# Tutorial 3

## Outline

- Scikit-learn:
  - Data preprocessing: StandardScaler (for continuous featrues), OnehotEncoder (for categorical features)
  - Data splitting
- Artificial Neural Network:
  - Activation function
- RDKit
  - SMILES: Simplified Molecular Input Line Entry System

## 1. Scikit-learn

A package that provides implementation of various machine learning algorithms (including supervised learning and unsupervised learning), as well as tools for data preprocessing and analysis.

Documentations:

- Scikit-learn
- StandardScaler
- OneHotEncoder
- KFold

### StandardScaler

$$X_{\text{scale}} = \frac{X - \text{avg}(X)}{\text{std}(X)}$$

In [38]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

In [39]:
```
df = pd.read_csv("Datasets/titanic.csv")
df.head()
# data cleaning
subdf = df[["Pclass", "Sex", "SibSp", "Parch", "Embarked", "Age", "Fare", "Survived

# Categorical features: ["Pclass", "Sex", "Embarked"]
categorical_features = subdf[["Pclass", "Sex", "Embarked"]]
# Continuous features: ["Age", "Fare", "SibSp", "Parch"]
```

```
continuous_features = subdf[["Age", "Fare", "SibSp", "Parch"]]

X_cont = continuous_features.values # n_sample * n_feature
print(X_cont.shape)
```

```
(712, 4)
```

In [40]:
```
# calculate avg. and std.
print("Avg:", np.mean(X_cont, axis=0))
print("Std:", np.std(X_cont, axis=0))
```

```
Avg: [29.6420927  34.5672514   0.51404494  0.43258427]
Std: [14.4827517  52.9014591   0.93003832  0.85358139]
```

In [41]:
```
# scale (or normalize)
scaler = StandardScaler()
X_norm = scaler.fit_transform(X_cont)
```

In [42]:
```
# avg. and std. of scaled data
print("After scaling:")
print("Avg:", np.mean(X_norm, axis=0))
print("Std:", np.std(X_norm, axis=0))
```

```
After scaling:
Avg: [ 2.94396218e-16 -6.73618464e-17 -1.49692992e-17  1.49692992e-17]
Std: [1. 1. 1. 1.]
```

## OnehotEncoder

Good approach to represent categorical features

| Fruits | Label Encoding | One-hot Encoding |
|---|---|---|
| Apple | 1 | [0, 1] |
| Banana | 2 | [1, 0] |

In [43]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [44]:
```
# find three catagorical features as an example
X_cate = categorical_features.values
X_cate
```

Out[44]:
```
array([[3, 'male', 'S'],
       [1, 'female', 'C'],
       [3, 'female', 'S'],
       ...,
       [1, 'female', 'S'],
       [1, 'male', 'C'],
       [3, 'male', 'Q']], dtype=object)
```

In [45]:
```
# OneHotEncoding
encoder = OneHotEncoder()

X_onehot = encoder.fit_transform(X_cate).toarray()
```

```
print(X_onehot.shape)
X_onehot
```

```
(712, 8)
```

Out[45]:
```
array([[0., 0., 1., ..., 0., 0., 1.],
       [1., 0., 0., ..., 1., 0., 0.],
       [0., 0., 1., ..., 0., 0., 1.],
       ...,
       [1., 0., 0., ..., 0., 0., 1.],
       [1., 0., 0., ..., 1., 0., 0.],
       [0., 0., 1., ..., 0., 1., 0.]])
```

In [46]:
```
# access categories
encoder.categories_
```

Out[46]:
```
[array([1, 2, 3], dtype=object),
 array(['female', 'male'], dtype=object),
 array(['C', 'Q', 'S'], dtype=object)]
```

In [47]:
```
# tranform from One-Hot encoding back to original data
X_test = X_onehot[:3]
print(X_test)
encoder.inverse_transform(X_test)
```

```
[[0. 0. 1. 0. 1. 0. 0. 1.]
 [1. 0. 0. 1. 0. 1. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 1.]]
```

Out[47]:
```
array([[3, 'male', 'S'],
       [1, 'female', 'C'],
       [3, 'female', 'S']], dtype=object)
```

# Data splitting

## Overfitting

No description has been provided for this image

Split data to Train/Validation/Test set can resolve this issue to some extent:

No description has been provided for this image

No description has been provided for this image

Prepare the Titanic data:

In [48]:
```
# combine categorical & continuous features
X = np.hstack((X_norm, X_onehot))

# it is recommended to reshape the outputs
# to (n_samples, 1) in order to avoid unexpected broadcasting
y = subdf['Survived'].values.reshape(-1, 1)
```

```
# print the dimensions
print(X.shape, y.shape)
```

```
(712, 12) (712, 1)
```

## Train-test split

In [49]:
```
from sklearn.model_selection import train_test_split
```

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [51]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(569, 12) (569, 1)
(143, 12) (143, 1)
```

## K-Fold

No description has been provided for this image

In [52]:
```
from sklearn.model_selection import KFold
```

In [53]:
```
# Kfold
kf = KFold(n_splits=5, shuffle=True)

for train_index, test_index in kf.split(X):
    print(train_index, test_index)
```

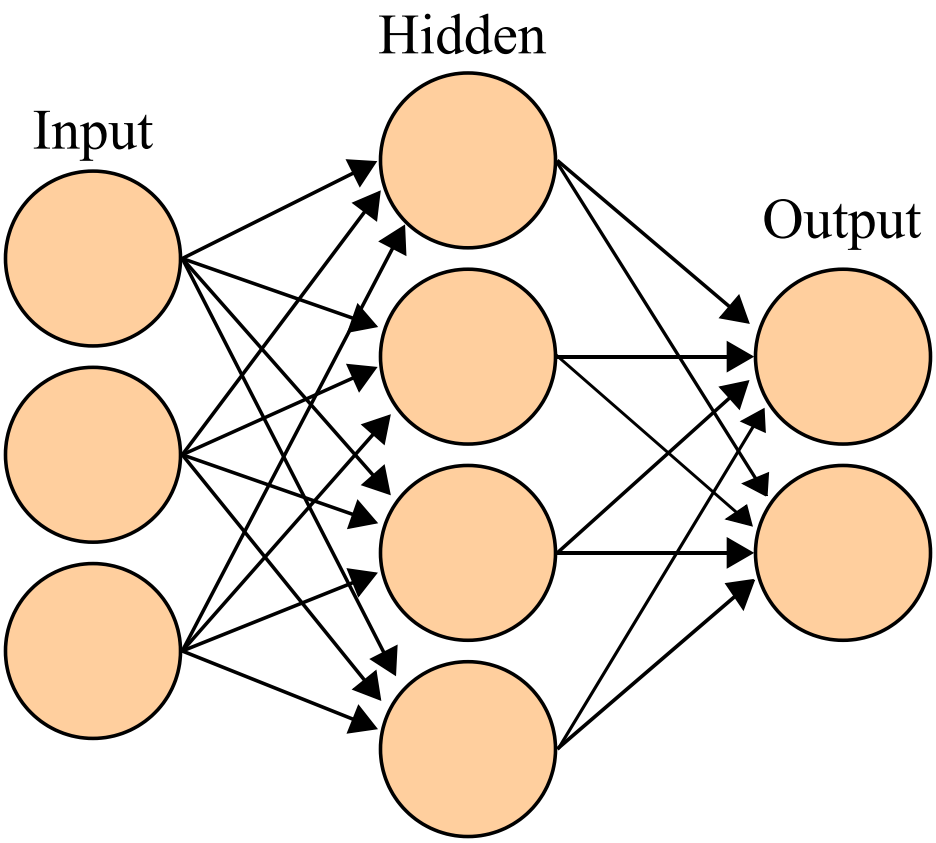
```
[  0   1   2   3   4   5   7   8   9  10  11  12  14  15  17  19  20  21
  22  23  28  29  30  31  32  33  34  35  37  38  39  40  42  43  44  45
  46  47  48  49  50  53  55  56  57  58  61  62  63  64  66  67  68  69
  70  71  72  73  74  75  76  77  78  79  80  81  82  84  85  86  87  88
  89  90  91  93  94  95  96  97 100 104 105 106 107 108 110 111 112 113
 114 116 117 118 119 120 122 123 124 125 126 127 129 130 131 133 135 136
 137 138 139 141 142 143 144 145 146 147 148 149 150 151 153 155 156 157
 158 159 160 161 162 163 164 165 166 168 169 170 171 172 173 174 175 176
 178 179 180 181 182 184 186 187 188 189 190 191 192 194 195 196 197 198
 199 200 201 202 203 204 207 209 210 211 212 213 214 216 217 219 221 223
 224 225 226 227 228 229 230 231 232 233 234 236 237 238 239 240 241 242
 243 244 245 246 247 248 249 250 253 254 255 256 257 258 259 260 261 262
 263 265 266 268 269 270 272 273 275 276 277 278 280 282 283 285 286 288
 289 290 291 292 293 295 296 297 299 301 303 304 305 307 309 310 311 312
 315 316 317 319 321 322 325 327 328 330 331 332 333 334 335 336 338 339
 340 343 344 345 346 347 348 351 352 353 354 355 356 358 359 360 361 362
 363 364 365 366 367 368 369 370 371 374 375 376 377 378 379 382 383 384
 385 387 388 389 392 393 394 395 396 397 398 399 402 403 404 405 406 408
 409 410 411 413 416 419 420 421 422 424 425 426 427 428 429 430 431 432
 433 434 436 438 440 443 444 445 446 447 448 449 450 451 452 453 454 456
 458 459 460 461 462 463 464 465 466 467 468 470 471 472 473 474 475 476
 477 478 479 480 481 482 483 484 486 487 488 490 491 492 493 494 495 496
 497 499 500 501 502 504 505 506 507 509 511 512 513 515 516 517 518 519
 520 521 522 523 524 525 526 527 528 529 530 531 533 534 535 537 539 540
 541 542 543 544 545 546 547 548 550 551 552 553 554 555 557 558 560 561
 562 563 564 565 566 569 571 573 574 575 577 578 579 580 581 583 584 585
 586 588 590 591 592 593 594 595 596 597 598 601 602 603 604 605 606 607
 608 609 610 612 614 617 618 619 620 621 623 624 625 627 628 629 630 631
 632 634 636 637 638 639 640 642 643 644 645 646 648 649 650 651 652 653
 654 655 656 657 658 659 660 661 662 664 665 666 667 668 669 670 671 673
 674 675 676 677 679 680 681 683 685 686 687 688 691 693 694 695 696 698
 699 700 701 702 704 705 706 707 708 710 711] [  6  13  16  18  24  25  26  27  36
41  51  52  54  59  60  65  83  92
  98  99 101 102 103 109 115 121 128 132 134 140 152 154 167 177 183 185
 193 205 206 208 215 218 220 222 235 251 252 264 267 271 274 279 281 284
 287 294 298 300 302 306 308 313 314 318 320 323 324 326 329 337 341 342
 349 350 357 372 373 380 381 386 390 391 400 401 407 412 414 415 417 418
 423 435 437 439 441 442 455 457 469 485 489 498 503 508 510 514 532 536
 538 549 556 559 567 568 570 572 576 582 587 589 599 600 611 613 615 616
 622 626 633 635 641 647 663 672 678 682 684 689 690 692 697 703 709]
[  2   3   6   7   8   9  10  11  13  14  15  16  17  18  19  22  24  25
  26  27  28  29  30  31  32  34  35  36  38  39  40  41  42  43  44  45
  46  47  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66
  67  68  69  70  72  73  74  75  77  78  80  81  83  84  85  88  89  91
  92  93  94  95  96  97  98  99 101 102 103 104 105 107 108 109 110 111
 112 113 115 116 118 120 121 122 123 124 125 127 128 129 130 131 132 133
 134 136 137 138 139 140 141 143 144 146 147 148 149 152 153 154 156 160
 161 162 163 165 166 167 168 169 170 172 173 174 175 176 177 178 179 180
 181 183 184 185 186 187 188 189 190 191 192 193 194 196 198 199 200 201
 202 203 204 205 206 208 209 210 211 212 213 215 217 218 219 220 222 225
 226 227 229 230 231 232 233 235 236 238 239 240 241 242 243 244 245 246
 247 248 249 250 251 252 253 255 256 257 258 260 261 262 263 264 265 266
 267 268 269 270 271 272 273 274 275 276 278 279 281 282 283 284 285 286
 287 288 289 291 293 294 296 297 298 300 301 302 305 306 307 308 309 312
 313 314 315 316 318 319 320 321 322 323 324 325 326 327 329 330 331 333
 334 335 336 337 338 340 341 342 343 344 345 346 347 348 349 350 351 352
```

```
 354 355 356 357 358 359 360 361 362 364 365 366 367 368 369 371 372 373
 375 376 377 378 380 381 382 383 385 386 387 388 389 390 391 392 393 394
 395 396 397 399 400 401 402 403 404 406 407 408 410 411 412 413 414 415
 416 417 418 419 423 425 427 429 432 433 434 435 436 437 439 441 442 443
 445 446 447 449 450 451 453 454 455 456 457 458 459 461 463 464 465 466
 468 469 470 471 474 475 476 478 479 482 485 486 487 488 489 490 491 492
 493 495 496 497 498 499 500 501 503 504 506 508 509 510 511 513 514 515
 516 517 518 519 520 521 522 525 526 528 529 530 531 532 533 534 536 537
 538 540 541 542 543 545 546 547 548 549 551 552 553 554 556 557 558 559
 560 561 562 563 566 567 568 569 570 571 572 573 575 576 577 579 581 582
 583 584 586 587 588 589 590 591 592 593 594 597 598 599 600 601 602 603
 605 606 608 609 610 611 613 614 615 616 617 618 620 621 622 623 624 625
 626 627 628 630 632 633 634 635 636 637 638 639 640 641 644 645 646 647
 648 650 651 654 655 656 657 658 659 661 663 665 666 667 668 669 670 671
 672 673 675 677 678 679 680 681 682 683 684 685 687 689 690 692 693 694
 695 697 698 700 703 705 707 708 709 710 711] [  0   1   4   5  12  20  21  23  33
37  48  49  50  71  76  79  82  86
  87  90 100 106 114 117 119 126 135 142 145 150 151 155 157 158 159 164
 171 182 195 197 207 214 216 221 223 224 228 234 237 254 259 277 280 290
 292 295 299 303 304 310 311 317 328 332 339 353 363 370 374 379 384 398
 405 409 420 421 422 424 426 428 430 431 438 440 444 448 452 460 462 467
 472 473 477 480 481 483 484 494 502 505 507 512 523 524 527 535 539 544
 550 555 564 565 574 578 580 585 595 596 604 607 612 619 629 631 642 643
 649 652 653 660 662 664 674 676 686 688 691 696 699 701 702 704 706]
[  0   1   2   3   4   5   6   7  10  11  12  13  14  15  16  17  18  20
  21  23  24  25  26  27  28  30  31  32  33  34  35  36  37  40  41  42
  46  48  49  50  51  52  53  54  55  56  58  59  60  61  62  64  65  66
  67  68  71  72  74  75  76  78  79  81  82  83  84  85  86  87  88  89
  90  92  93  94  95  96  98  99 100 101 102 103 104 105 106 107 108 109
 110 113 114 115 117 118 119 121 122 123 126 127 128 131 132 133 134 135
 137 138 140 142 143 144 145 146 147 149 150 151 152 153 154 155 157 158
 159 160 161 162 163 164 165 166 167 168 169 171 172 173 175 176 177 179
 181 182 183 184 185 186 187 188 189 191 193 195 196 197 199 200 201 204
 205 206 207 208 209 210 211 212 214 215 216 218 219 220 221 222 223 224
 225 226 228 229 230 232 233 234 235 236 237 238 239 241 242 243 244 245
 246 247 248 249 250 251 252 253 254 257 258 259 260 264 265 267 268 271
 272 273 274 275 277 279 280 281 282 283 284 286 287 289 290 292 293 294
 295 296 298 299 300 301 302 303 304 305 306 307 308 310 311 312 313 314
 315 316 317 318 320 321 323 324 325 326 328 329 330 331 332 337 339 340
 341 342 343 345 346 347 349 350 353 356 357 358 362 363 364 365 368 370
 371 372 373 374 375 376 378 379 380 381 382 383 384 385 386 387 388 389
 390 391 392 393 394 395 397 398 399 400 401 402 403 404 405 406 407 408
 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426
 427 428 429 430 431 432 435 436 437 438 439 440 441 442 443 444 445 446
 448 449 450 451 452 453 455 456 457 459 460 461 462 464 465 466 467 468
 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
 489 490 491 492 493 494 495 496 497 498 499 500 502 503 504 505 506 507
 508 510 511 512 514 515 516 519 520 521 522 523 524 526 527 529 530 531
 532 535 536 537 538 539 540 541 542 543 544 545 546 547 549 550 551 552
 553 555 556 557 558 559 560 564 565 567 568 570 572 573 574 575 576 577
 578 579 580 582 583 584 585 586 587 588 589 590 593 594 595 596 597 599
 600 604 605 607 608 609 611 612 613 614 615 616 617 619 622 623 624 625
 626 628 629 630 631 632 633 634 635 637 638 639 640 641 642 643 646 647
 649 650 651 652 653 654 657 659 660 661 662 663 664 665 666 667 669 670
 672 673 674 676 678 679 681 682 683 684 685 686 687 688 689 690 691 692
 696 697 699 701 702 703 704 705 706 707 709 710] [  8   9  19  22  29  38  39  43
```

```
44  45  47  57  63  69  70  73  77  80
  91  97 111 112 116 120 124 125 129 130 136 139 141 148 156 170 174 178
 180 190 192 194 198 202 203 213 217 227 231 240 255 256 261 262 263 266
 269 270 276 278 285 288 291 297 309 319 322 327 333 334 335 336 338 344
 348 351 352 354 355 359 360 361 366 367 369 377 396 433 434 447 454 458
 463 487 488 501 509 513 517 518 525 528 533 534 548 554 561 562 563 566
 569 571 581 591 592 598 601 602 603 606 610 618 620 621 627 636 644 645
 648 655 656 658 668 671 675 677 680 693 694 695 698 700 708 711]
[  0   1   4   5   6   7   8   9  10  11  12  13  14  15  16  18  19  20
  21  22  23  24  25  26  27  29  31  32  33  35  36  37  38  39  41  43
  44  45  46  47  48  49  50  51  52  53  54  56  57  59  60  61  63  64
  65  67  69  70  71  73  75  76  77  79  80  81  82  83  86  87  88  90
  91  92  94  96  97  98  99 100 101 102 103 106 108 109 110 111 112 113
 114 115 116 117 119 120 121 122 124 125 126 127 128 129 130 132 134 135
 136 138 139 140 141 142 143 145 148 150 151 152 154 155 156 157 158 159
 160 161 162 163 164 167 169 170 171 172 173 174 177 178 179 180 182 183
 184 185 186 189 190 191 192 193 194 195 196 197 198 200 201 202 203 205
 206 207 208 210 213 214 215 216 217 218 219 220 221 222 223 224 225 226
 227 228 231 232 233 234 235 236 237 240 241 242 245 246 247 248 250 251
 252 253 254 255 256 257 259 260 261 262 263 264 265 266 267 269 270 271
 272 273 274 276 277 278 279 280 281 282 283 284 285 287 288 289 290 291
 292 294 295 297 298 299 300 301 302 303 304 306 308 309 310 311 312 313
 314 316 317 318 319 320 322 323 324 326 327 328 329 331 332 333 334 335
 336 337 338 339 341 342 343 344 346 347 348 349 350 351 352 353 354 355
 357 359 360 361 362 363 364 365 366 367 369 370 372 373 374 375 376 377
 378 379 380 381 383 384 386 388 390 391 394 395 396 397 398 400 401 405
 406 407 409 410 411 412 413 414 415 416 417 418 420 421 422 423 424 425
 426 427 428 430 431 432 433 434 435 437 438 439 440 441 442 444 446 447
 448 451 452 454 455 456 457 458 460 461 462 463 465 467 469 472 473 474
 477 480 481 483 484 485 486 487 488 489 491 492 494 495 496 498 499 500
 501 502 503 504 505 507 508 509 510 511 512 513 514 515 516 517 518 522
 523 524 525 527 528 532 533 534 535 536 537 538 539 541 542 543 544 546
 548 549 550 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568
 569 570 571 572 573 574 576 578 580 581 582 585 586 587 588 589 591 592
 594 595 596 598 599 600 601 602 603 604 606 607 610 611 612 613 614 615
 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 633 634
 635 636 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653
 654 655 656 658 660 661 662 663 664 665 668 671 672 674 675 676 677 678
 679 680 682 683 684 685 686 688 689 690 691 692 693 694 695 696 697 698
 699 700 701 702 703 704 705 706 707 708 709 711] [  2   3  17  28  30  34  40  42
55  58  62  66  68  72  74  78  84  85
  89  93  95 104 105 107 118 123 131 133 137 144 146 147 149 153 165 166
 168 175 176 181 187 188 199 204 209 211 212 229 230 238 239 243 244 249
 258 268 275 286 293 296 305 307 315 321 325 330 340 345 356 358 368 371
 382 385 387 389 392 393 399 402 403 404 408 419 429 436 443 445 449 450
 453 459 464 466 468 470 471 475 476 478 479 482 490 493 497 506 519 520
 521 526 529 530 531 540 545 547 551 552 553 575 577 579 583 584 590 593
 597 605 608 609 632 637 657 659 666 667 669 670 673 681 687 710]
[  0   1   2   3   4   5   6   8   9  12  13  16  17  18  19  20  21  22
  23  24  25  26  27  28  29  30  33  34  36  37  38  39  40  41  42  43
  44  45  47  48  49  50  51  52  54  55  57  58  59  60  62  63  65  66
  68  69  70  71  72  73  74  76  77  78  79  80  82  83  84  85  86  87
  89  90  91  92  93  95  97  98  99 100 101 102 103 104 105 106 107 109
 111 112 114 115 116 117 118 119 120 121 123 124 125 126 128 129 130 131
 132 133 134 135 136 137 139 140 141 142 144 145 146 147 148 149 150 151
 152 153 154 155 156 157 158 159 164 165 166 167 168 170 171 174 175 176
```

```
 177 178 180 181 182 183 185 187 188 190 192 193 194 195 197 198 199 202
 203 204 205 206 207 208 209 211 212 213 214 215 216 217 218 220 221 222
 223 224 227 228 229 230 231 234 235 237 238 239 240 243 244 249 251 252
 254 255 256 258 259 261 262 263 264 266 267 268 269 270 271 274 275 276
 277 278 279 280 281 284 285 286 287 288 290 291 292 293 294 295 296 297
 298 299 300 302 303 304 305 306 307 308 309 310 311 313 314 315 317 318
 319 320 321 322 323 324 325 326 327 328 329 330 332 333 334 335 336 337
 338 339 340 341 342 344 345 348 349 350 351 352 353 354 355 356 357 358
 359 360 361 363 366 367 368 369 370 371 372 373 374 377 379 380 381 382
 384 385 386 387 389 390 391 392 393 396 398 399 400 401 402 403 404 405
 407 408 409 412 414 415 417 418 419 420 421 422 423 424 426 428 429 430
 431 433 434 435 436 437 438 439 440 441 442 443 444 445 447 448 449 450
 452 453 454 455 457 458 459 460 462 463 464 466 467 468 469 470 471 472
 473 475 476 477 478 479 480 481 482 483 484 485 487 488 489 490 493 494
 497 498 501 502 503 505 506 507 508 509 510 512 513 514 517 518 519 520
 521 523 524 525 526 527 528 529 530 531 532 533 534 535 536 538 539 540
 544 545 547 548 549 550 551 552 553 554 555 556 559 561 562 563 564 565
 566 567 568 569 570 571 572 574 575 576 577 578 579 580 581 582 583 584
 585 587 589 590 591 592 593 595 596 597 598 599 600 601 602 603 604 605
 606 607 608 609 610 611 612 613 615 616 618 619 620 621 622 626 627 629
 631 632 633 635 636 637 641 642 643 644 645 647 648 649 652 653 655 656
 657 658 659 660 662 663 664 666 667 668 669 670 671 672 673 674 675 676
 677 678 680 681 682 684 686 687 688 689 690 691 692 693 694 695 696 697
 698 699 700 701 702 703 704 706 708 709 710 711] [  7  10  11  14  15  31  32  35
46  53  56  61  64  67  75  81  88  94
  96 108 110 113 122 127 138 143 160 161 162 163 169 172 173 179 184 186
 189 191 196 200 201 210 219 225 226 232 233 236 241 242 245 246 247 248
 250 253 257 260 265 272 273 282 283 289 301 312 316 331 343 346 347 362
 364 365 375 376 378 383 388 394 395 397 406 410 411 413 416 425 427 432
 446 451 456 461 465 474 486 491 492 495 496 499 500 504 511 515 516 522
 537 541 542 543 546 557 558 560 573 586 588 594 614 617 623 624 625 628
 630 634 638 639 640 646 650 651 654 661 665 679 683 685 705 707]
```

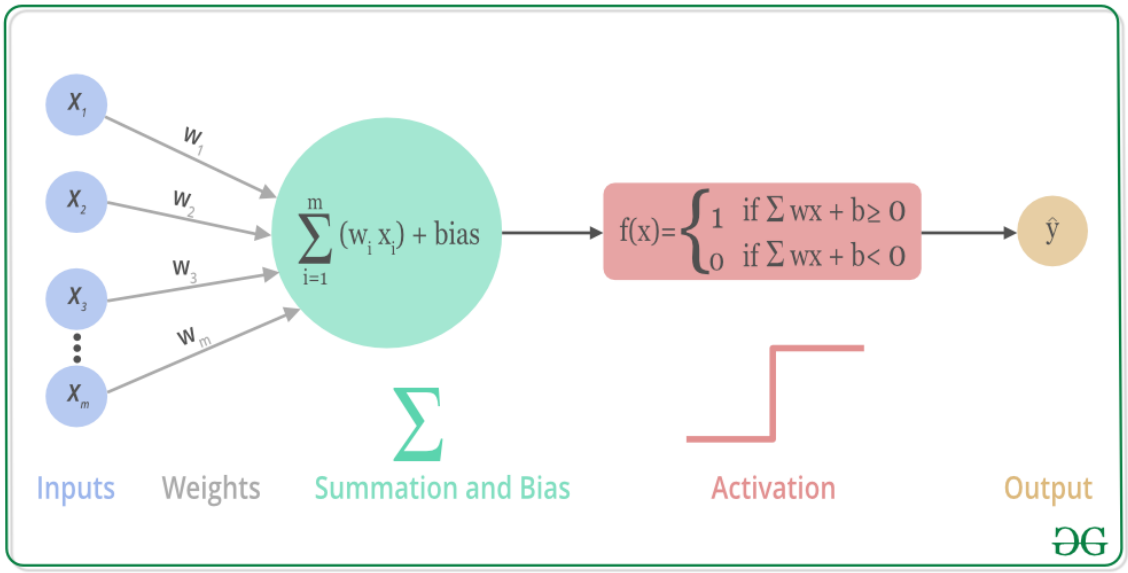
# 2. ANN




In [54]:
```
import matplotlib.pyplot as plt


def plot(func, name):
    x = np.linspace(-5, 5, 200)
    y = func(x)
    fig, ax = plt.subplots(1, 1, figsize=(4, 3))
    ax.plot(x, y)
    ax.grid(True)
    ax.set_title(name)
```

## Activation Function

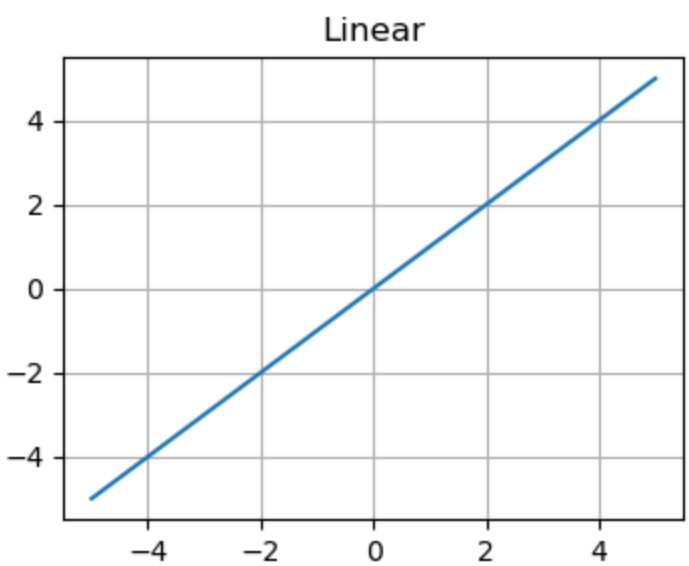



- Linear

$$z(x) = x$$

$$z'(x) = 1$$

In [55]:
```
plot(lambda x: x, "Linear")
```

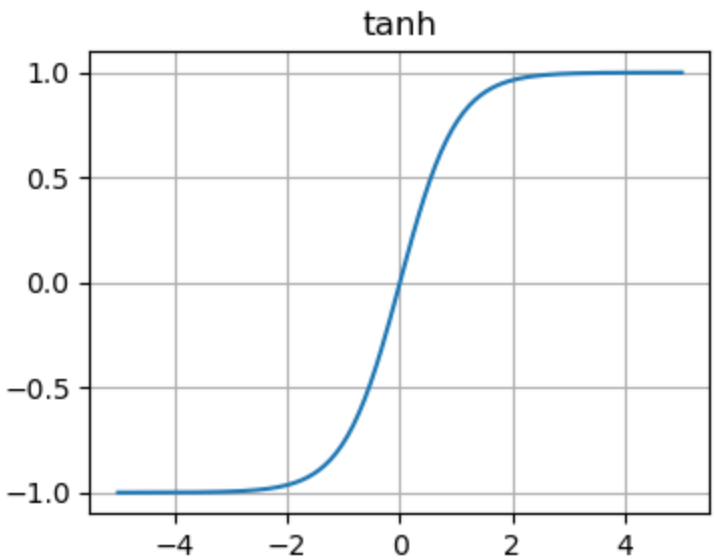



- tanh

$$z(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$z'(x) = 1 - \tanh^2 x$$

In [56]:
```
plot(np.tanh, "tanh")
```




- sigmoid

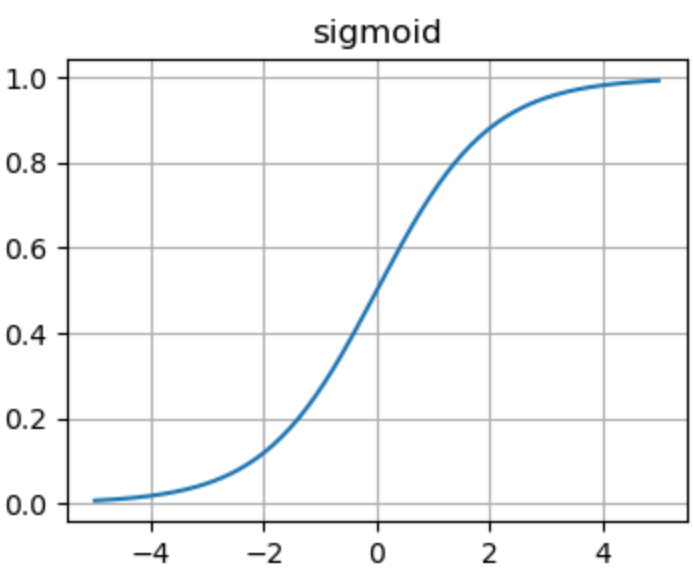
$$z(x) = \frac{1}{1+e^{-x}}$$

$$z'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = z(x)[1-z(x)]$$

*Hint for HW3*: You are going to use this in Logistic Regression

In [57]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

plot(sigmoid, "sigmoid")
```



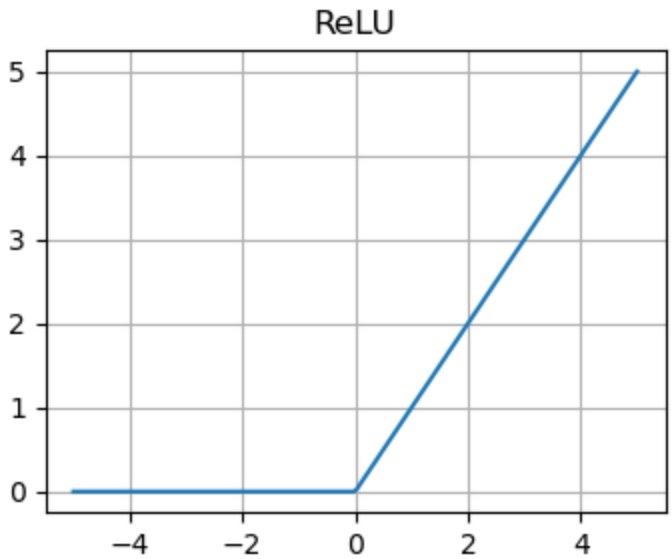

- ReLU

$$z(x) = \max(0, x)$$

$$z'(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

In [58]:
```
def relu(x):
    return x * (x > 0)

plot(relu, "ReLU")
```




- More: https://en.wikipedia.org/wiki/Activation_function

# 3. RDKit

## Introduction

RDKit is open-source toolkit for cheminformatics and it provides API in C++, Python, Java, C# and even JavaScript.

Funtionalities:

- 2D and 3D molecular operations
- Descriptor generation for machine learning
- ...

References:

- RDKit Website
- RDKit Python API

## Installation

`conda activate c142` (replace with your environment name)

`conda install rdkit -c conda-forge -y` (this may take some time)

## SMILES

- Reference
- A website for converting structures to SMILES
- A website for converting SMILES to structures

SMILES (**S**implified **M**olecular **I**nput **L**ine **E**ntry **S**ystem) is a line notation (a typographical method using printable characters) for entering and representing molecules and reactions.

### Examples:

- Methane: C
- Ethene: C=C
- Hydrogen cyanide: C#N
- Neopentane: C(C)(C)(C)C
- Cyclohexane: C1CCCCC1
- Benzene: c1ccccc1

### Basic Rules:

- Atoms are specified by its symbol with square brakets `[]` except for B, C, N, O, P, S, F, Cl, Br, I when they are normal valenced. **Hydrogens are implicitly represented.**
- Bonds are specified with "-" (single), "=" (double) or "#" (triple).
- Branches are specified by enclosing them in parentheses, and can be nested or stacked.
- Cyclic structures are represented by breaking one bond in each ring. The bonds are numbered in any order, designating ring opening (or ring closure) bonds by a digit immediately following the atomic symbol at each ring closure.
- Aromatic systems can be specified with lowercase characters or in Kekule form (in practice the latter may be preferred).
- ...

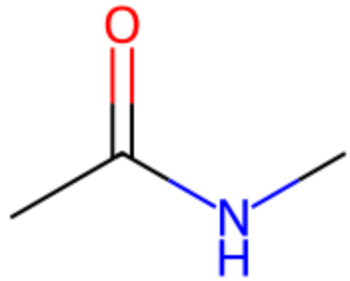
## Basic Usage

In [59]:
```
from rdkit import Chem
from rdkit.Chem import AllChem, Draw
```

Parse a SMILES string to a `Mol` object.

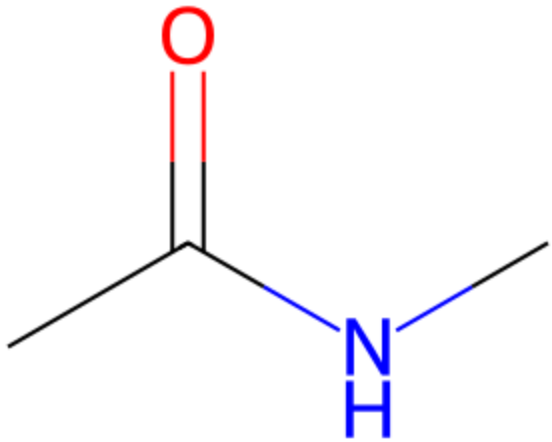
In [60]:
```
mol = Chem.MolFromSmiles("CC(=O)NC")
mol
```

Out[60]:




In [61]:
```
# if not displayed, use the following code
Draw.MolToImage(mol)
```

Out[61]:




Export a molecule to SMILES

In [62]:
```
Chem.MolToSmiles(mol)
```

Out[62]: 'CNC(C)=O'

Looping over atoms

In [63]:
```
for atom in mol.GetAtoms():
    print(type(atom), atom.GetIdx(), atom.GetSymbol())
```

```
<class 'rdkit.Chem.rdchem.Atom'> 0 C
<class 'rdkit.Chem.rdchem.Atom'> 1 C
<class 'rdkit.Chem.rdchem.Atom'> 2 O
<class 'rdkit.Chem.rdchem.Atom'> 3 N
<class 'rdkit.Chem.rdchem.Atom'> 4 C
```

Add hydrogens (make hydrogens explicitly)

In [64]:
```
mol_h = Chem.AddHs(mol)
mol_h
```

Out[64]:

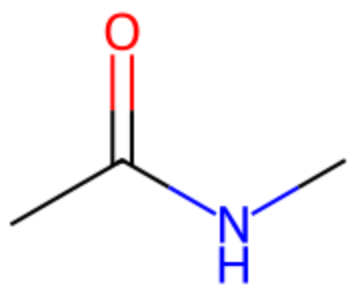

In [65]:
```
for atom in mol_h.GetAtoms():
    print(type(atom), atom.GetIdx(), atom.GetSymbol())
```

```
<class 'rdkit.Chem.rdchem.Atom'> 0 C
<class 'rdkit.Chem.rdchem.Atom'> 1 C
<class 'rdkit.Chem.rdchem.Atom'> 2 O
<class 'rdkit.Chem.rdchem.Atom'> 3 N
<class 'rdkit.Chem.rdchem.Atom'> 4 C
<class 'rdkit.Chem.rdchem.Atom'> 5 H
<class 'rdkit.Chem.rdchem.Atom'> 6 H
<class 'rdkit.Chem.rdchem.Atom'> 7 H
<class 'rdkit.Chem.rdchem.Atom'> 8 H
<class 'rdkit.Chem.rdchem.Atom'> 9 H
<class 'rdkit.Chem.rdchem.Atom'> 10 H
<class 'rdkit.Chem.rdchem.Atom'> 11 H
```

Delete hydrogens

In [66]:
```
mol_no_h = Chem.RemoveHs(mol)
mol_no_h
```

Out[66]:




Generate a 3D structure.

In [67]:
```
AllChem.EmbedMolecule(mol_h)
```

Out[67]: 0

Optimize the structure with MMFF94 force field

In [68]:
```
AllChem.MMFFOptimizeMolecule(mol_h)
```

Out[68]: 0

I/O with .mol or .sdf format

In [69]:
```
Chem.MolToMolFile(mol_h, "molecule.mol")
```

In [70]:
```
writer = Chem.SDWriter("molecule.sdf")
writer.write(mol_h)
writer.close()
```

In [71]:
```
mol_h = Chem.MolFromMolFile("molecule.mol", removeHs=False)
mol_h = Chem.SDMolSupplier("molecule.sdf", removeHs=False)[0]
```

In [ ]: