: [from <mark>sklear</mark>			
	#load data, def load_da	in sparse format ta(filename): en('dorothea/%s.data' % file	name)	
	lin if	<pre>in range(100000): de = f.readline() len(line) == 0: print ('done') break de = [int(x) for x in line.spectrum (line)</pre>	plit()]	
· ·	I.e return def load_la	extend(line) extend(k for i in range(len(ss.coo_matrix((np.ones(len(abels(filename): en('dorothea/%s.labels' % fi	I)), (I,J)))	
	<pre>y = [] for k i lin if</pre>	<pre>in range(100000): de = f.readline() len(line) == 0: print('done') break deppend(int(line))</pre>		
2	return Xtrain = lo ytrain = lo Xtest= load	np.array(y) pad_data('dorothea_train') pad_labels('dorothea_train') d_data('dorothea_valid')		
		d_labels('dorothea_valid') n.shape, Xtest.shape, Xtra	in.nnz)	
: 2	Q1) Spars	7 = Xtrain.todense()	V+ main mas)	
: [:	(800,) (800 ss.coo_matr <800x100001	n.shape, xTrainNumpy.shape 1, 100001) 727760 rix(Xtrain) 2 sparse matrix of type ' <claim '<claim="" of="" of<="" td="" type=""><td>ass 'numpy.float64'>'</td><td></td></claim>	ass 'numpy.float64'>'	
s I I	sparsity = print("Numb print("Numb print("Spar	<pre>v = Xtrain.todense() 1.0 - (Xtrain.nnz / float(x') per of Zero Elements: "+ str per of Non Zero Elements: "+ esity : % 0.3f" % (sparsity))</pre>	<pre>(xTrainNumpy.size - count_non str(Xtrain.nnz))</pre>	nzero(xTrainNumpy)))
2	Number of N Sparsity: def get_cla uniqueV total =	on Zero Elements: 727760	r,return_counts= True)	
	negativ for ind if els	<pre>reValues = 0 dex, value in enumerate(uniq value == 1: positiveValues = counts[in- se: negativeValues = counts[in-</pre>	dex]	ng please check"
I	print ('Tra print ('Tes Training cl		rently %.2f +1, %.2f -1' % ge y %.2f +1, %.2f -1' % get_cl 10 +1, 0.90 -1	
1	1) dorothea dat	nents on class_balanc tset is highly imbalanced dataset with the for both training and test sets.	ee and sparsity h only 10% positives (rare events) and 9	90& normal events.
r	3) The number number of feat	of training examples in this dataset is tures in dorothea dataset is 100,000 a sclass (y, yhat):	ements as zeros and only 0.9% non-ze is almost of the same order as compare as compared to only 54 features in covt	ed to covtype dataset (800 vs 468) but
	<pre>for i i if return def get_f1(</pre>	<pre>imples = len(y) in range(totalSamples): y[i] != yhat[i]: incorrect += 1 incorrect / totalSamples (y, yhat): imples = len(y)</pre>		
	trueNeg falsePo falseNe for i i	<pre>stives = 0 gatives = 0 sitives = 0 egatives = 0 .n range(totalSamples): y[i] == 1 and yhat[i] == 1: truePostives +=1</pre>		
	eli eli	<pre>f y[i] == -1 and yhat[i] == trueNegatives +=1 f y[i] == -1 and yhat[i] == falsePositives +=1 f y[i] == 1 and yhat[i] == falseNegatives +=1</pre> <pre>con = truePostives/(truePost</pre>	-1:	
	<pre>assert(is wrong" return depth = 3</pre>	<pre>= truePostives/(truePostive ((truePostives + trueNegative (2 * (precision* recall))/() DecisionTreeClassifier(crite)</pre>	res + falsePositives + falseNe	egatives) == totalSamples),"s
I	clf = clf.f print(get_m	spli max_c clas fit(Xtrain, ytrain)	<pre>tter='best', depth=depth, s_weight='balanced') (Xtrain)), get_f1(ytrain, cl</pre>	f.predict(Xtrain)))
: 5	yPredTrain	= clf.predict(Xtrain) = clf.predict(Xtest)		
1		<pre>pend(("train",</pre>	ain, yPredTrain), ain),	
	columns = [<pre>pend(("test",</pre>	est, yPredTest), (t), (est)))	
(pd.set_opti df = pd.Dat	'missclassification-score ('missclassification-score ('F1-score (custom)', 'F1-score (in-built)',] con("display.max_colwidth",2 caFrame(results, columns=colex('dataset')	<pre>in-built)',)</pre>	
:	df _			stom) F1-score (in-built) 72414 0.772414 03175 0.603175
1	1) F1 score is a	a better metric than misclassification	if we say every event id not a rare-ever	-
3	So, using F1 so Without a	core is a better metric, as it gives wei any cross-validation coreAndPlot (Xtrain, ytrain, Xt	ightage for both precision and recall an	-
	depth results for d i	= list(range(2,11))		
	yPr yPr res	<pre>E = clf.fit(Xtrain, ytrain) redTrain = clf.predict(Xtrain) redTest = clf.predict(Xtest) redTest = clf.predict(Xtrain, ytrain) redTest = clf.predict(Xtest) re</pre>	yPredTrain),	
	pd.set_ df = pd	'F1-score (train)', 'F1-score (test)',] option("display.max_colwidt. DataFrame(results, columns.		='psql'))
	<pre>testF1 plt.fig x = dep plt.plo</pre>	= [resultsAtAGivenDepth[2] gure(figsize=(14,8)) th # Create domain for plot ot(x, trainF1, label='Traini	ng F1 Score') # Plot training	results] g error over domain
2	plt.xla plt.yla plt.tit xis	abel('Maximum Depth', fontsize abel('F1 Score', fontsize=12) ale('Depth Vs F1 Score(Test spend() # Show plot labels as bw()	<pre># Label y-axis and Train) Without any cross-</pre>	
: 1	testF1Score ++ depth + 2	F1-score (train) F1-s	coreAndPlot(Xtrain,ytrain,Xte	est,ytest)
	3 4 5 6 7 8 9	0.872483	0.580645 0.615385 0.571429 0.626866 0.483871 0.597015 0.571429 0.575758	
+	— Те	Depth Vs F1 Score(raining F1 Score esting F1 Score	+ (Test and Train) Without an	ny cross-Validation
	0.9 - 0.8 -			
	0.6 -			
	0.5 -	3 4	5 6 7	8 9 10
I	2	ross-validation (k=5)	5 6 7 Maximum Depth	8 9 10
: 2	<pre>import rand randomIndic xSplit = [] ySplit = [] for i in ra randomR</pre>	coss-validation (k=5) dom ces = random.sample(range(80) ange(5): RowIndices = randomIndices[(Maximum Depth 0), 800) i*160):(i*160)+160]	8 9 10
	import rand randomIndic xSplit = [] ySplit = [] for i in ra randomR xSplit. ySplit. def prepare assert assert xTrainW	coss-validation (k=5) lom lom longe(5): RowIndices = randomIndices[(append(xTrainNumpy[randomRowIndi append(ytrain[randomRowIndi eTrainSetForKFoldCV(xTrain,y) np.shape(xSplit) == (5,160, np.shape(ySplit) == (5,160) WithoutValidationSet = []	Maximum Depth 0), 800) i*160):(i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain"	8 9 10
	import rand randomIndic xSplit = [] ySplit = [] for i in ra randomR xSplit. ySplit. ySplit. def prepare assert assert xTrainW yTrainW for ind	coss-validation (k=5) lom lom longe(5): RowIndices = randomIndices[(append(xTrainNumpy[randomRowIndi append(ytrain[randomRowIndi append(y	<pre>Maximum Depth i*160): (i*160)+160] iwIndices]) ces]) Train, indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" ctionSet) == 0: inSet = xTrain[index] inSet = yTrain[index] inSet = np.concatenate([xTrain]</pre>	nWithoutValidationSet, xTrain
	import rand randomIndic xSplit = [] ySplit = [] for i in ra randomR xSplit. ySplit. ySplit. def prepare assert xTrainW yTrainW yTrainW for ind if return def buildTr	coss-validation (k=5) clom clom close = random.sample(range(80) close = random.sample(random.sample(som)) close = random.sample(close = random.sample(som) close = random.sample(close =	<pre>Maximum Depth i*160): (i*160)+160] wIndices]) ces]) Train, indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" ctionSet) == 0: cnSet = xTrain[index] cnSet = yTrain[index] cnSet = np.concatenate([xTrain yTrainWithoutValidationSet) cin, c, yValidation,</pre>	nWithoutValidationSet, xTrain
	<pre>import rand randomIndic xSplit = [] ySplit = [] for i in ra randomR xSplit. ySplit. def prepare assert xTrainW yTrainW yTrainW for ind if return def buildTr clf = t clf = c yPredTr yPredVa</pre>	coss-validation (k=5) lom longe(5): lowIndices = randomIndices[(append(xTrainNumpy[randomRowIndi longe(5): lowIndices = randomIndices[(append(xTrainNumpy[randomRowIndi longe(5): longe(1): longe(1): longe(2): longe(3): longe(3): longe(3): longe(4): longe(5): longe(4): longe(5): longe(5): longe(5): longe(5): longe(5): longe(5): longe(5): longe(5): longe(1): longe(5): longe(5): longe(1): longe(5): longe(5): longe(5): longe(5): longe(1): lo	<pre>Maximum Depth i*160): (i*160)+160] wIndices]) ces]) Train, indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" tionSet) == 0: mSet = xTrain[index] mSet = yTrain[index] mSet = np.concatenate([xTrain yTrainWithoutValidationSet in, yYvalidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in)</pre>	NWithoutValidationSet, xTrain
	K-fold creating and an and an and an	coss-validation (k=5) lom les = random.sample(range(80) longe(5): lowIndices = randomIndices[(append(xTrainNumpy[randomRowappend(ytrain[randomRowIndiances]] longe(5): longe(5): longe(5): longe(1): longe(2): longe(2): longe(3): longe(4): longe(5): longe(6): longe	<pre>Maximum Depth i*160): (i*160)+160] wIndices]) ces]) Train, indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" tionSet) == 0: mSet = xTrain[index] mSet = yTrain[index] mSet = np.concatenate([xTrain yTrainWithoutValidationSet) in, yTvalidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in) idation)</pre>	NWithoutValidationSet, xTrain
	K-fold creating import rand randomIndic exsplit = [] ysplit = [] for i in randomR xsplit. ysplit. def prepare assert assert xTrainW yTrainW for ind if return clf = t clf = c yPredTr yPredVa return depth = l allValidati averageVali allResults for d in de results for i i xTr xVa yVa	coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(coss = 10) co	<pre>Maximum Depth i*160): (i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" tionSet) == 0: wnSet = xTrain[index] wnSet = yTrain[index] wnSet = np.concatenate([xTrain yTrainWithoutValidationSet in, , yValidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in) idation) , get_f1(YValidation, yPredValidation) reforkFoldCV(xSplit,ySplit,i) e = buildTreeAndgetF1Score(XTXV)</pre>	WithoutValidationSet, xTrain WithoutValidationSet, yTrain Alidation)
	K-fold creating and an andom Indic and an analom and an analom and an analom and an analom and an	coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(coss = 10) co	Maximum Depth (0), 800) i*160):(i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" tionSet = 0: wnSet = xTrain[index] wnSet = pyTrain[index] wnSet = np.concatenate([xTrain yTrainWithoutValidationSet in, ,,YValidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in) idation) , get_fl(YValidation, yPredValidation) , get_fl(YValidation, yPredValidation) ctForKFoldCV(xSplit,ySplit,i) e = buildTreeAndgetFlScore(XT xv i, ralidationFlScore)	WithoutValidationSet, xTrain(WithoutValidationSet, yTrain(WithoutValidationSet, yTrain(WithoutValidation))
	K-fold creating and an and an analyse is a sert assert assert assert assert assert assert arain and an	coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(random.pos) coss = random.sample(random.pos) coss = random.sample(range(80) coss = random.sample(range(80) coss = random.pos(varian,y coss = random.pos(varian) coss = random.p	Maximum Depth (i*160):(i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" tionSet) == 0: mSet = xTrain[index] mSet = pyrain[index] mSet = np.concatenate([yTrain yTrainWithoutValidationSet in, ,YValidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in) idation) , get_f1(YValidation, yPredValidation, yPredValidation) tForkFoldCV(xSplit,ySplit,i) e = buildTreeAndgetF1Score(XT XV	WithoutValidationSet, xTrain WithoutValidationSet, yTrain alidation) Prain, yTrain, Validation, yValidation, d)
	K-fold created randomIndic exsplit = [] ysplit = [] ysplit = [] for i in randomR xsplit. ysplit. def prepare assert assert xTrainW yTrainW for ind if return def buildTr clf = t clf = c yPredTr yPredVa return depth = l alvalidatial allResults for d in de results for i i i i i i i i i i i i i i i i i i i	Coss-validation (k=5) Lom Les = random.sample(range(80) LowIndices = randomIndices[(append(xTrainNumpy[randomRoiappend(ytrain[randomRowIndi LowIndices = randomIndices[(append(xTrainNumpy[randomRoiappend(ytrain[randomRowIndi LowIndices = (5,160, np.shape(xSplit) == (5,160, np.shape(ySplit) == (5,160) InthoutValidationSet = [] Low in range(5): Low in range(5): Low index!=i: Low if len(xTrainWithoutValidation yTrainWithoutValidation yTrainWithoutValidation yTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation yTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidation yTrainWithoutValidation xTrainWithoutValidation xTrainWithoutValidat	Maximum Depth (i*160):(i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain" tionSet) == 0: mSet = xTrain[index] mSet = xTrain[index] mSet = np.concatenate([xTrain yTrainWithoutValidationSet in, yYValidation, criterion='entropy', splitter='best', max_depth=d, class_weight='balanced') in) idation) , get_f1(YValidation, yPredValidation, yPredValidation) tForKFoldCV(xSplit,ySplit,i) e = buildTreeAndgetF1Score(XT xV xv in	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation, yValidation, d) 10
	K-fold created and mandom and man	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(range(s), 160) coss = random.sample(range(s), 160) coss = random.sample(range(s), 160) coss = random.sample(range(s), 160) coss = random.sample(randex) coss = random.sample(randex) coss = random.sample(randex) coss = random.sample(range(s), 160) coss = random.sample(randex) coss = random.sample(randex) coss = random.sample(randex) coss = random.sample(range(s), 160) coss = random.sample(randex) coss =	Maximum Depth (i*160):(i*160)+160] wIndices]) ces]) (Train, indexOfTheValidation): 100001), "Check xTrain" , "Check yTrain" (tionSet) == 0: inSet = xTrain[index] inSet = yTrain[index] inset = np.concatenate([xTrain yTrainWithoutValidationSet in, , YValidation, (criterion='entropy',	WithoutValidationSet,xTrain WithoutValidationSet,yTrain WithoutValidationSet,yTrain Oracle (validation) O.545455 O.682696 O.774194 O.774194 O.70666667
	K-fold crows import random random Indice in random Rasplit. The second of the second in the second i	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (coss-coss-coss-coss-coss-coss-coss-coss	Maximum Depth (1) (i*160):(i*160)+160] (i*160):(i*160)+160] (imidices]) (ces]) (Train,indexOfTheValidation): (100001), "Check xTrain" ("Check yTrain" tionSet) == 0: (nSet = xTrain[index] (nset = yTrain[index] (nset = np.concatenate([xTrain (xTrainWithoutValidationSet) (in, (,YValidation, (criterion='entropy',	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain "rain, yTrain,
	K-fold crown import random random Indic explit = [] ysplit = [] ysplit = [] ysplit. The random explit explit expressed in the random explit expressed expres	coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(range(s), foother) coss = random.nample(range(s), foother) coss = random.nample(solog) co	Maximum Depth (a)	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain ''paql')) ''psql')) ''psql')) ''psql')) ''psql')) ''psql')) ''psql')) ''psql') ''psql')
	K-fold creating and an analysis of a verage print (to depth lall validation and a verage print (to depth lall valid	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = randomIndices[(append(xTrainNumpy[randomRoappend(ytrain[randomRowIndiappend(ytrain[randomRowIndiappend(ytrain[set][cos][cos][cos][cos][cos][cos][cos][cos	Maximum Depth (a) (i*160):(i*160)+160] wIndices]) ces]) Train,indexOfTheValidation): 100001), "Check xTrain" (check yTrain" tionSet) == 0:	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation Wi
	K-fold Cramport rand random ra	coss-validation (k=5) tom tes = random.sample(range(80) tinge(5): towIndices = randomIndices[(Maximum Depth (1) (1*160): (1*160)+160] wIndices]) (ces]) Train,indexOfTheValidation): 100001), "Check xTrain" ("Check yTrain" tionSet) == 0:	WithoutValidationSet,xTrain WithoutValidationSet,yTrain WithoutValidationSet,yTrain WithoutValidationSet,yTrain WithoutValidationSet,yTrain WithoutValidation, Validation,YValidation, d) 'psql'))
	K-fold crows import rand random Indic Indication Indic Indic Indic Indic Indic Indic Indic Indic Indic Indi	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(80) coss = (5)	Maximum Depth (a)	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain Validation, yValidation, d) 'psq1'))
	K-fold craimport rand import rand randomIndic xsplit = [] ysplit = [] for i in ra randomR xsplit. ysplit. def prepare assert xTrainW yTrainW for ind if return def buildTr clf = t clf = t clf = c yPredTr yPredVa return depth = l allValidatia averageValia allResults for d in de results for d in de results for i all res all col df average print(t depth	coss-validation (k=5) coss = random.sample(range(80) coss = random.sample(range(s) coss = random.nample(s)	Maximum Depth (interpretation of the property	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain Alidation) **Prain, yTrain, **(alidation, yValidation, d) **(b) **(a) **(b) **(a) **(b) **(a) **(b) **(a) **(a) **(b) **(a) **(a) **(b) **(a)
	K-fold Crambor random Indic candom Indic casplit = [] for i in random xsplit. Ysplit. defor i in random xsplit. Ysplit. def prepare assert assert xTrainW yTrainW for ind if for inde results for inde results for i independent all all all average Valial all results for i independent for i independen	Coss-validation (k=5) Low Low Loss = random.sample(range(R0 Loss) = random.sample(range(Loss)) Loss) = random.sample(range(S0 Loss) = random.sample(range(S0 Loss) = random.sample(range(S0 Loss) = random.sample(range(S) Loss) = random.sample(R) Loss) = ra	Maximum Depth	WithoutValidationSet, XTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation, //alidation, yValidation, d) O.571429 O.545455 O.608696 O.774194 O.608667 O.516129 O.666667 O.551724 O.666667 O.551724 O.685714 O.685714 O.685714 O.685714 O.686667 O.69060 O.68667 O.69060 O.
		COSS-validation (k=5) Low Indices = random.sample (range(80 append (xTrainNumpyIrandomNo append (xTrainNumpyIrandomNo append (xTrainSetForKFoldCV (xTrain, ynp.shape(xSplit) == (5,160) ithoutValidationSet = (1) ithoutValidationSet, xTrainNithoutValidation yrrainNithoutValidation yrrainNithoutValidation yrrainNithoutValidation xTrainNithoutValidation yrrainNithoutValidation yrrainNithoutValidation yrrainNithoutValidation in the color of the col	### Maximum Depth (a) (1000) (1100)	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain Indication, yTrain, Validation, yTrain, Valid
		COSS-validation (k=5) Lom Lom Lom Lom Lom Lom Lom Lom Lom Lo	Maximum Depth Maximum Depth (1-160): (i-160)+160] windices]) ceal) Train, indexOfTheValidation): 100001), "Check wTrain" tionSet) 0:	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation, William Comment of the Comment o
		COSS-validation (k=5) Low Low Low Low Low Low Low Low Low Lo	Maximum Depth	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation, withoutValidation, 'psql') ore (validation) 0.571429 0.454555 0.62669 0.52699 0.566667 0.516129 0.666667 0.516129 0.666667 0.531724 0.685714
	K-fold croimport rand import rand randomIndic xSplit = [] ySplit = [] ySplit = [] for in ra	coss-validation (k=5) com coss = random.sample(range(80) coss = random.sample(range(5)) coss = random.sample(coss) coss = range(5) coss = random.sample(coss) coss = random.sample(coss	### Maximum Depth Maximum Depth	inithoutValidationSet, xTrain whithoutValidationSet, yTrain whithoutValidationSet, yTrain whithoutValidationSet, yTrain whithoutValidation whithou
		coss-validation (k=5) com coss = random.sample (range (80) coss = random.sample (range (20) coss = (5,160) coss = (5,160) coss = (6,160) coss = (6,160) coss = (7,160) coss = (1,160)	### Maximum Depth ### Maximum D	(i) (ii) (iii) (iiidation) (iidation) (ii
	K-fold crade management of the first state of the f	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss = random.sample(range(8) mage(5): cowIndices = randomIndices(append(xfrainNumpy(randomRowIndices)(append(xfrainNumpy(randomRowIndices)(append(xfrainNumpy(randomRowIndices)(append(xfrainNumpy(randomRowIndices)(append(xfrainNumpy(randomRowIndices)(itchoutValidationSet = [] cost in range(5): if lean(xfrainNithoutValidation	### Maximum Depth Maximum Depth	withoutValidationSet, xTrain withoutValidationSet, yTrain withoutValidationSet, yTrain withoutValidationSet, yTrain withoutValidation, (a) (b) (real, yTrain, (rain, yValidation, (a) (b) (c) (validation, yValidation, (c) (c) (c) (c) (c) (c) (c) (c
	K-fold Cr import rand randomIndic rxSplit = [] rySplit = [] refor i in randomR	coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=5) coss-validation (k=1) coss-validation (k=1) coss-validation (k=1) coss-validation (coss-validation (ytrain) ytraindown (ytrain) ytraindown (ytrain) yraindown (ytrain) yraindown (ytrain) yraindown (ytrain) yraindown (ytrain) yraindown (ytrain) (ytrain) (ytraindown (ytrain) (ytraindown (ytraindow	Maximum Depth (a), 800) (i*160):(i*160)*160] windices]) (ces]) windices]) (ces]) train,indexOfThevalidation): (10001), "Check wirain" promise = wirain[index] mase = wirain[index] mase = wirain[index] mase = pr.concatenate((yirain miset - np.concatenate((yirain miset - np.concatenate((yirain miset - max depth-d, class weight='balanced') in, (yvalidation, critorion='cntropy', splitter='best', max depth-d, class weight='balanced') indiction) , get_fl(vvalidation, yFredvalidation) promise = buildTrecAndgetFlScore(Xivalidation) crel) decv', con)'] unna-columns) 'fl-score(validation)'].mean '(), headers='keys', tablefint= promise = consection Fl-score() c. 730495 c. 730495 c. 730495 c. 730496 c	WithoutValidationSet, xTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidationSet, yTrain WithoutValidation, yTrain WithoutValidation, yTrain WithoutValidation, yTrain WithoutValidation, yTrain WithoutValidation, yTrain WithoutValidation, wTrain WithoutValidation WithoutValidat
	K-fold cr import rand randomIndic xSplit = [] ySplit = [] ySplit in randomR xSplit. ySplit. define randomR xSplit. ySplit. define prepare assert xTrainW yTrainW for ind if return def buildTr clf = c yPredTr yPredVa return def buildTr clf = t clf = c yPredTr yPredVa return def buildTr clf = t for in all clf = t clf = c yPredTr yPredVa return depth = l all all all all all all all all all a	OSS-validation (k=5) OSS-validation (k=5) OSS-validation (k=5) OSS-validation (k=5) OSS-validation (k=5) OSS-validation (k=6) OSS-validation (k=6) OSS-validation (k=6) OSS-validation (c=6)	### Maximum Depth	WithoutValidationSet, x?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidation, Alidation, yvalidation, yvalidation, Alidation, yvalidation, yvalidatio
		OSS-Validation (k=5) OSS-Validation (k=5) OSS-Validation (k=5) OSS-Validation (k=5) OSS-Validation (k=6)	### Maximum Depth ### Maximum Depth	WithoutValidationSet, x?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidation, Alidation, yvalidation, yvalidation, Alidation, yvalidation, yvalidatio
		OSS-Validation (k=5) Ion where = random.sample(range(8) sampe(5):	### Maximum Depth ### Maximum Depth	WithoutValidationSet, x?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidation, Alidation, yvalidation, yvalidation, Alidation, yvalidation, yvalidatio
	K-fold crade import rand random rando	OSS-Validation (k=5) Ion where = random.sample(range(8) sampe(5):	### Maximum Depth ### Maximum Depth	WithoutValidationSet, x?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidationSet, y?rain WithoutValidation, Alidation, yvalidation, yvalidation, Alidation, yvalidation, yvalidatio
		Coss-validation (k=5) Coss-validation (k=5) Coss = random.sample(range(8) coss = random.sample(sex) coss = r	### Maximum Depth ### Maximum Depth	### A
		Coss-validation (k=5) Coss-validation (k=5) Coss = random.sample(range(8) coss = random.sample(sex) coss = r	### Assistance Place Place	### A
		Coss-validation (k=5) Coss-validation (k=5) Coss-validation (k=5) Coss-validation (coss-validation) Coss-validation (### Assistance Place Place	### A
		Coss-validation (k=5) Coss-validation (k=5) Coss-validation (k=5) Coss-validation (coss-validation) Coss-validation (### Assistance Place Place	### A
		Coss-validation (k=5) Coss-validation (k=5) Coss-validation (k=5) Coss-validation (coss-validation) Coss-validation (### Assistance Place Place	### A
		oss-validation (k=5) tom so = random.samole(range (8) towner(s): cwindicos = rendomination(sappend(strain) and samole(se): sappend(strain) and samole(se): sappend(strain) = (5,50) rendomination(strain) = (5,50) re	Maximum Depth (1, 809) (MithoutValidationNet.wTrain MithoutValidation Train,yTrain, Alidation ()
		oss-validation (k=5) tom so = random.samole(range (8) towner(s): cwindicos = rendomination(sappend(strain) and samole(se): sappend(strain) and samole(se): sappend(strain) = (5,50) rendomination(strain) = (5,50) re	Maximum Depth (a), 600) 1=160): (1=160): (1=160) Windices:) (est) Whether of the state of	MithoutValidationNet.wTrain MithoutValidation Train,yTrain, Alidation ()
		oss-validation (k=5) tom so = random.samole(range (8) towner(s): cwindicos = rendomination(sappend(strain) and samole(se): sappend(strain) and samole(se): sappend(strain) = (5,50) rendomination(strain) = (5,50) re	Maximum Depth (a), 600) 1=160): (1=160): (1=160) Windices:) (est) Whether of the state of	MithoutValidationNet.wTrain MithoutValidation Train,yTrain, Alidation ()

```
In [16]: plt.figure(figsize=(14,8))
    depth = list(range(2,11))

plt.plot(depth, averageValidationScores, label='Average F1 Score for Validation Set') # Plot training e
    rror over domain
    testF1Scores = list(testF1ScoresWithoutValidation['F1-score (test)'])
    plt.plot(depth, testF1Scores, label='F1 Score for Training Set') # Plot testing error over domain
    plt.xlabel('Maximum Depth',fontsize=12) # Label x-axis
    plt.ylabel('F1 Score',fontsize=12) # Label y-axis
    plt.title('Depth Vs F1 Score(TestSet and Average of Validation Sets) ',fontsize=18) # Label y-axis
    plt.legend() # Show plot labels as legend
    plt.show()
```



