Trai plt.s plt.x	<pre>np.rand = cdf[m = cdf[~m sinple l n data c catter(t label("E label("E</pre>	dom.rand nsk] nsk] linear re	tion GINESIZE, ize")	< 0.8				ar model with	coefficients	B = (B1,,	
6]:	400 - 350 - 9 300 - 250 - 200 - 150 - 100 -	i ż	3 e data us	4 5 Engine si:	ze	8						
	from regr: train train regr. # The print print Coeffi	sklearn = linear _x = npy = np. fit(trai _coeffic ('Coeff ('Inter cients:	import in a model. I as anyar in anyar in as anyar in	linear_mo LinearRec ray(trair ray(trair ain_y) : ', regr , regr.int	odel gression() n[['ENGINESIZ] n[['CO2EMISSIG							
	plt.s plt.p plt.x plt.y	catter(t lot(trai label( <mark>"E</mark> label("E	rain.ENG	gr.coef_  ize") ")	the data train.CO2EM				-r')			
8]:	250 - 200 - 150 -	1 2	metrics	4 5 Engine siz	ze	8						
	test_ test_ print print print Mean a Residu R2-sco	y = np.a y_ = reg ("Mean a ("Residu ("R2-sco bsolute al sum o re: 0.75	ubsolute al sum of error:	error: % of square f" % r2_s 22.99 es (MSE)	%.2f" % np.mea es (MSE): %.2 score(test_y	S']]) an(np.ab f" % np. , test_y	mean((test	_y_ <b>-</b> te	est_y) ** 2))			
	train test_ regr. predi print print print Mean A Residu R2-sco	x = tra  x = test  = linear  fit(trai  ctions =  ("Mean A  ("R2-sco  bsolute  al sum o  re: 0.75	in[["FUELO c_model.I n_x, tra regr.pa absolute aal sum o pre: %.21	CONSUMPTI LinearRegain_y) redict(te Error: % of square f" % r2_s	PTION_COMB"]]  ION_COMB"]]  gression()  est_x)  %.2f" % np.med es (MSE): %.2;  score(test_y)  : 965.95	an(np.ab f" % np.	solute(pre mean((test	_ dictions	s - test_y)))			
2]: [ 4]: [ 4]:	<pre>impor impor impor impor %matp</pre> df = df.he	t matplo t pandas t pylab t numpy lotlib i  pd.read_ ad(10)	as pd as pl as np nline csv('htt		olt -courses-data						loperSkillsNet	
	0 1 2 3 4 5 6	2014 2014 2014 2014 2014 2014	ACURA	ILX ILX HYBRID MDX 4WD RDX AWD RLX TL	COMPACT COMPACT COMPACT SUV - SMALL SUV - SMALL MID-SIZE MID-SIZE	3	2.0 2.4 1.5 3.5 3.5 3.5	4 4 4 6 6 6 6	AS5 M6 AV7 AS6 AS6 AS6 AS6	Z Z Z Z Z Z Z	9. 11. 6. 12. 12. 11.	
5]:	cdf.h	2014 2014 df[['ENead(10)			MID-SIZE  MID-SIZE  COMPACT  NDERS', 'FUELCO  CONSUMPTION_C	2 ONSUMPTI			FUELCONSUM		12 13 10 FION_COMB', 'CO	
	1 2 3 4 5 6 7	2.4 1.5 3.5 3.5 3.5 3.5 3.7 3.7		4 4 6 6 6 6 6 6 6		11.2 6.0 12.7 12.1 11.9 11.8 12.8 13.4		7.7 5.8 9.1 8.7 7.7 8.1 9.0		9.6 5.9 11.1 10.6 10.0 10.1 11.1 11.6	221 136 255 244 230 232 255 267	
6]:	plt.x plt.y plt.s 500 - 450 - 400 -	label("E label("E	edf.ENGIN Engine si Emission'	ize")	edf.CO2EMISSI	10.6  ONS, co	lor='blue'	7.5		9.2	212	
7]:	msk =			4 5 Engine siz	ze	8						
8]:	plt.s plt.x plt.y plt.s	<pre>cdf[~m catter(t label("E label("E</pre>	nsk]	ize")	train.CO2EM	ISSIONS,	color='b	lue')				
9]:					ze			_			s of cars to p	
Ø]:	<pre>from   regr:   x = n;   y = n;   regr.   print</pre>	sklearn = linear p.asanya p.asanya fit (x,  ('Coeff cients:	import in a model. I array (transport in a model) in a model in a	linear_mc LinearRec ain[['ENC ain[['CO2 : ', regi	odel gression() GINESIZE','CY ZEMISSIONS']] r.coef_) 7.88799292	LINDERS' ) 9.323560	,'FUELCONS	UMPTION_	_COMB']])		(OLS) method.	
1]:	x = ny y = ny print # Exp print Residu Varian # Mul	p.asanya p.asanya ("Residu % np.me  lained v ('Varian  al sum o ce score  tiple li	array(tes array(tes al sum ( ean((y_ha variance ace score of square e: 0.88	st[['ENG] st[['CO2F of square at - y) * score: 1 e: %.2f' es: 527.9	1 is perfect not seek to seek the seek to seek the seek to seek the seek th	INDERS',  predicti (x, y))	'FUELCONSU	MPTION_C	COMB']])	Y instead of 1	FUELCONSUMPTIO	
	<pre>x = n; y = n; regr. print y_= r x = n; y = n; print print Coeffi Residu</pre>	p.asanya p.asanya fit (x,	array(tray)  icients: dict(test array(test a	ain[['ENG ain[['CO2 : ', regi t[['ENGIN st[['ENGIN st[['CO2F of square e: %.2f'	NESIZE','CYLII INESIZE','CYLI EMISSIONS']]) es: %.2f"% np % regr.score	NDERS',' INDERS', .mean((y(x, y))	FUELCONSUM 'FUELCONSU '- y) **	PTION_CI MPTION_C 2))	TY','FUELCONS	SUMPTION_HWY'	1])	
2]: [	df.he	2014 2014	MAKE ACURA ACURA	MODEL VILX ILX HYBRID		2	<b>ZE CYLINDE</b> 2.0 2.4 1.5	<b>RS TRAN</b> 4 4	AS5 M6 AV7	Z Z Z	<b>NSUMPTION_CIT</b> 9. 11. 6.	
	<ul><li>3</li><li>4</li><li>5</li><li>6</li><li>7</li><li>8</li><li>9</li></ul>	2014 2014 2014	ACURA ACURA ACURA ACURA	MDX 4WD RDX AWD RLX TL TL AWD TL AWD	SUV - SMALL  SUV - SMALL  MID-SIZE  MID-SIZE  MID-SIZE  MID-SIZE  COMPACT	3 3 3	3.5 3.5 3.5 3.7 3.7	6 6 6 6 6	AS6 AS6 AS6 AS6 AS6 AS6 AS6	Z Z Z Z Z Z Z Z	12. 12. 11. 11. 12. 13.	
	•	2014	NOONN	d second essing <b>in</b> linear_mo	degree plynom	mial for			7100	_	10.	
3]:	<pre>from from train train test_ test_</pre>	2014  2014  2014  g ENGINE sklearn x = np. y = np. x = np.a y = np.a	preproce import i asanyari asanyari asanyaria	ray(trair ay(test[  ay(test[  tures(deg		E']]) ONS']])  ]]) S']])	res					
	<pre>from from train train test_ test_  poly train train  array(</pre>	2014  2014  2014  2014  g ENGINE sklearn sklearn x = np. y = np. a y = np. a y = np. in inear y = cl ('Coeff	preproce import asanyarra	ray(train ay(test[  ay(test[  tures(deg fit_trans 4. ], 5.76], 2.25], 9. ], 10.24], 10.24]]; inearRegn rain_x_po : ', clf.	n[['CO2EMISSION: ['ENGINESIZE' ['CO2EMISSION: gree=2) sform(train_x) ) ression() oly, train_y) .coef_)	E']]) ONS']])  ]]) S']])	res					
3]: 4]:	from from train train test_ test_ poly train tra	2014  2014  2014  g ENGINE sklearn sklearn x = np. y = np. x = np.a y = np.a y = np.a  Polync x_poly x_poly [[1. , [1. ,	preproce import asanyarra	ray(train ay(test[  ay(test[  ay(test[  tures(deg fit_trans  4.  ], 5.76], 2.25],  9.  ], 10.24], 10.24]]; inearReg rain_x_po : ', clf. inte  3302]  GINESIZE, 10.0, 0.1 0]+ clf. 0 ize") ")	ression() coef_) ercept_) train.CO2EMISSIO	E']]) ONS']])  31) S']])  1.540043 ISSIONS,	369]] color='b	lue')	er(XX, 2)			
3]: 4]:	from from train train test_ test_ poly train tra	2014  2014  2014  g ENGINE sklearn sklearn x = np. y = np. x = np.a y = np.a y = np.a  Polync x_poly x_poly [[1. , [1. ,	preproce import asanyarra	ray(train ay(test[  ay(test[  ay(test[  tures(deg fit_trans  4.  ], 5.76], 2.25],  9.  ], 10.24], 10.24]]; inearReg rain_x_po : ', clf. inte  3302]  GINESIZE, 10.0, 0.1 0]+ clf. 0 ize") ")	ression() oly, train_y) coef_) ercept_) train.CO2EMI coef_[0][1]*X	E']]) ONS']])  31) S']])  1.540043 ISSIONS,	369]] color='b	lue')	er(XX, 2)			
3]: 5]:	from from train train test_ poly train train array(  clf = train print coeffi Interc  plt.s XX = yy = plt.y plt.y Text(0  500 f  450 f  400 f  450 f  460 f  470 f  480 f	2014  2014  2014  g ENGINE sklearn sklearn x = np. y = np. x = np.a y = np.a y = np.a y = np.a y = np.a  [ 1. ,	preproce import asanyarra asanyarra sanyarra san	ray(train ay(test[  ay(tes	cession() coef_) creept_) coef_() creept_) coef_() creept_) coef_() creept_) coef_() creept_) coef_() coef_()() coef_()() coef_()() coef_()() coef_()() coef_()()() coef_()()()() coef_()()()()() coef_()()()()()() coef_()()()()()()()()()()()()()()()()()()()	E']]) ONS']])  1.540043  ISSIONS, X+ clf.c	color='bcoef_[0][2]	lue') *np.powe	eest_y)))			
3]: 5]:	from train train test_ poly train train array(  clf = train print print Coeffi Interc  plt.s XX = yy = plt.y plt.y  Text(0  500 f 450 f 400 f 450 f 45	2014  2014  2014  2014  2014  2014  g ENGINE  sklearn  sklearn  x = np.  y = np.  x = np.  y = np.  x = poly  x poly  x poly  [[1. ,  [	metrics  met	ray(train ay(test[] ay(tes	f [ 'CO2EMISSION   Co2EMISSION   C2EMISSION   C2EMISSION	E']]) ONS']])  1) S']])  1.540043  ISSIONS,  X+ clf.c	color='b cof_[0][2]  color='b color='b	lue')  *np.powe	eest_y))) est_y) ** 2))	<ul><li>+ ρ1 °</li></ul>		
3]: 4]: 5]:	from train train test_ test_ polyitrain train array(  clf = train print coeffice Interco  sty = plt.y plt.x plt.y  Text(0  450  400  450  100  from test_ print print test_ print print test_ print print coeffice sty plt.y  test_ print print coeffice sty	2014  2014  2014  2014  2014  2014  2014  2014  2014  2014  2016  Sklearn.  Sklearn.  Sklearn.  X = np.  Y = np.  X = np.  Y = np.  Y = np.  Y = np.  ("Polyno  X = poly  X = poly  ("Inter  Cients:  ept: [1]  Catter(t  np.arang  ('Coeff  ('Inter  cients:  ept: [1]  catter(t  np.arang  ('Coeff  ('Inter  cients:  ept: [1]  catter(t  np.arang  ('Residu  ('	metrics  metrics  metrics  metrics  metrics  metrics  model.Lifetitititicients  cept: ',  1.8.3.2,  3.2,  model.Lifetititicients  cept: ',  cept:	ray(train ay(test[] ay(tes	f[['CO2EMISSION: Incomplete (CO2EMISSION: Inco	E']]) ONS']])  S']])  1.540043  ISSIONS,  X+ clf.c  an(np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f" % np. test_y_  an(np.ab f" % np. test_y3_	color='b coef_[0][2] cof_[0][2] coef_[0][2] foef_[0][2] foef_[0][2] coef_[0][2] foef_[0][2] foef_[0][2] coef_[0][2] foef_[0][2] foef_[0][2	lue') *np.powe  t_y te  y te  ee (cubi	<pre>power(XX, 2)  test_y)))  test_y) ** 2))</pre>		[0][3]*np.powe	
3]: 4]: 5]:	from train train test_ polyitrain train array(  clf = train print Coeffi Interc  plt.s XX = yy = plt.x y Text(0  400  400  400  400  from test_ test_ print print Mean a Residu R2-sco  # Pol  poly3 train train  print print Coeffi Interc  Accord  # Pol  coeffi	2014  2014  2014  2014  2014  2014  2014  2014  2014  2016  Sklearn.  Sklearn.  Sklearn.  Y = np.  Y =	metrics  metrics  metrics  metrics  metrics  metrics  model.Lifetitititicients  cept: ',  1.8.3.2,  3.2,  model.Lifetititicients  cept: ',  cept:	ray(train ay(test[] ay(tes	['CO2EMISSIGNIC CO2EMISSION:  ['ENGINESIZE' ['CO2EMISSION:  gree=2) sform(train_x  ]  pression() coef_) creept_)  50.08730154  train.CO2EMI  coef_[0][1]*XX  gree=3) ansform(train, gression() coef_[0][1]*XX  gree=3) ansform(train, gression() coef_[0][1];  coef_(1) coef_(2) coef_(2) coef_(3) coef_(4) coef_(5) coef_(6) coef_(6) coef_(7) coef_(7) coef_(7) coef_(8) coef_(8) coef_(1) coef_(8) coef_(1) coef_	E']]) ONS']])  S']])  1.540043  ISSIONS,  X+ clf.c  an(np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f" % np. test_y_  an(np.ab f" % np. test_y3_	color='b coef_[0][2] cof_[0][2] coef_[0][2] foef_[0][2] foef_[0][2] coef_[0][2] foef_[0][2] foef_[0][2] coef_[0][2] foef_[0][2] foef_[0][2	lue') *np.powe  t_y te  y te  ee (cubi	<pre>power(XX, 2)  test_y)))  test_y) ** 2))</pre>		[0][3]*np.powe	
3]: 5]:	from train test_ polyitrain train array(  clf = train print Coeffi Interc  plt.s XX = plt.y Text(0  450  400  400  from test_ print print Mean a Residu R2-sco  # Pol  poly3 train train  print print print Coeffi Interc  Mean a Residu R2-sco  450  400  400  400  400  400  400  40	2014  2014  2014  2014  2014  2014  2014  2014  2014  2014  2018  sklearn.  sklearn.  x = np.  y = np.  x = np.  y = np.  ay = np.  ay = np.  y = np.  ("Polyno  x = polyno  x = polyno  ("Coeff ("Inter  cients: ept: [1]  catter(ton clarange ("Inter  cients: ept: [1]  catter(ton clarange ("Inter  cients: ept: [1]  catter(ton clarange ("Residue ("R2-sco coents: eptolyno x = polyno x =	preproce import asanyarra	ray(train ay(test[] ay(tes	cession() cossion() cossio	E']]) ONS']]) S']]) S']]) S']])  1.540043  ISSIONS,  X+ clf.c  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f" % np. test_y_  with deg 3.176604	color='b coef_[0][2]  color='b f3.coef_[0  solute(test ))  f3.coef_[0  solute(test ))  solute(test ))  solute(test ))  solute(test ))  solute(test ))	lue') *np.powe  *y te  ee (cubi  t_y te  8774]]	<pre>cest_y))) est_y) ** 2))  test_y)); est_y) ** 2))</pre>	a from 1960 to	D 2014	
3]: 4]: 5]:	from train test_ polyitain train array(  clf = train print coeffic Interco  from test_ polyitain train array(  clf = train print coeffic Interco  from test_ polyitain train array(  from test_ polyitain train array(  from test_ polyitain train tra	2014  2014  2014  2014  g ENGINE sklearn xklearn xy = np. xx = np.	preproce import asanyari asany	ray(train ay(test[] ay(tes	cession() cossion() cossio	E']]) ONS']]) S']]) S']]) S']])  1.540043  ISSIONS,  X+ clf.c  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f" % np. test_y_  3.176604  to verif	solute(tesmean((test)))  color='b coef_[0][2]  f3.coef_[0  solute(tesmean((test)))  19 -0.3894	lue') *np.powe  t_y te  ee (cubi  t_y te  8774]]	<pre>power(XX, 2)  test_y) ** 2))  test_y) ** 2))  test_y) ** 2))</pre>	a from 1960 to	D 2014	
3]: 5]: 9]:	from train test_ polyintrain test_ polyintrain train array(  clf = train print Coeffic Interc  plt.s  XX = plt.y  Text(0  450 -	2014  2014  2014  2014  g ENGINE sklearn xklearn xy = np. xx = np.	preproce import : asanyari asa	ray(train ay(test[] ay(tes	collinear relations ("CO2EMISSION ("ENGINESIZE" ("CO2EMISSION ("CO2EMISSION ("CO2EMISSION ("CO2EMISTON ("CO2E	E']]) ONS']]) S']]) S']]) S']])  1.540043  ISSIONS,  X+ clf.c  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f" % np. test_y_  3.176604  to verif	solute(tesmean((test)))  color='b coef_[0][2]  f3.coef_[0  solute(tesmean((test)))  19 -0.3894	lue') *np.powe  t_y te  ee (cubi  t_y te  8774]]	<pre>power(XX, 2)  test_y) ** 2))  test_y) ** 2))  test_y) ** 2))</pre>	a from 1960 to	D 2014	
3]: 5]: 9]:	from train train train train train train train array(  clf = print print Coeffic Interc  sxy = pplt.x yplt.x yplt.	2014  2014  2014  2014  g ENGINE sklearn. sklearn. x = np.a x = np.a y = np	metrics  model.Lif.fit(total)  model.Lif.fit	ray(train ay(test[] ay(tes	cession() cression() c	E']]) ONS']])  I)  1.540043  ISSIONS,  X+ clf.c  an(np.ab f"% np. test_y3  3.176604  inear re onship h  to verif e)  to verif e)	color='b  color='b  f3.coef_[0]  solute(test ))  color='b  f3.coef_[0]  solute(test ))  solute	lue') *np.powe  ee (cubi  lue')  [2]*np.  *sing Chi depende  ges in t	cest_y))) est_y) ** 2))  test_y))) est_y) ** 2))  test_y) ** 2))  the graph  the graph	a from 1960 to	o 2014 riable with a	
3]: 5]: 8]:	from from train train test  polyination train tr	2014  2014  2014  2014  2014  g ENGINE sklearn. x = np. x = np. x = np. x = np. y = np. x = np. y = np	preprocedimport in a sanyarra as anyarra anyarra as anyarra anyarra anyarra anyarra as anyarra as anyarra a	ay(test[] ay(tes	ression() play, train_y ression() play, train_y recept_) shorm(train_y recept_) score(test_x) poly) secore(test_y, secore(test	E']]) ONS']]) I) S']])  1.540043  ISSIONS, X+ clf.c  an(np.ab f" % np. test_y_  with deg  x) _y)  ISSIONS, *XX + cl  an(np.ab f" % np. test_y3_ 3.176604  to verif e)  to verif e)	color='b coef_[0][2]  solute(tes mean((test )))  color='b f3.coef_[0  solute(tes mean((test )))  f2.coef_[0]  solute(tes mean())  f3.coef_[0]  solute(tes mean())  f3.coef_[0]  solute(tes mean)  fy the chan  solute(tes mean)  fy the chan  solute(tes mean)  solute(t	lue') *np.powe  ee (cubi  t_y3 y3 t  8774]]  sing Chi depende  ges in t	power(XX, 2)  test_y)))  test_y) ** 2))  test_y) ** 2))  test_y) ** 2))  test_y) ** 2);  test_y) ** 2);	a from 1960 to	o 2014 riable with a	
3]:         6]:         9]:	from train t	2014  2014  2014  2014  2014  2014  2014  2014  2016  Sklearn.  x = np.  y = np.  y = np.  y = np.  x = np.  y = np.  y = np.  y = np.  x = np.  y = np.  x = np.  y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y = np. y	preproce import asanyaria	reay(train ay(test[] ay(te	delegation of the control of the con	E']]) ONS']]) ONS']]) S']])  1.540043  ISSIONS,  X+ clf.c  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  3.176604  Lo verif e)  to verif e)  4  Lo verif e)  4  Lo verif e)	color='b coef_[0][2]  solute(tes mean((test )))  color='b f3.coef_[0  solute(tes mean((test )))  f2.coef_[0]  solute(tes mean())  f3.coef_[0]  solute(tes mean())  f3.coef_[0]  solute(tes mean)  fy the chan  solute(tes mean)  fy the chan  solute(tes mean)  solute(t	lue') *np.powe  ee (cubi  t_y3 y3 t  8774]]  sing Chi depende  ges in t	power(XX, 2)  test_y)))  test_y) ** 2))  test_y) ** 2))  test_y) ** 2))  test_y) ** 2);  test_y) ** 2);	a from 1960 to	o 2014 riable with a	
3]: 5]: 8]:	from from train test  planin train  aray(  claim  confinit the st  print  confinit the st  print the st  p	2014  2014  2014  2014  2014  2014  2014  2014  2018  2018  2018  2018  2018  2019	preprocedimport in a sanyarra a s	restication was a server of square for squar	ression() cression() c	E']]) ONS']]) S']]) S']])  1.540043  ISSIONS,  X+ clf.c  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  Ito verif  to verif to verif  to verif  xto verif  xto verif  xto verif  xto verif	solute(tesmean((test)))  color='b f3.coef_[0  solute(tesmean((test)))  f3.toef_[0]  solute(tesmean(test))  f3.toef_[10  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  fy the chan	lue')  t_y te  y te  ge (cubi  lue')  [2]*np.  sing Chi depende  ges in te  ges in te	cest_y))) est_y) ** 2))  test_y)) est_y) ** 2))  ina's GDP data ent and the in the graph  the graph	a from 1960 to	o 2014 riable with a	
3]:         6]:         9]:	from from train test  polyin train t	2014  2014  2014  2014  2014  2014  2014  2014  2018  2018  2018  2018  2018  2019	preprocedimport in a sanyarra a s	ay(train ay(test[] ay(test	ression() cression() c	E']]) ONS']]) S']]) S']])  1.540043  ISSIONS,  X+ clf.c  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an(np.ab f"% np. test_y_  with deg  _x) _y)  Ito verif  to verif to verif  to verif  xto verif  xto verif  xto verif  xto verif	solute(tesmean((test)))  color='b f3.coef_[0  solute(tesmean((test)))  f3.toef_[0]  solute(tesmean(test))  f3.toef_[10  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  f3.toef_[10]  fy the chan	lue')  t_y te  y te  ge (cubi  lue')  [2]*np.  sing Chi depende  ges in te  ges in te	cest_y))) est_y) ** 2))  test_y)) est_y) ** 2))  ina's GDP data ent and the in the graph  the graph	a from 1960 to	o 2014 riable with a	
3]:         6]:         9]:	from from the first test of th	2014 2014 2014 2014 2014 2014 2014 2014	preprocedimport in a sanyaria asanyaria asanya	rey(train  ay(test [ ay(te	design ()  cession () coly, train_y, color ('CO2EMISSION: design (	E']]) ONS'])) S']))  S']))  1.540043  ISSIONS,  X+ clf.c  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  xi  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  xi  an (np.ab f" % np.	solute(tesmen) (test) (	lue') *np.powe  t_y te  ee (cubi  lue')  [2]*np.  t_y3 te  sing Chi depende  ges in te  ges in te  ges in te	cest_y))) est_y) ** 2))  test_y)) est_y) ** 2))  test_y)) est_y) ** 2))  the graph  the graph  the graph  the graph	a from 1960 to	es x and the d	
3]: 6]: 9]:	from training ()  Consising are to the print	2014  2014  2014  2014  2014  2014  2014  2014  2014  2014  2016  Sklearn  x = np.  x = np. x = np.  x = np. x = np. x = np. x = np. x = np. x = np. x = np. x = np. x = np. x = np. x	preprocedimport in asanyaria asanyar	restion  agraph  finance  agraph  finance  agraph  finance  fit_trans  4. ],  5. 76],  2. 76],  9. ],  10. 24],  10. 30  agraph  finance  financ	description of the continuous	E']]) ONS'])) S']))  S']))  1.540043  ISSIONS,  X+ clf.c  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  xi  an (np.ab f" % np. test_y_  with deg  _x) _y)  ISSIONS,  *XX + cl  an (np.ab f" % np. test_y_  xi  an (np.ab f" % np.	solute(tesmen) (test) (	lue') *np.powe  t_y te  ee (cubi  lue')  [2]*np.  t_y3 te  sing Chi depende  ges in te  ges in te  ges in te	cest_y))) est_y) ** 2))  test_y)) est_y) ** 2))  test_y)) est_y) ** 2))  the graph  the graph  the graph  the graph	a from 1960 to	o 2014 riable with a	
3]: 6]: 9]:	from minimum array ()  The state of the stat	2014  2014  2014  2014  2014  2014  2018  2018  2018  2018  2018  2019	preproces import asanyaria	atures (de de d	ression()	e']]) oNS']]) s'])) s']) s']	solute(tesmean((test)))  color='bf3.coef_[0]  solute(tesmean((test)))  free of thr  color='bf3.coef_[0]  solute(tesmean((test)))  ify the chan  fy the chan	lue') *np.powe  t_y te  y te  y te  ee (cubii  lue')  [2]*np.  sing Chi depende  ges in t  ges in t  ges in t  ges in t	power(XX, 2)  test_y);  test_y);  test_y);  test_y);  the graph	a from 1960 to dependent variable and so on.	o 2014 riable with a o the output y	
3]:   6]:   9]:	# In the standard of the stand	2014  2014  2014  2014  2014  2014  2014  2016  2016  2016  3018  3019	preproceding procedure and	## Company  ## Com	resident state of the state of	E'!]) ONS']) ONS']) ONS']) I) S'])  1.540043  ISSIONS, *X+ clf.c.  an (np.ab f" % np. test_y_  with deg _x) _y)  ISSIONS, *XX + cl  an (np.ab f" % np. test_y_  with deg _x) _y)  ISSIONS, *XX + cl  an (np.ab f" % np. test_y_  with deg _x)  Lo verif e)  10  S  inear re e: y=ax3 ke expon  to verif e)  14  44  A  or onship h  to verif e)  44  to verif e)  45  46  47  47  48  49  40  40  40  40  41  41  41  42  43  44  45  46  47  47  48  49  40  40  40  40  40  41  41  42  43  44  45  46  47  48  49  40  40  40  40  40  40  40  40  40	solute(tesmean((test)))  color='b f3.coef_[0  solute(tesmean((test)))  ree of thr  color='b f3.coef_[0  solute(test))  f3.coef_[10  solute(test))  fy the chan  representation under the solute of the chan  fy the chan	lue') **np.powe  t_Y te  y te  y te  fee (cubi  lue')  [2]*np.  sing Chi depende  ges in t	the graph  the graph  the graph  the graph  the graph  the graph	a from 1960 to adependent variable and so on.	o 2014  riable with a specific property of the output yellow the output yellow the second property of the output yellow	
3]:   6]:   9]:	from in the set of the state of	## 2014  ## 2014  ## 2014  ## 2014  ## 3	preproceding of the proceding of the pro	## Company  ## Com	ariable ('ENGINESIZE' ('ENGINESIZE' ('ENGINESIZE' ('COZEMISSION: ('ENGINESIZE' ('COZEMISSION: ('	E'!]) ONS']) ONS']) ONS']) I) S'])  1.540043  ISSIONS, *X+ clf.c.  an (np.ab f" % np. test_y_  with deg _x) _y)  ISSIONS, *XX + cl  an (np.ab f" % np. test_y_  with deg _x) _y)  ISSIONS, *XX + cl  an (np.ab f" % np. test_y_  with deg _x)  Lo verif e)  10  S  inear re e: y=ax3 ke expon  to verif e)  14  44  A  or onship h  to verif e)  44  to verif e)  45  46  47  47  48  49  40  40  40  40  41  41  41  42  43  44  45  46  47  47  48  49  40  40  40  40  40  41  41  42  43  44  45  46  47  48  49  40  40  40  40  40  40  40  40  40	solute(tesmean((test)))  color='b f3.coef_[0  solute(tesmean((test)))  ree of thr  color='b f3.coef_[0  solute(test))  f3.coef_[10  solute(test))  fy the chan  representation under the solute of the chan  fy the chan	lue') **np.powe  t_Y te  y te  y te  fee (cubi  lue')  [2]*np.  sing Chi depende  ges in t	the graph  the graph  the graph  the graph  the graph  the graph	a from 1960 to adependent variable and so on.	o 2014  riable with a specific property of the output yellow the output yellow the second property of the output yellow	
3]:   6]:   9]:	from in the state of the state	2014 2014 2014 2014 2014 2014 2014 2014	regress:  compared to the state of the state	dependent variable services (train x process (MSE))  Tession  Anguerate services (train x process (MSE))  Tession using services (Tession using services (MSE))  Tession using services (Tession using services (MSE))  Tession using services (MSE)  Tessio	ression()  respt  respt  ression()  respt  r	E']]) ONS']) ONS']) ONS']) S']) S'])  1.  1.  1.  1.  1.  1.  1.  1.  1.  1	solute (tesmean (test))  color='b f3.coef_[0 f3.coef_[0 f3.coef_[0 f3.coef_[0 f3.coef_n fy the chan fy	d by Y=a  t_y te  t_y te  fee (cubic  lue')  y te  ee (cubic  ges in t  sing Chi depende  ges in t  ges in t	the graph	a from 1960 to adependent variable and so on.	o 2014  riable with a   o the output y   ries.	
3]:         9]:         3]:	from in the state of the state	## A POLYMENT OF THE PROPERTY	preproceding and	dependent variable of square (dranke)  stream (asyltrain asyltest [asyltrain asyltrain	ression() in(['COZEMISSION in(['COZEMISSION in("ENGINESIZE" 'COZEMISSION interest incozemistry i	E']]) ONS']) ONS']) S'])  1.540043  ISSIONS, X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x) _y)  ISSIONS, X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x) _y)  ISSIONS, X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x) _y)  ISSIONS,  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x) _y)  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x) _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y  *X+ clf.c  an(np.ab f" % np. test_y_ with deg  _x _y _y _y  *X+ cl	solute (tesmean (test))  color='b f3.coef_[0 f3.coef_[0 f3.coef_[0 f3.coef_[0 f3.coef_n fy the chan fy	d by Y=a  t_y te  t_y te  fee (cubic  lue')  y te  ee (cubic  ges in t  sing Chi depende  ges in t  ges in t	the graph	a from 1960 to adependent variable and so on.	o 2014  riable with a   o the output y   ries.	
3]:         6]:         7]:         4]:	# Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y	2014 2014 2014 2014 2014 2014 2014 2014	preproductions and	## A Page of the control of the cont	responding to the control of the con	e)	solute(tesment)  solute(tesment)  solute(tesment)  color='b  f3.coef_[0]  solute(tesment)  fy the chan  solute(tesment)  fy the chan  f	lue') *np.powe  t_y te  y te  ce (cubi  lue')  [2]*np.  sing Chi depende  ges in t	test_y())  ist_y()**2))  test_y()**2))  test_y()**2)  test_y()**2)	a from 1960 to and so on.  The input x to of the x value.	of the output years.	
3]:         6]:         9]:	form in the state of the state	### ### ### ### ### ### ### ### ### ##	regressions and state of the st	## A Page of the control of the cont	responding to the control of the con	e)	solute(tesment)  solute(tesment)  solute(tesment)  color='b  f3.coef_[0]  solute(tesment)  fy the chan  solute(tesment)  fy the chan  f	lue') *np.powe  t_y te  y te  ce (cubi  lue')  [2]*np.  sing Chi depende  ges in t	test_y())  ist_y()**2))  test_y()**2))  test_y()**2)  test_y()**2)	a from 1960 to and so on.  The input x to of the x value.	o 2014 riable with a o the output y	
3]:         6]:         7]:         4]:	from minimate for the first state of the first stat	2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2016   3016	regress:  asayar: asay	## A Property of the Control of the	ression()	E']]) ONS) ONS) ONS) ISSIONS, X+ clf.c  ISSIONS, X+ clf.c  IN (np.ab f"% np. test_y_  with deg  x) y)  ISSIONS, A  * A  * Clf.c  * A  * A  * Clf.c  * A  * A  * A  * A  * A  * A  * A  *	solute(test octop='b oct_[0][2]  solute(test ) )  ree of thr  color='b f3.coef_[0]  solute(test ) )  ree of thr  color='b f3.coef_[0]  solute(test ) )  fy the chan  fy the chan  y the chan	lue') *np.powe  t_y te  y te  ce (cubi  lue')  [2]*np.  sing Chi depende  ges in t	test_y())  ist_y()**2))  test_y()**2))  test_y()**2)  test_y()**2)	a from 1960 to and so on.  The input x to of the x value.	es x and the d othe output y othe output y othes.	
8]:         6]:         5]:	from minimate for the first state of the first stat	2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2014   2016   2016   3016	regress:  asayar: asay	dependent variable transform and son end and and and and and and and and and a	ression()	E']]) ONS) ONS) ONS) ISSIONS, X+ clf.c  ISSIONS, X+ clf.c  IN (np.ab f"% np. test_y_  with deg  x) y)  ISSIONS, A  * A  * Clf.c  * A  * A  * Clf.c  * A  * A  * A  * A  * A  * A  * A  *	solute(test octop='b oct_[0][2]  solute(test ) )  ree of thr  color='b f3.coef_[0]  solute(test ) )  ree of thr  color='b f3.coef_[0]  solute(test ) )  fy the chan  fy the chan  y the chan	lue') *np.powe  t_y te  y te  ce (cubi  lue')  [2]*np.  sing Chi depende  ges in t	test_y())  ist_y()**2))  test_y()**2))  test_y()**2)  test_y()**2)	a from 1960 to and so on.  The input x to of the x value.	es x and the d othe output y othe output y othes.	

#plot initial prediction against datapoints plt.plot(x_data, Y_pred*1500000000000) plt.plot(x_data, Y_data, 'ro')  [ <matplotlib.lines.line2d 0x7ffa6c781340="" at="">]  14  12  10  08  06  04  02  00  1970  1980  1990  2000  2010  2010  2011  # normalizing the data x_data = x_data/max(x_data) ydata = y_data/max(y_data)  # from scipy.optimize import curve_fit popt, pcov = curve_fit(sigmoid, xdata, ydata) #print the final parameters print(" beta_1 = %f, beta_2 = %f" % (popt[0], popt[1]))  beta_1 = 690.453017, beta_2 = 0.997207</matplotlib.lines.line2d>	[28]: [29]:	<pre>beta_2 = 1990.0  #logistic function Y_pred = sigmoid(x_data, beta_1 , beta_2)</pre>
### State of the content of the cont	[29]:	<pre>#plot initial prediction against datapoints plt.plot(x_data, Y_pred*1500000000000.) plt.plot(x_data, y_data, 'ro')  [<matplotlib.lines.line2d 0x7ffa6c781340="" at="">]  14 12 10 0.8</matplotlib.lines.line2d></pre>
<pre>compt, note = current fificiquates spring to be fine) parameters refine("beta   = \$f, beta 2 = \$f" % (ront(0), pont(1))  Deta_1 = 690.19507, Deta_2 = 0.997207  lette_1 = 690.19507, Deta_2 = 0.997207  lette_1 = 690.19507, Deta_2 = 0.997207  x = y-non(s) x = y-non(s) lette_1 = 690.19507, Deta_2 = 0.997207  x = y-non(s) lette_1 = 600.19507, Deta_2 = 0.997207  x = y-non(s) lette_1 = 0.907207, Deta_2 = 0.997207  x = y-non(s) lette_1 = y-non(s) le</pre>	[30]:	# normalizing the data xdata = x_data/max(x_data) ydata = y_data/max(y_data)
y = signoid(x, *popt) plt.plot(x,yt, linewidth=3.0, label='fit') plt.plot(x,yt, linewidth=3.0, label='fit') plt.ylabel('GDY') plt.xlabel('GDY') plt.xlabel('SDY') plt.xlabel('	31]:	<pre>popt, pcov = curve_fit(sigmoid, xdata, ydata) #print the final parameters print(" beta_1 = %f, beta_2 = %f" % (popt[0], popt[1]))  beta_1 = 690.453017, beta_2 = 0.997207  #Resulting regression  x = np.linspace(1960, 2015, 55) x = x/max(x) plt.figure(figsize=(8,5)) y = sigmoid(x, *popt)</pre>
# Calculating the accuracy of the model  # split data into train/test msk = np.random.rand(len(df)) < 0.8 train_x = xdata[msk] test_x = xdata[msk] test_y = ydata[msk] test_y = ydata[msk]  # build the model using train set popt, prov = curve_fit(sigmoid, train_x, train_y)  # predict using test set y_hat = sigmoid(test_x, *popt)  # evaluation print("Mean absolute error: 8.2f" % np.mean(np.absolute(y_hat - test_y))) print("Residual sum of squares (MSE): 8.2f" % np.mean((y_hat - test_y) ** 2))  from sklearn.metrics import r2_score print("82-score: 8.2f" % r2_score(test_y, y_hat))  Mean absolute error: 0.04 Residual sum of squares (MSE): 0.00 R2-score: 0.96		<pre>y = sigmoid(x, *popt) plt.plot(xdata, ydata, 'ro', label='data') plt.plot(x,y, linewidth=3.0, label='fit') plt.legend(loc='best') plt.ylabel('GDP') plt.xlabel('Year') plt.show()</pre>
<pre># split data into train/test msk = np.random.rand(len(df)) &lt; 0.8 train_x = xdata[msk] test_x = xdata[msk] train_y = ydata[msk] test_y = ydata[msk]  # build the model using train set popt, pcov = curve_fit(sigmoid, train_x, train_y)  # predict using test set y_hat = sigmoid(test_x, *popt)  # evaluation print("Mean absolute error: %.2f" % np.mean(np.absolute(y_hat - test_y))) print("Residual sum of squares (MSE): %.2f" % np.mean((y_hat - test_y) ** 2)) from sklearn.metrics import r2_score print("R2-score: %.2f" % r2_score(test_y,y_hat) )  Mean absolute error: 0.04 Residual sum of squares (MSE): 0.00 R2-score: 0.96</pre>		0.4 0.2 0.0 0.975 0.980 0.985 0.990 0.995 1.000
<pre>y_hat = sigmoid(test_x, *popt)  # evaluation print("Mean absolute error: %.2f" % np.mean(np.absolute(y_hat - test_y))) print("Residual sum of squares (MSE): %.2f" % np.mean((y_hat - test_y) ** 2)) from sklearn.metrics import r2_score print("R2-score: %.2f" % r2_score(test_y,y_hat))</pre> Mean absolute error: 0.04 Residual sum of squares (MSE): 0.00 R2-score: 0.96	3]:	<pre># split data into train/test msk = np.random.rand(len(df)) &lt; 0.8 train_x = xdata[msk] test_x = xdata[~msk] train_y = ydata[msk] test_y = ydata[~msk]  # build the model using train set popt, pcov = curve_fit(sigmoid, train_x, train_y)</pre>
		<pre>print("Mean absolute error: %.2f" % np.mean(np.absolute(y_hat - test_y))) print("Residual sum of squares (MSE): %.2f" % np.mean((y_hat - test_y) ** 2)) from sklearn.metrics import r2_score print("R2-score: %.2f" % r2_score(test_y,y_hat))</pre> Mean absolute error: 0.04 Residual sum of squares (MSE): 0.00