



07/19/21  
16:57:41

# 2

```

193: # little
194: # little
195: def contrasts(
196:     here, there, t):
197:     "Report the ranges that are most different in two classes."
198:     like(d, kl):
199:         out = prior = (hs[kl] + K) / (n + K*2)
200:         out = prior = (hs[kl] + K) / (n + K*2)
201:         for at, span in d.items():
202:             f = has.get((kl, (at, span)), 0)
203:             out *= (f + M*prior) / (hs[kl] + M)
204:             f = has.get((kl, (at, span)), 0)
205:             out *= (f + M*prior) / (hs[kl] + M)
206:         return out
207:
208: 202:
209: def val(d):
210:     return GOAL(like(d, True), like(d, False)), d
211: 204: top(a):
212:     return sorted(a, reversed=True, key=first)[TOP]
213:
214: has = {(kl, (at, (lo, hi))): f
215:         205:
216:         has = {(kl, (at, (lo, hi))): f
217:                 for col1, col2 in zip(here.cols.x, there.cols.x)
218:                 208: f, kl, (at, (lo, hi)) in col1.bins(col2))
219:
220: n = len(here.rows, there.rows)
221: hs = {True: len(here.rows), False: len(there.rows)}
222: 213: os = [val(dict(good), 200): n = len(here.rows, there.rows)
223:        218: {True: len(here.rows), False: len(there.rows)}
224: 219: 215: os = [val(dict(at=x)) for at, x in set({z for z in has})]
225: 214: ranges = {}
226: 215: ranges = {}
227: 213: for d, d in top(solos):
228: 216: 214: in d:
229:         ranges[k] = ranges.get(k, set()).add(d[k])
230: 218: 215: ranges[k] = ranges.get(k, set()).add(d[k])
231: 216: for rule in top([val(d) for d in dict_product(ranges)]):
232: 219: 217: (rule)
233: 220: 319: f
234: 221: 218: -----
235: 222: 219: # little
236: 220: class Eg:
237: 223:     "Unit tests."
238: 224: 221: is():
239: 225:     222: all examples."
240: 226: 220: ("examples:")
241: 227: 224: k, f in vars(Eg).items():
242: 228:     224: 0] != " ":
243: 229:     f" {k:<13} {f.__doc__}")

```

"democrat"):

k="positive"):

```

281:276 def first(a): return a[0]
282:277 def last(a): return a[-1]
283:278 def per(a, p=5): return a[int(p*len(a))]
284:
285:279:-----
286:280:
287:281: class o(object):
288:282:     "objects"
289:283:     def __init__(i, **k): i.__dict__.update(**k)
290:284:     def __getitem__(i, k): return i.__dict__[k]
291:285:     def __repr__(i): return i.__class__.__name__+str(public(i.__dict__))
292:286:     def __setitem__(i, k, v): i.__dict__[k] = v
293:287:     # -----
294:288:-----
295:289: # in
296:290: def csv(f=None, sep=","):
297:291:     "read csv files"
298:292:     def prep(s): return re.sub(r'([\\tr ]|#\.)', '', s)
299:293:     with open(f) as fp:
300:294:         for s in fp:
301:295:             s = prep(s)
302:296:             yield s.split(sep)
303:297:     in sys.stdin:
304:298:         yield s.split(sep)
305:299:     # -----
306:300:
307:301: def cli(f):
308:302:     "Drive command line flags from function annotations."
309:303:     p = parse(prog=f.__name__, parser=parse, description=f.__doc__,
310:304:               formatter_class=textual)
311:305:     for (k, h), b4 in zip(
312:306:         list(f.__annotations__.items()), f.__defaults__):
313:307:         zip(list(f.__annotations__.items()), f.__defaults__):
314:308:         argument(303:(k[0].lower()), action="store_true", default=False, action=
315:309:         argument(306:(k[0].lower()), action="store_true", default=b4,
316:310:         help=h+307+str(b4)+"", type=type(b4),
317:311:         metavar=k)
318:312:         gs().__dict__

```