Diego Valdes

IST 707

Home Work 7

5/23/2019

**Introduction**

For as long as humans have been on the Earth, they have tried to use tools to improve the lifestyle of the species. It started with bone weapons for hunting, domesticating animals for labor and hunting, and then advances moved to more advanced materials and concepts. Bone weapons evolved to stone and then metal; economy changed from trading farm goods and livestock to using valuable metals, then coins, and promissory notes. The tools became more elaborate with time, allowing mankind to create cities, travel the world, and explore the heavens.

While these tools are impressive to most, in the 21st century, humans have moved to designing a different type of tool—a smart tool. Silicon is the new metal and the computer is the new dog. The computer is the tool that has opened immense possibilities and new avenues of creativity for the species. It is quite possibly the most adaptable tool ever designed. It has gotten more powerful as it has gotten smaller, it has evolved to do differing tasks, it is responsible for whole industries rising, it has changed the lives and interactions of humans, and it has made the impossible easy to accomplish. It can even mimic its creator.

Part of the machine learning to mimic humans is recognizing images. Image processing is useful for more advanced causes, like identifying cancerous cells, but it can also be used to find Waldo—clearly the most valuable use of the processing power of a computer. This ability while can be used for the most worthy endeavors, or the simplest, has practical everyday applications as well. Being able to identify human handwriting, with all of the intricacies that are present between each writer, can be as valuable to a corporation as the documents that could be processed at record speeds if done accurately.

**Analysis**

**About the Data**

The dataset originates from the Kaggle Digit Recognizer competition, with the goal of recognizing handwriting samples. The original datasets have 42,000 (train) and 28,000 (test) observations with 785 attributes. Each attribute represents a pixel on a 28 x 28 grid, where a number greater than 0 represents that pixel is part of the handwriting sample. Each pixel attribute is labeled ‘pixelNNN’ where NNN is the assigned number starting from 0 - 783 (fig 1.1).

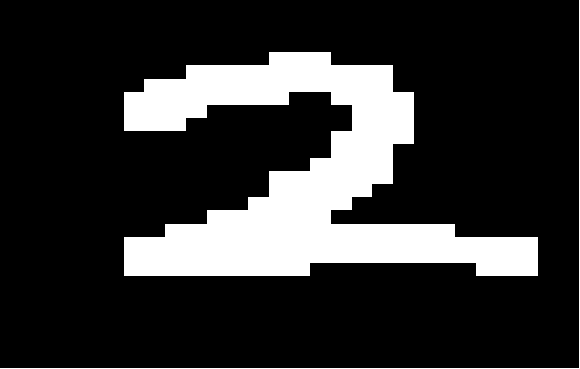
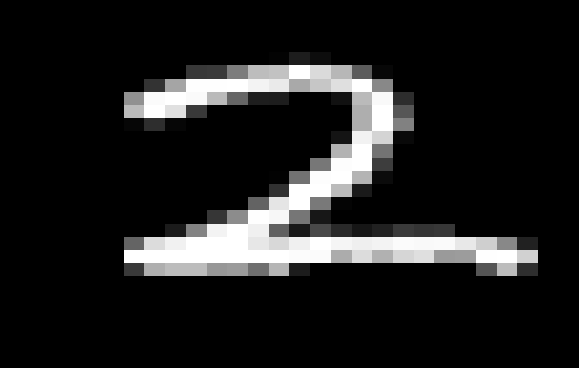
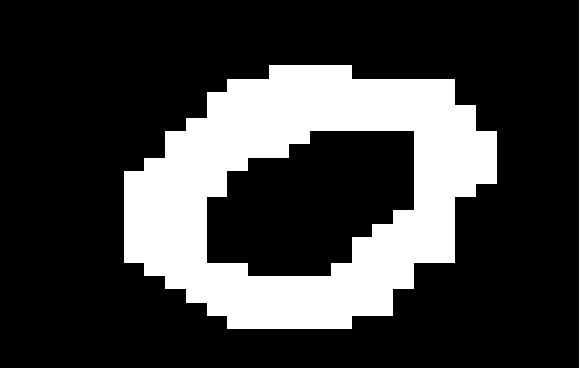
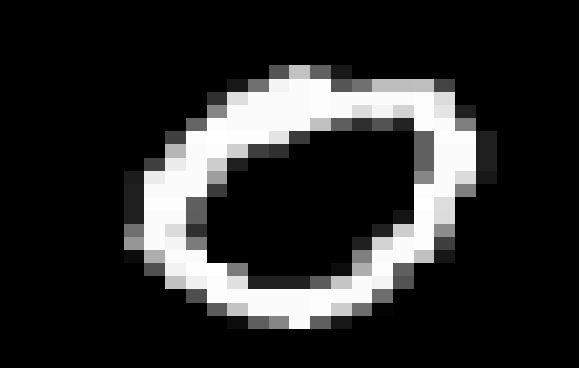
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |
| 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 |
| 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 |
| 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 |
| 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 |
| 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 |
| 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 |
| 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 |
| 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 |
| 392 | 393 | 394 | 395 | 396 | 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 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 |
| 448 | 449 | 450 | 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 |
| 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 | 501 | 502 | 503 |
| 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 | 526 | 527 | 528 | 529 | 530 | 531 |
| 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 |
| 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 | 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 |
| 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 | 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 |
| 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 | 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 |
| 644 | 645 | 646 | 647 | 648 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 656 | 657 | 658 | 659 | 660 | 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 670 | 671 |
| 672 | 673 | 674 | 675 | 676 | 677 | 678 | 679 | 680 | 681 | 682 | 683 | 684 | 685 | 686 | 687 | 688 | 689 | 690 | 691 | 692 | 693 | 694 | 695 | 696 | 697 | 698 | 699 |
| 700 | 701 | 702 | 703 | 704 | 705 | 706 | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | 717 | 718 | 719 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 |
| 728 | 729 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 750 | 751 | 752 | 753 | 754 | 755 |
| 756 | 757 | 758 | 759 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 780 | 781 | 782 | 783 |

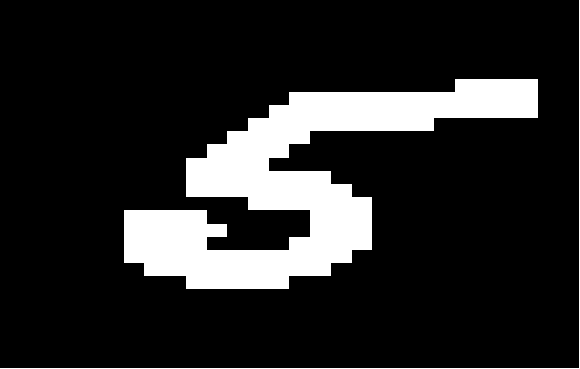
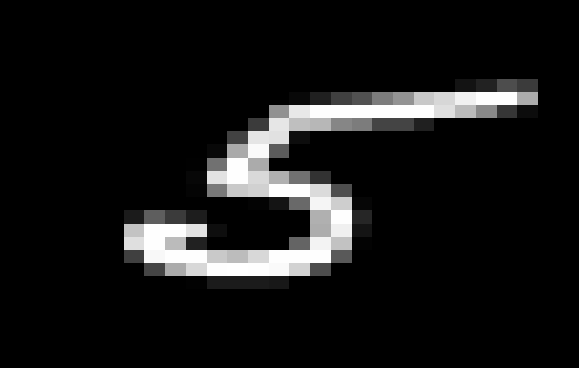
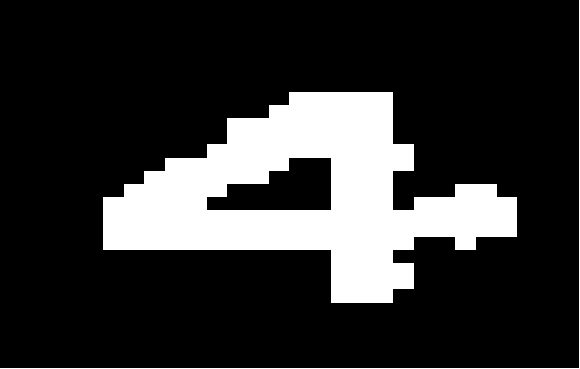
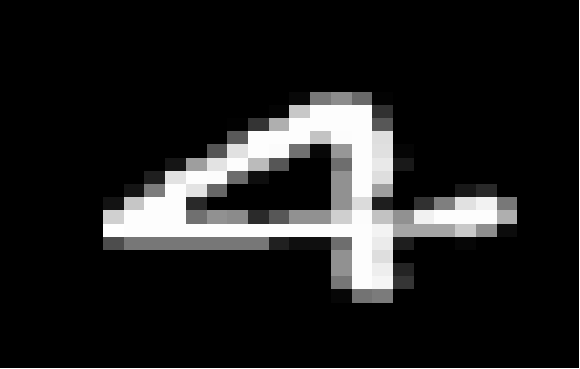
Fig 1.1: 28x28 matrix mapped with column numbers

The datasets used for the analysis consisted of 10% of the original data, sampled from each 10th element, ie: 10, 20, 30, … These sets had 1400 (train) and 1000 (test) observations (digit\_orig). The dataset had no missing values and required no cleaning. The first column, label, identifies the digit. This column was transformed into a factor. All other columns were left as numbers.

The range for pixels is 0 – 255. An alternative dataset (digit\_binary) was created where values above 0 where converted to 1. The original number range represents the color variation in the sample, but this was done to analyze whether a binary analysis would prove to provide a better result, as the goal of the analysis is to properly identify digits, not the various colors possible in creating the digit.

A sample of each digit (0 – 9) was rearranged into a 28x28 grid to plot the sample. This was done for both digit\_orig and digit\_binary datasets (fig 1.2). Both samples can be clearly identified as there respective digit, but the digit\_binary samples produce more defined samples.





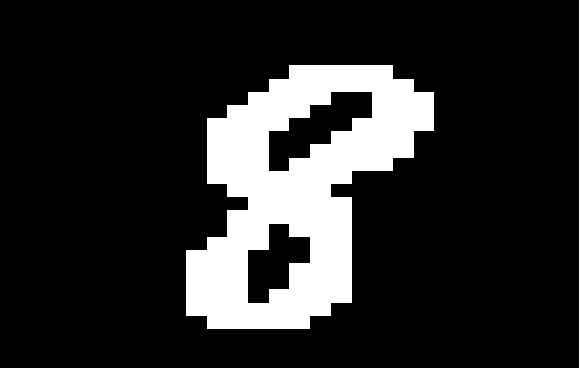
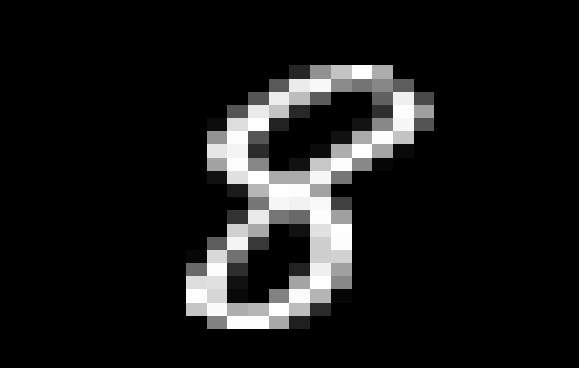
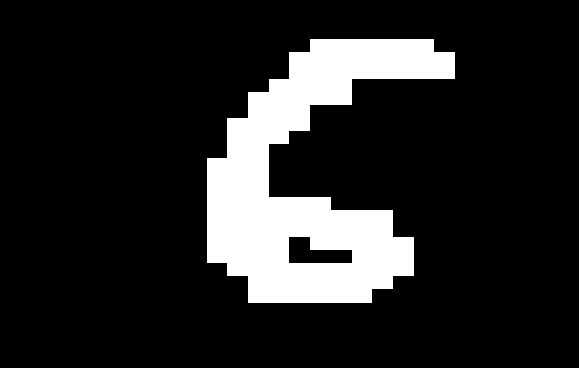
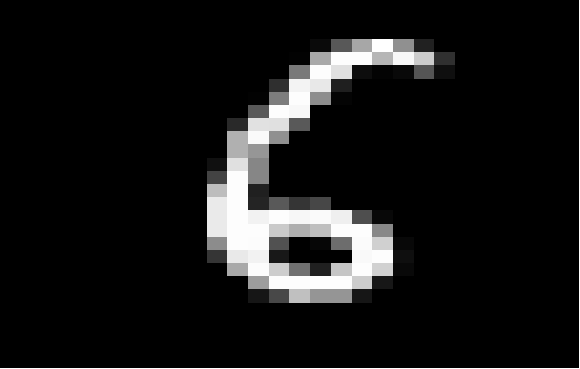


Fig 1.2: plots of digits (0,2,4,5,6,8) generated by transforming sample row to 28x28 matrix

The datasets were divided into a training sets with 70% of randomly sampled data, and test sets with the remaining 30%. There is also a test set where the label column is filled with ‘?’ as part of the Kaggle competition. This set is good for a submission but can’t be used for testing the accuracy of the model.

**Models**

In a previous analysis on both datasets using decision tree(rpart) and Naïve Bayes (e1071), there was statistical significance between the models that were generated with like datasets, ie: decision tree(digit\_orig) vs Naïve Bayes (digit\_orig), to determine that the decision tree was better suited to make the predictions. There was, however, only one example of a statistically significant difference when comparing the same model generated with differing datasets: Naïve Bayes(digit\_orig) vs Naïve Bayes(digit\_binary). Due to these findings, this analysis will focus on a comparing the performance of the different models using only the digit\_orig dataset.

**kNN (class::knn)**

The kNN model was created several times using different k values and then tested for accuracy. The value producing the best overall results was 5. An attempt to plot the model didn’t produce suitable results. There is too much dimensionality to make any sense of the visualization (fig 2.1).

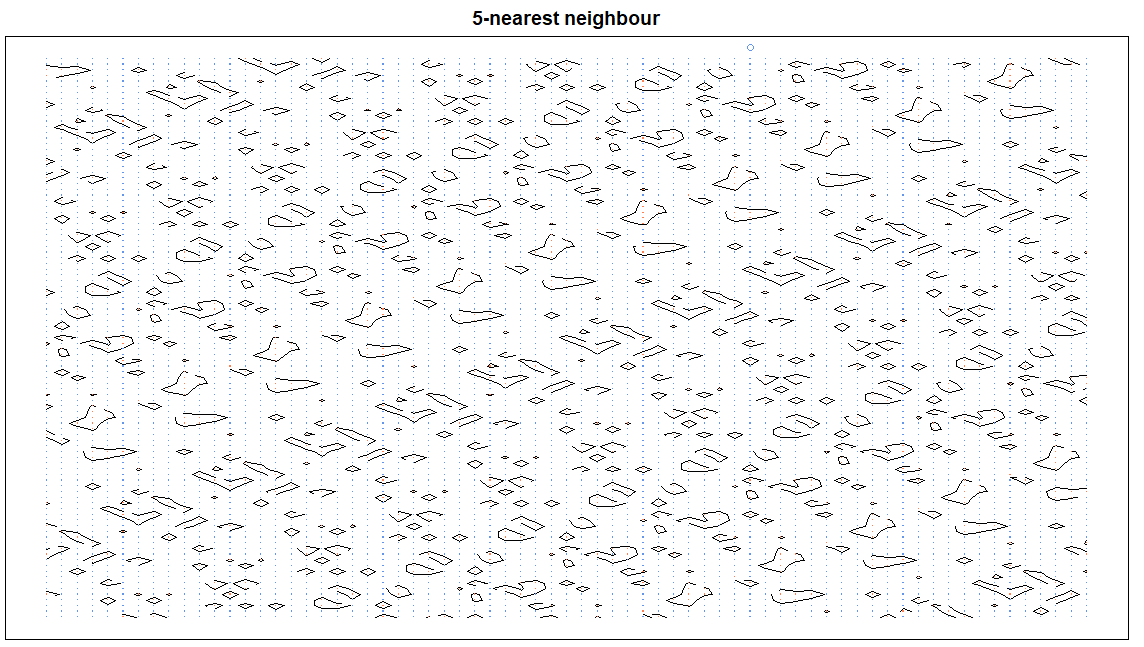


Fig 2.1: attempt at plotting kNN model

**SVM (e1071::svm)**

The SVM model underwent many variations with the possible variables. Both the kernels ‘polynomial’ and ‘linear’ provided the best results from other options, but ‘polynomial’ provided the highest accuracy and lowest support vectors between the two. Cost was adjusted to improve accuracy while altering kernel choices. This variable didn’t have any positive impact as it was raised. Finally it was left at .5.

As with kNN, attempting to visualize the model proved a challenge; none of them were able to provide a graphic due to the dimensionality of the data.

**Random Forest (randomForest::randomForest)**

The random forest model was initially generated and tested with default values and produced high accuracy results. Adjustments were made to trim the size of the trees beginning at 100 and increasing in increments of 50. The accuracy increased with each jump until reaching 500, the default value of trees.

**Results**



The models created in previous analysis (decision tree, Naïve Bayes) where included with new model results (kNN, SVM, Random Forest) for comparison. Part of the conclusion for previous models was that they didn’t seem to be suited to handle the digit data for prediction. This assessment was made based on how those models attain results. There is simply too much variation in the way a digit can be written, and there may not be enough consistency for those models to make accurate predictions.

The new models (kNN, SVM, Random Forest) don’t appear to suffer from those short comings. All three performed statistically better than the previous two, providing p-values below 0.05 significance and rejecting the null hypothesis that there was no difference between the results. This differs when comparing the new models to each other. In this instance (kNN vs SVM, kNN vs Random Forest, SVM vs Random Forest) the p-values for all where above the 0.05 significance level and didn’t allow for the rejection of the null hypothesis.

An interesting observation that becomes apparent when examining the confusion matrix for the models is that the digits share many of the same false positive and false negative predictions, as well as the same digits never being mistaken for another digit. An example of the latter is ‘3’, which has false positive and negative results for 5 and 8 and is never identified as 0, 1, 4, 6, 7 across all the models, (figs 3.1a -c).

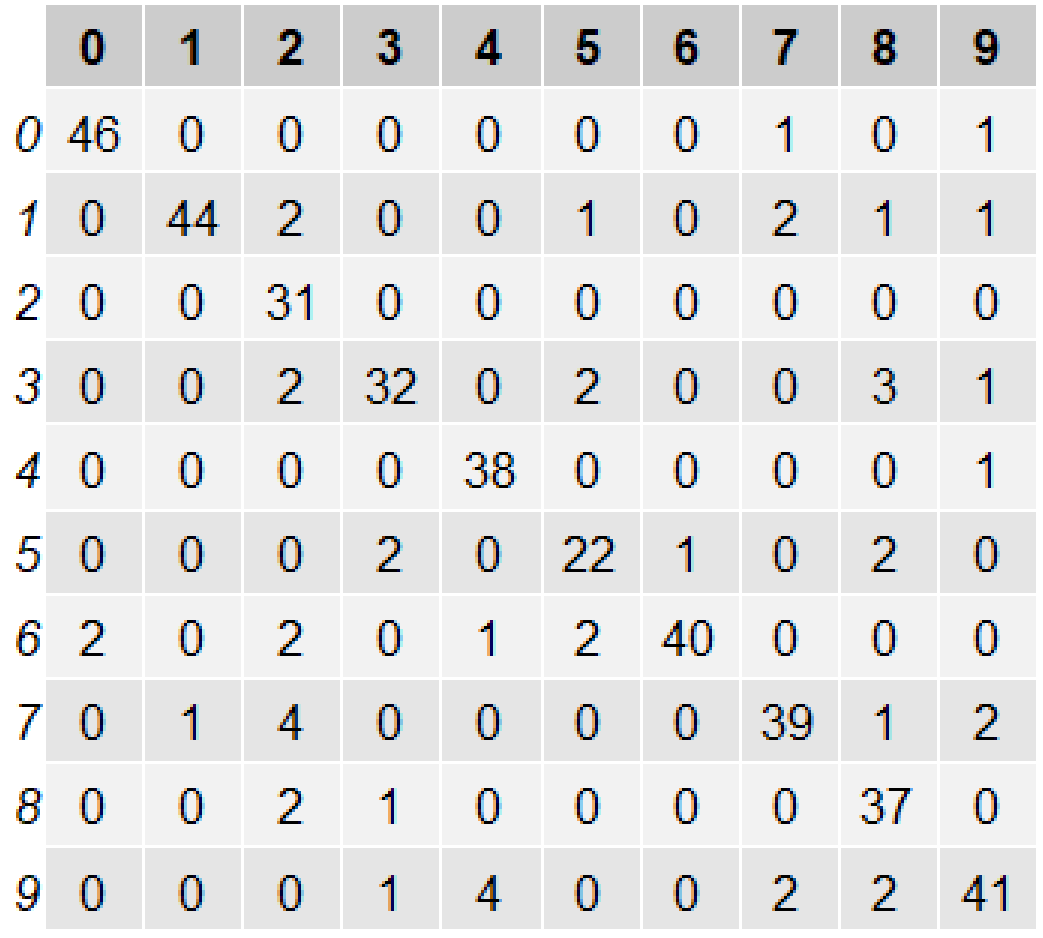
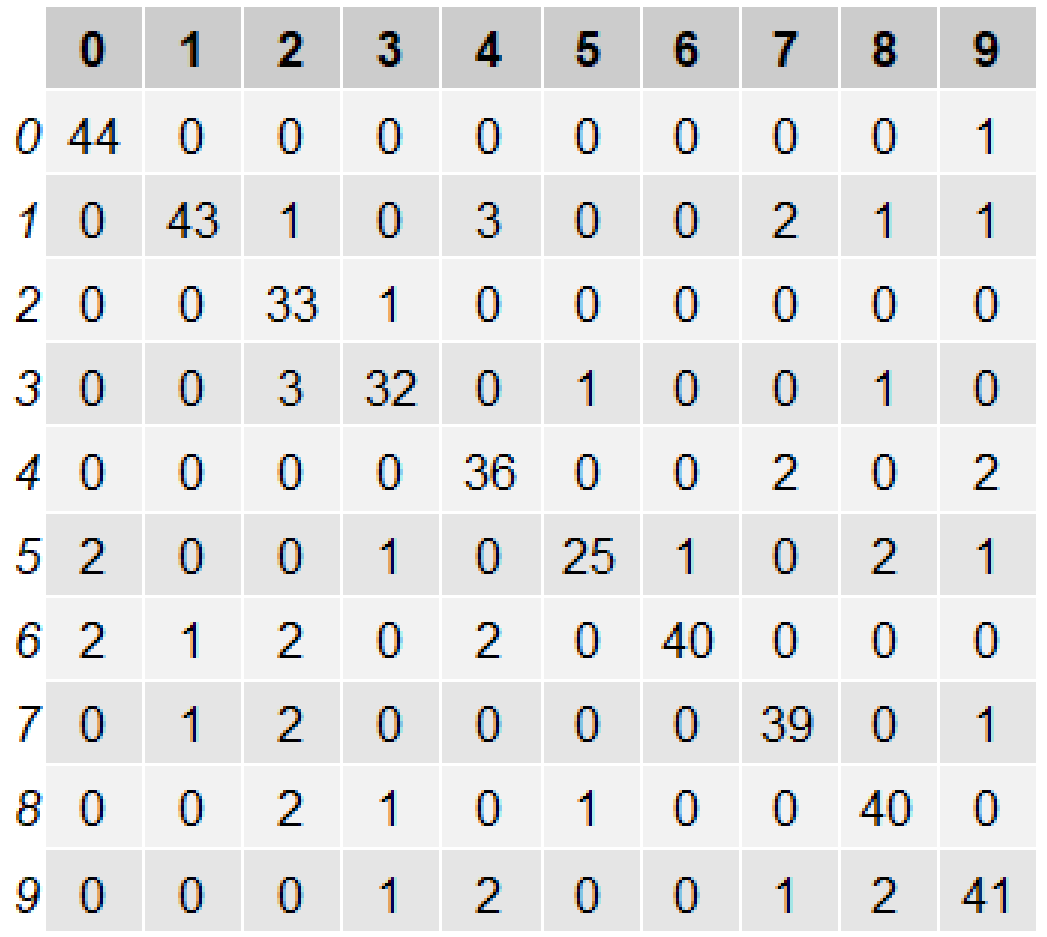
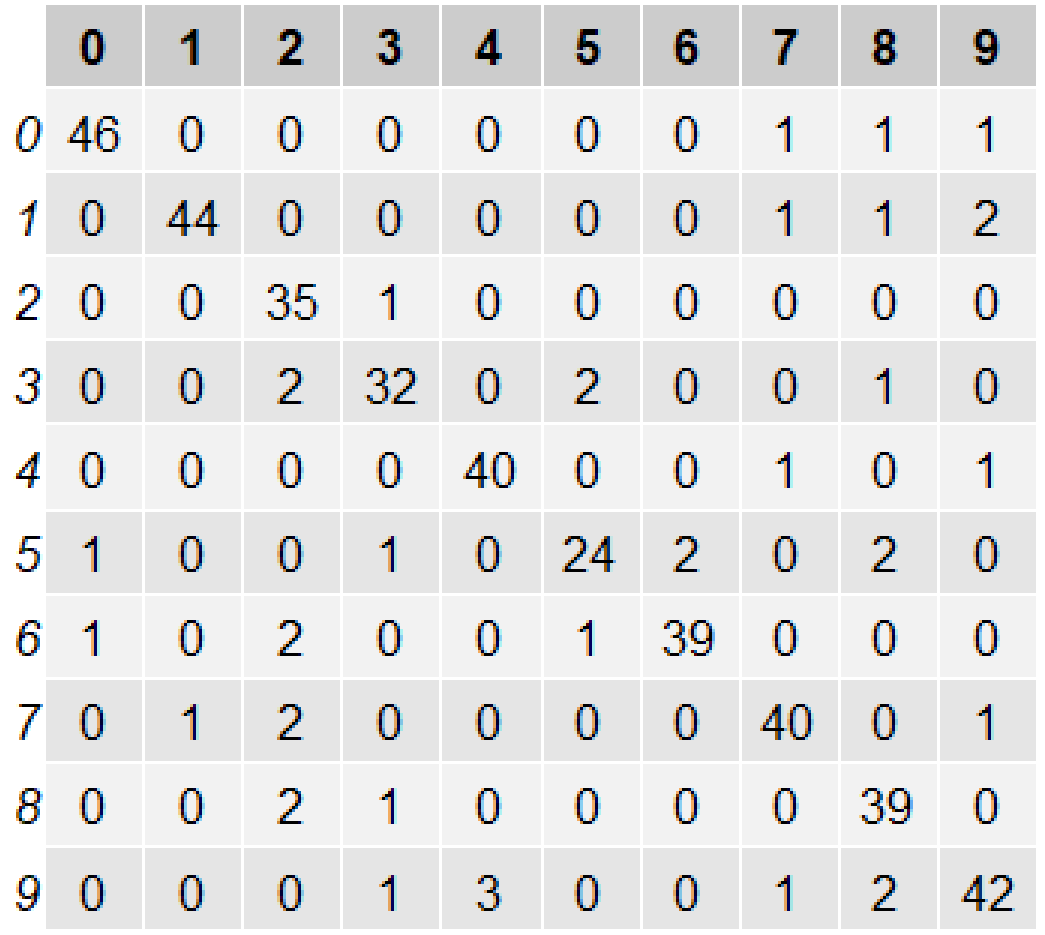
  

Fig 3.1a kNN Fig 3.1b SVM Fig 3.1c Random Forest

Knn svm random forest

**Conclusion**

The technology to be able to accurately read handwriting samples can be valuable to a multitude of organizations; especially those that have a collection of handwritten documents. Before, a company would need an army of employees just to read through and input the data into a system. Replacing this workforce with computers would be cost effective and increase the time to process documents.

Naïve bayes and decision tree were shown to not be the best models for this type of predictive analysis. The new models, SVM, kNN, Random Forest performed considerably better than those previous models. While all three models performed well, there was no significant difference between them to conclude that one should be chosen over the other. However, the model with the maximum potential higher accuracy is Random Forest. The nature of Random Forest, with each tree providing a vote seems to be able to discern the variance that can be found in handwriting samples and provide less confusion between similar digits: (1, 1, 1, 1, 1) vs. (7, 7, 7, 7, 7).

All three models also performed similarly across digits, having the same errors mistaking one digit for another. As an example, for each model the digit 3 can be mistaken for an 8 and vice versa. This may not only be a limitation of the algorithms, but the nature of handwriting identification. Many humans have most likely shared the experience of turning to a coworker and asking, “What number do you think this is? 4 or 9?” Just to guess and a few minutes later have them turn to you, “How about this one, 5 or 6?” Considering that the greatest computer ever designed can’t 100% accurately make these predictions, perhaps 88% and above is more than good enough.