

Digit Recognizer

-A data mining project

Presented By: Alreshidi Abdulaziz, Bader Albulayhis,

Sidi Mohamed Cheikh Ahmed, Yogita Singh

Outline

- Abstract
- Introduction
- Approach
- Data Source
- Data Attributes
- Tools used
- Data Preprocessing
- Data Mining Algorithms for classification
- Comparison
- Challenges Faced
- Applications
- Our performance result Vs Kaggle winner
- Things we would like to do differently if do it next time
- Conclusions
- References

Abstract

The purpose of this project is to apply various data mining techniques on around 73MB of data found on Kaggle.com and rightly classifying the pixel data of tens of thousands of images, of hand drawn digits to one of the 10 digits from 0 through 9. This project describes various techniques used to preprocess the images of handwritten digits' data and is focused on applying data mining algorithms like Decision Trees, Random Forest, KNN (K-Nearest Neighbors), Naïve Bayes etc. on the same. It also tries to figure out the best algorithm for our classification problem based on the model's performance.

Introduction

Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. Digit recognition is an art of classifying and identifying digit from input image and then it is converted into ASCII or other equivalent machine editable form so that it can be processed. This project focusses on using various data mining algorithms to rightly classify the hand drawn images from the pixel data and then on applying various techniques to improve our models' performance. The algorithms are written in python.

Approach:

We tried to incorporate most of the data mining algorithms namely Decision Trees, Random Forest, KNN (K-Nearest Neighbors), Naïve Bayes, Neural Networks etc., we studied in class to put everything to practice. As the data was pretty balanced for all the 10 labels, we could safely use accuracy as an indicator of a model's performance. Thus, we built various models and then

tried to improve the accuracy of each. At last we compared them based on their performance and selected the one with the maximum accuracy as the best model suggested for our problem.

Data Source:

The data for this project was taken from Kaggle website using the MNIST (Modified National Institute of Standards and Technology) dataset. Following is the link to the dataset:

<https://www.kaggle.com/c/digit-recognizer/data>

Data Attributes:

The data is about images of hand-drawn images of digits from 0 through 9. Each image is vectorized with 28 pixels in height and 28 pixels in width for a total of 784 pixels. Each pixel has a single pixel value associated with it ranging from 0 and 255, inclusive. The dataset consists of a csv file having an extra column for the target label. The rest of the 784 columns contain the pixel values of the associated image of the hand drawn digit. Below is a glimpse of our data:

```
df = pd.read_csv('train.csv')
df.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pix
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

Tools used

We were a team of 4 and chose to write our algorithms in Python. However, each one of us had a preference for an IDE they were comfortable with. Following is the detailed description of the tools that our team used collectively and the libraries we used to process, analyze and visualize the data:

➤ Coordination:

For team meetings and document sharing we used the following tools:

- Google Docs: To share the documents, code and work in progress
The reason we chose Google Docs for this project for version control, was because this tool is easy to use, free and all of us had this already.

Source: One can find it under google apps on one's machine. You just need to have a Gmail account to use it.

- Google Hangout: For weekly meetings and decide the plan of action

The reason we chose Google Docs for this project for our virtual project meeting, was because it gives you the option of video/audio chat along with screenshare. Moreover, it's easy to use, free and all of us had this already.

Source: One can find it under google apps on one's machine. You just need to have a Gmail account to use it.

➤ IDEs/distribution:

- Anaconda

Anaconda is a python distribution which makes installing python quite easy. It also comes with a lot of packages/libraries already installed and some great IDEs like Jupyter Notebook, Spyder etc.

The reason we use it was because of its easy installation of python and its packages for data mining.

Source: You could download it from:

<https://anaconda.org/>

- Jupyter Notebook

Source: If you use Anaconda, it comes installed

- Spyder

Source: If you use Anaconda, it comes installed

- PyCharm

Source: You could download it from:

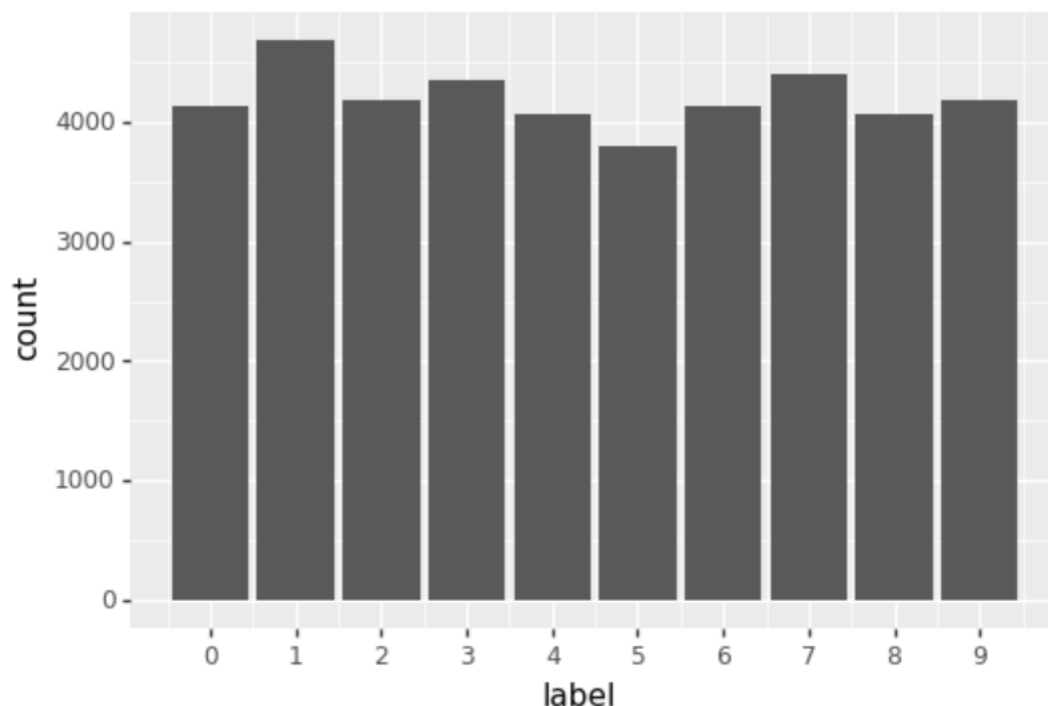
<https://www.jetbrains.com/pycharm/download/#section=windows>

➤ Libraries Used

Libraries Used	Purpose
Pandas	Pandas is a package used for data processing in python that offers data frame objects for data manipulation and analysis.
Numpy	Numpy is a python package for large multidimensional arrays and matrices.
Matplotlib	Matplotlib is a 2D plot python library which produces beautiful figures and plots for data visualization
Sklearn	Scikit-learn is a python library for data mining and analysis. It features various data mining and machine learning algorithms and works well with other python libraries such as Scipy, Numpy etc.

Data Preprocessing

After a careful look at the data, we thought of preparing our data for further processing. For this we thought of figuring out if there was any missing data. To our surprise there was none. Then we tried to figure out the distribution of the data for the ten classes namely digits 0 through 9. Again, it turned out to be pretty balanced. Thus, we could safely use accuracy to measure model's performance. Below is the data distribution plot for all 10 digits:



Data Mining Algorithms for classification:

We wanted to try out different data mining techniques (namely, classification, clustering, dimensionality reduction) and their algorithm to learn and see which model worked best with the problem at hand. We started by creating different models and then working on them individually to improve each one's performance. After then we compared the models based on the best performance. Below is a detailed account of all the models and the results produced:

1. Decision Tree

Decision tree is a non-parametric supervised learning method used for classification. It tries to learn from different decision rules inferred from the data attributes (independent variable) and predicts the value of a target variable (dependent variable).

We built the decision tree using following steps:

- Normalization of data: Generally, we need to normalize the input data to get the value range between 0 and 1. However, as we had the pixel data with each image having 784 pixels, with 0 and 1 representing the pixels ranging from 0 to 255. Moreover, we do so to eliminate different units to compare between them and in our case all the columns represented the pixel values. Thus, we didn't need to normalize the data.
- Splitting the data into train and test: We split the data into train and test data.

- Assumptions:

Below are the main assumptions we make while constructing a decision tree:

1. In the beginning, all the attributes in the data are considered as the root.
2. The labels are considered categorical, if not, they are discretized.
3. The most informative attribute becomes the root.

- Weakness of the assumptions to our problem:

With just the labels it did not give us an overall picture and not the probability of the prediction. Also, it can easily overfit.

- Below are the performance characteristics of this model:

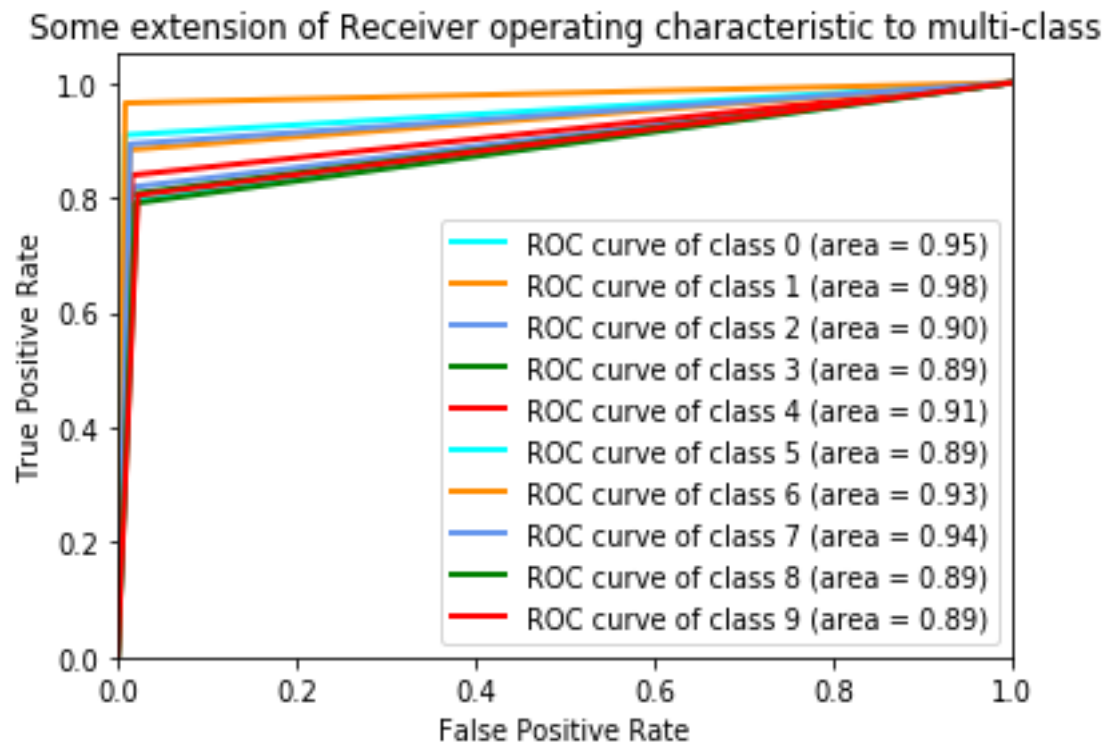
Accuracy :85.36%

Confusion Matrix:

	0	1	2	3	4	5	6	7	8	9
0	739	2	9	8	4	16	14	1	7	13
1	0	927	7	9	5	2	4	1	6	0
2	11	9	704	33	12	13	14	25	26	13
3	8	9	33	697	9	41	5	21	25	15
4	2	6	10	5	694	11	23	12	21	43
5	6	9	10	40	9	601	22	5	25	29
6	14	4	15	3	19	22	743	3	11	7
7	6	5	19	14	7	7	4	803	9	25
8	4	16	17	29	15	30	15	9	607	26
9	11	3	14	25	43	13	1	29	20	653

	precision	recall	f1-score	support
0	0.92	0.91	0.92	813
1	0.94	0.96	0.95	961
2	0.84	0.82	0.83	860
3	0.81	0.81	0.81	863
4	0.85	0.84	0.84	827
5	0.79	0.79	0.79	756
6	0.88	0.88	0.88	841
7	0.88	0.89	0.89	899
8	0.80	0.79	0.80	768
9	0.79	0.80	0.80	812
avg / total	0.85	0.85	0.85	8400

The below is the ROC curve :



Features Important:

1. feature 434 (0.060050)
2. feature 409 (0.044703)
3. feature 155 (0.043825)
4. feature 657 (0.040340)
5. feature 239 (0.039635)
6. feature 431 (0.033198)
7. feature 488 (0.031044)
8. feature 375 (0.029751)
9. feature 319 (0.028281)
10. feature 485 (0.025171)
11. feature 377 (0.019316)
12. feature 96 (0.017617)
13. feature 550 (0.017584)

- 14. feature 455 (0.016383)
- 15. feature 290 (0.014592)
- 16. feature 569 (0.014583)
- 17. feature 351 (0.013280)
- 18. feature 322 (0.011691)
- 19. feature 183 (0.011451)
- 20. feature 270 (0.010409)

2. Naïve Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features

Assumptions:

Conditional independence assumption: To reduce number of parameters we assume that attribute values are independent of each other, given the class.

Weakness of the assumptions to our problem:

Conditional independence doesn't hold for our problem. The attributes which are various pixels are conditionally dependent on each other. The position of a pixel by itself does not carry information about the class.

However, Naïve Bayes performs well despite the conditional independence assumption. It is so because even though the probability estimates of Naïve Bayes are of low quality, its classification decision are good surprisingly. This is so because the classification is based on which class gets the highest score and not on how accurate the estimates are.

2.2) Multinomial Naïve Bayes

The multinomial Naive Bayes classifier is suitable for classification with discrete features.

Below are the performance characteristics of this model:

Accuracy: 82.55%

Classification report:

	precision	recall	f1-score	support
0	0.92	0.91	0.91	2064
1	0.90	0.93	0.91	2355
2	0.89	0.83	0.86	2132
3	0.78	0.80	0.79	2191
4	0.83	0.75	0.79	2026
5	0.84	0.65	0.73	1894
6	0.87	0.93	0.90	2077
7	0.95	0.83	0.88	2191
8	0.64	0.77	0.70	1991
9	0.70	0.82	0.75	2079
avg / total	0.83	0.83	0.83	21000

3. KNN

KNN i.e. K Nearest Neighbors is a non-parametric algorithm used for classification, where a case or a data point is classified by a majority vote of its neighbors which in turn is based on a similarity measure (e.g. distance function)

Below are the main assumptions we make while building a KNN algorithm:

- It performs much better if all of the data have the same scale
- It works well with a small number of input variables, but struggles when the number of inputs is very large
- It makes no assumptions about the functional form of the problem being solved

Weakness of the assumptions to our problem:

KNN takes a lot of time when the training data (input variable) is large as for every test data, the distance should be computed between test data and all of the training data. Thus, a lot of time is needed. As our data had 784-pixel values, it took a lot of computation time.

Accuracy: 96.0%

```
confusion_matrix:
[[2045   1    3    2    0    4    8    0    0    1]
 [   0 2330   8    2    2    1    4    5    2    1]
 [   19   18 2023   13    2    2    2   41    8    4]
 [   4    2   13 2082    0   38    3   15   24   10]
 [   1   19    0    0 1927    0    7    4    1   67]
 [   5    1    0   39    4 1789   36    1    4   15]
 [  11    3    1    0    3    8 2051    0    0    0]
 [   1   19   10    1    4    0    0 2129    0   27]
 [   7   19   10   45   11   37   10    5 1824   23]
 [   6    2    1   13   34    8    1   52    3 1959]]
```

Classification report:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	2064
1	0.97	0.99	0.98	2355
2	0.98	0.95	0.96	2132
3	0.95	0.95	0.95	2191
4	0.97	0.95	0.96	2026
5	0.95	0.94	0.95	1894
6	0.97	0.99	0.98	2077
7	0.95	0.97	0.96	2191
8	0.98	0.92	0.95	1991
9	0.93	0.94	0.94	2079
avg / total	0.96	0.96	0.96	21000

Plot of actual labels versus their predicted labels:



4. Random Forest

Random forest is an ensemble learning method for classification, which works by constructing multiple decision trees using training data and predict the class that is the mode of the classifications done by all the decision trees individually. It corrects for the decision tree's habit of overfitting.

Below are the main assumptions we make while building a KNN algorithm:

- At each step of building individual tree, we find the best split of data

- While building a tree we use not the whole dataset, but bootstrap sample and that sample is representative

Weakness of the assumptions to our problem:

- Whenever we increase the number of estimator (Number of trees in forest), the model become more complex and doesn't significantly affect the accuracy. In addition, it takes more time to create the model.

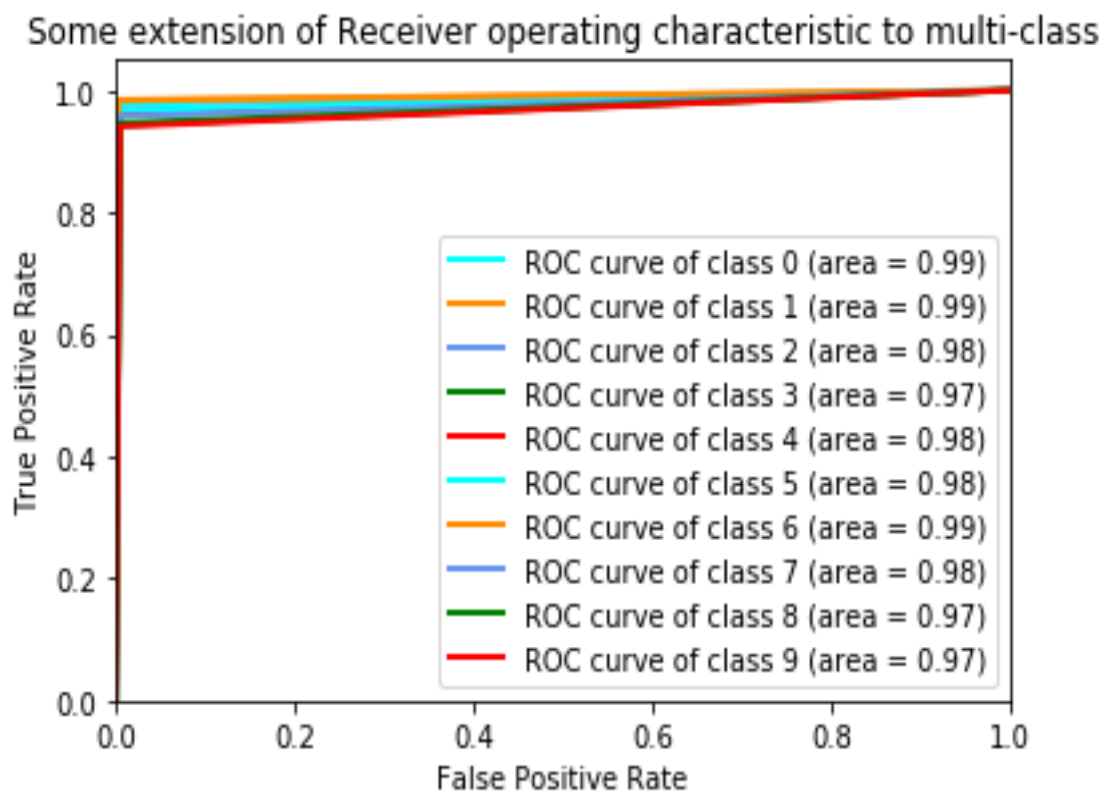
Accuracy: 96.62%

	0	1	2	3	4	5	6	7	8	9
0	795	0	0	1	2	2	6	0	7	0
1	0	945	7	3	2	1	0	1	2	0
2	3	0	832	2	4	1	4	6	6	2
3	4	2	17	814	0	6	1	8	8	3
4	2	3	1	0	794	0	2	2	5	18
5	3	1	1	9	0	731	5	1	3	2
6	2	1	0	0	1	6	828	0	3	0
7	2	3	14	1	5	0	0	861	3	10
8	2	4	0	12	1	11	7	1	727	3
9	10	1	2	9	12	0	0	10	4	764

The below is classification report :

	precision	recall	f1-score	support
0	0.97	0.98	0.97	813
1	0.98	0.98	0.98	961
2	0.95	0.97	0.96	860
3	0.96	0.94	0.95	863
4	0.97	0.96	0.96	827
5	0.96	0.97	0.97	756
6	0.97	0.98	0.98	841
7	0.97	0.96	0.96	899
8	0.95	0.95	0.95	768
9	0.95	0.94	0.95	812
avg / total	0.96	0.96	0.96	8400

Roc curve:



Features Important:

1. feature 378 (0.010993)
2. feature 409 (0.010052)
3. feature 350 (0.009852)
4. feature 461 (0.008025)
5. feature 375 (0.007983)
6. feature 433 (0.007618)
7. feature 487 (0.007218)
8. feature 210 (0.007122)
9. feature 406 (0.006999)
10. feature 488 (0.006800)
11. feature 489 (0.006711)
12. feature 291 (0.006691)
13. feature 542 (0.006636)
14. feature 405 (0.006342)
15. feature 318 (0.006330)
16. feature 154 (0.006282)
17. feature 515 (0.006182)
18. feature 569 (0.006090)
19. feature 239 (0.006077)
20. feature 211 (0.006042)

- **Experiment of imbalance data with Random forest**

We have experience the scenario that might occurred which is imbalance data. We manipulate the data by almost removed all observations of class 9 we only kept 100 observations.

We train the model with normal data and only 80 observations of class 9, and we test the model by 20 observations of class 9.

The result are shown below:

- R square: 96%
- Confusion matrix:

As you see in confusion matrix, zero correct for class 9

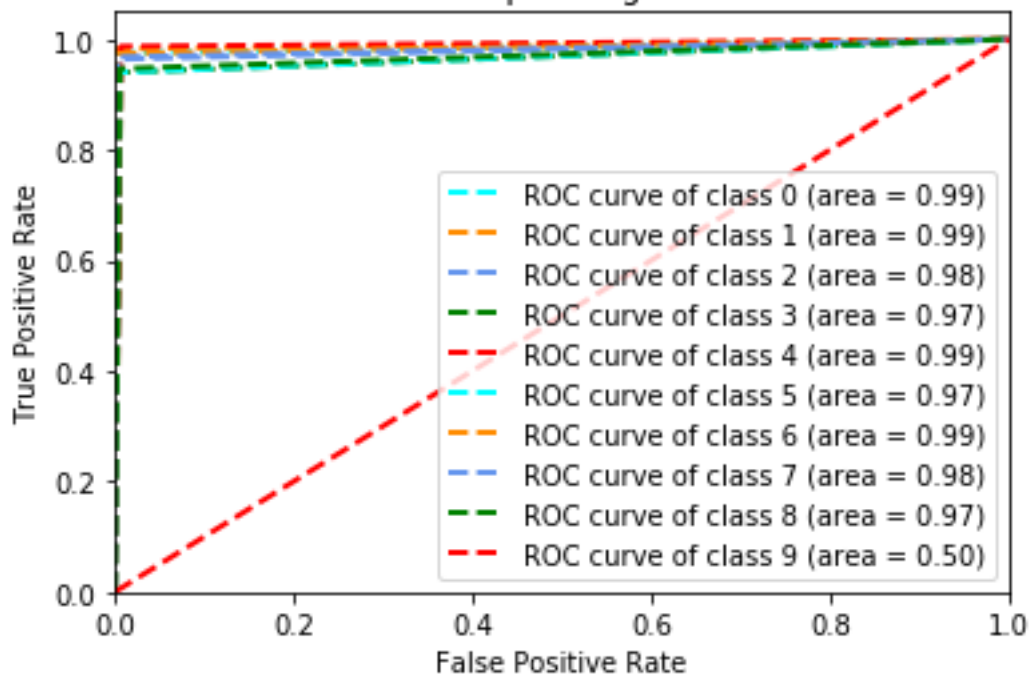
	0	1	2	3	4	5	6	7	8	9
0	804	0	3	3	1	2	5	0	6	0
1	0	924	5	4	4	1	0	1	1	0
2	3	2	810	5	5	0	3	10	2	0
3	0	1	19	801	0	10	1	7	13	0
4	0	0	0	0	766	0	3	2	5	0
5	5	3	0	20	4	756	7	2	7	0
6	4	1	1	0	3	7	801	0	5	0
7	1	6	11	0	9	1	0	850	0	0
8	3	3	4	13	6	8	6	1	783	0
9	0	0	0	0	15	1	0	4	0	0

Classification report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	824
1	0.98	0.98	0.98	940
2	0.95	0.96	0.96	840
3	0.95	0.94	0.94	852
4	0.94	0.99	0.96	776
5	0.96	0.94	0.95	804
6	0.97	0.97	0.97	822
7	0.97	0.97	0.97	878
8	0.95	0.95	0.95	827
9	0.00	0.00	0.00	20
avg / total	0.96	0.96	0.96	7583

ROC:

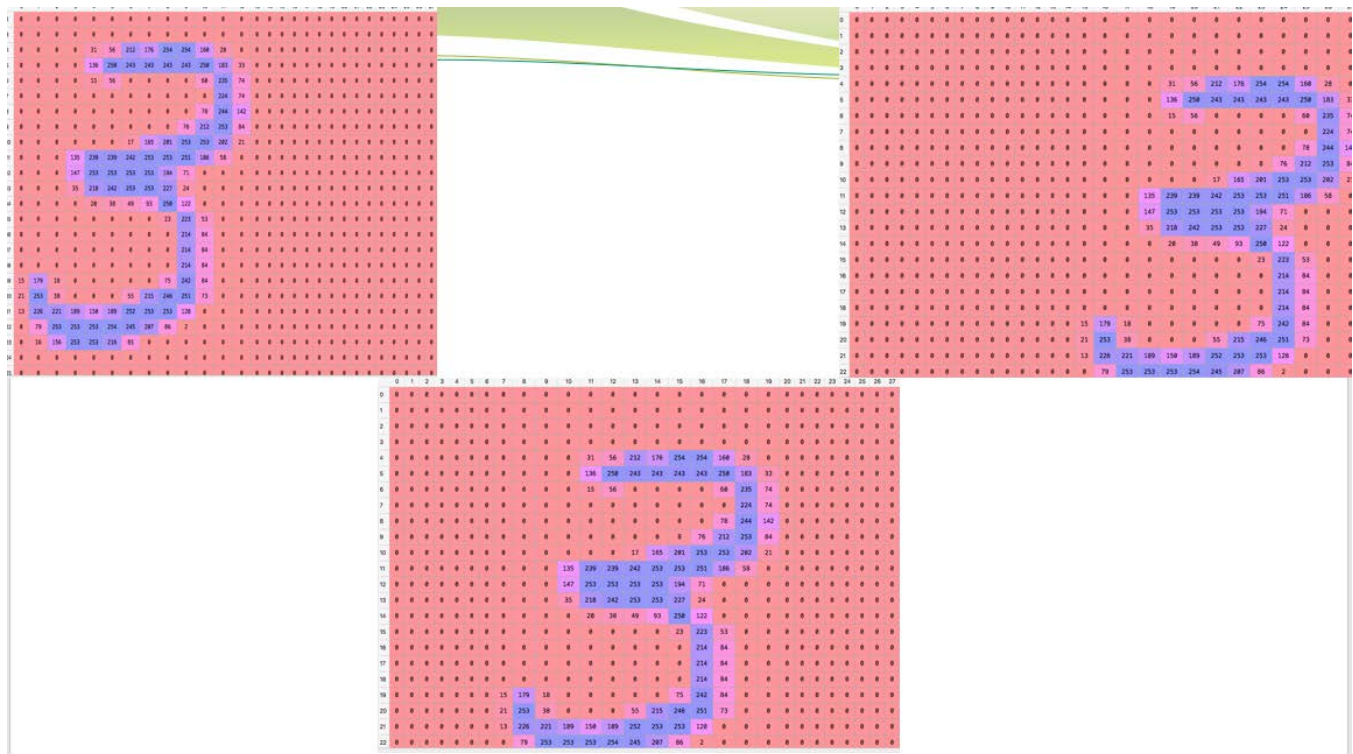
Some extension of Receiver operating characteristic to multi-class



Data augmentation

The data augmentation is a way to reduce the overfitting problem by manipulating the Images by shifting right, shifting left, zoom in, zoom out, flip and rotate. We initially shift all images by right but this approach doesn't

increase the accuracy. In fact, it slightly decrease the accuracy (93%). Then, we shift all the images to right and the result increased to 96%. So, we believe the result can be improved if we apply more methods such as zoom in and zoom out etc. Below are examples of images:



5. Multilayer perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Following are the advantages of Multi-layer Perceptron:

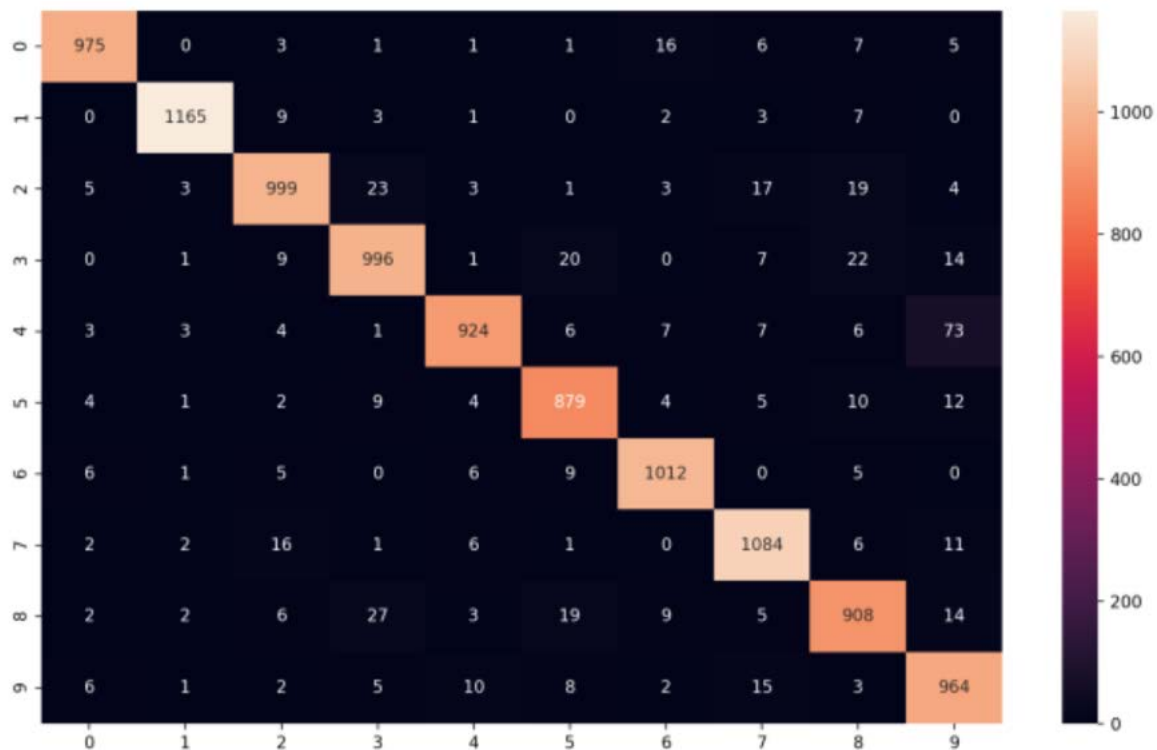
- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning) using partial fit.

Following are the disadvantages of Multi-layer Perceptron (MLP):

MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore, different random weight initializations can lead to different validation accuracy. MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.

Accuracy: 94%

Confusion Matrix:



Comparison of different algorithms:

Algorithm	Accuracy (%)
Decision Tree	85
Naïve Bayes	82.55
KNN	96.0
Random Forest	96
MLP	94

After looking and comparing the algorithms' performances we found that Random forest performed the best out of all the models we built.

Challenges Faced:

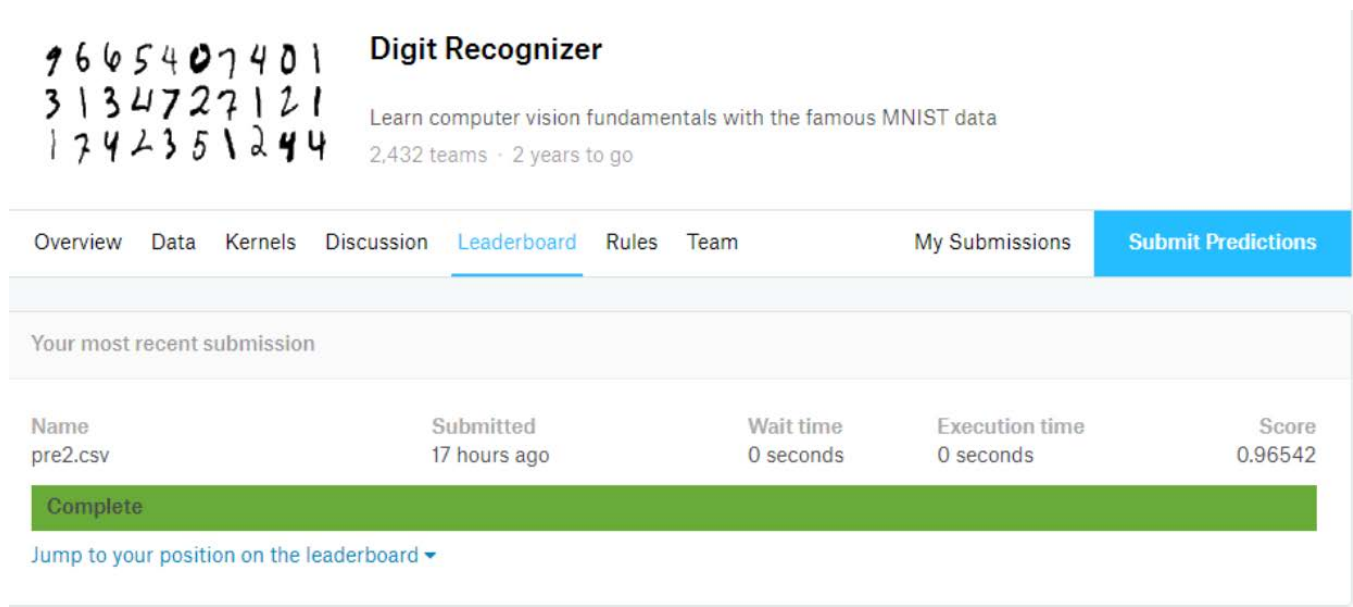
- Deciding the dataset: The foremost challenge that we faced was to make a consensus about the dataset that we wanted to use for this project. We decided to come up with 2 datasets per team member and then vote for the best one. However, there was a tie between two datasets namely Restaurant data and the Digit Recognition data. Although, we wanted to go for the restaurant data, yet, we decided to go for the later as the problem was something, none of us had experienced before and thus felt challenging and exciting.
- One of the team-members left midway: So, we were initially a team of 5 and were quite excited to work together and had delegated the tasks amongst each team member. Everything was going smooth until we learnt that one of our team members had dropped out of the course and thus won't be participating anymore.
- Huge dataset for our machines: Our dataset was 73 MB which is huge for our machines to process and thus would take a lot of time to process.

- Team coordination: Each one of us had different schedules and other work commitments and thus at times it was difficult to set a meeting.

Applications:

The application of a digit recognizer has been in practice since early 1980s as various commercial products incorporated handwriting recognition instead of keyboard input. Digit recognition is a well-researched problem and we found a lot research papers on the same, which are mentioned below in reference. It has an importance in several fields and it may probably be used in checks in banks or for recognizing numbers in cars plates, or many other applications.

Our Team's score on Kaggle:



The screenshot shows the Kaggle Digit Recognizer competition page. At the top, there's a header with the competition title "Digit Recognizer" and a description "Learn computer vision fundamentals with the famous MNIST data" along with "2,432 teams · 2 years to go". Below this is a navigation bar with tabs: Overview, Data, Kernels, Discussion, Leaderboard (active), Rules, Team, My Submissions, and a blue button for Submit Predictions. The main content area shows "Your most recent submission" with a table of submission details. The table has columns for Name, Submitted, Wait time, Execution time, and Score. The submission "pre2.csv" was submitted 17 hours ago, with 0 seconds wait and execution time, and a score of 0.96542. A green bar indicates the submission is "Complete". Below the table is a link to "Jump to your position on the leaderboard".

Name	Submitted	Wait time	Execution time	Score
pre2.csv	17 hours ago	0 seconds	0 seconds	0.96542

Complete

[Jump to your position on the leaderboard](#)

What would we like to do differently if do it next time:

Although we could try more data mining techniques to increase the accuracy, also we could use the auto encoder to reduce the dimensionality, yet we could not perform them all because of shortage of time. We would like to dig more

into data augmentation and manipulate the pictures by examining different methods.

Conclusion:

We tried various Data mining algorithm and examine the techniques to evaluate the model like ROC, AUC, Confusion Matrix etc. After building 5 models we found out the model that generated the best result was KNN with an accuracy of 97%.

References:

- 1) LeCun, Yann, et al.” Comparison of learning algorithms for handwritten digit recognition.” International conference on artificial neural networks. Vol. 60. 1995.
- 2) Maji, Subhransu, and Jitendra Malik. “Fast and accurate digit classification.” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159 (2009).
- 3) Sundaresan, Vishnu, and Jasper Lin. “Recognizing Handwritten Digits and Characters.” (1998).
- 4) Shalev-Shwartz, Shai, Yoram Singer, and Nathan Srebro. “Pegasos: Primal estimated sub-gradient solver for svm.” Proceedings of the 24th international conference on Machine learning. ACM, 2007.