Letter Recognition using Machine Learning

Vishal Kumar
Machine Learning Intern
AI Tech Systems,
https://www.ai-techsystems.com
Ranchi,India
kumar.vishal318@gmail.com

***Abstract*—Machine Learning techniques were used to train 16000 rows of data to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The dataset was obtained from UCI Machine Learning Repository where character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. Ensemble learning was used followed by hyper parameter tuning to obtain the best result.**

Keywords—Letter Recognition, Machine Learning, Ensemble Learning, Hyper parameter tuning

I. INTRODUCTION

Machine Learning is a solution to many problems in the world. As a human we learn from observation, by trial and error and making machines learn also follows similar approach. We need to provide data, lots of it to observe and make rules, and then we test its learnings. We ask questions that he the machine has never seen before and analyze how well it answers those questions. In a nut shell this is how machine learns. For my research, I implemented a supervised learning problem in which a large number (16000) of unique examples are presented to the machine learning model (Random Forest) along with a target variable that indicates the appropriate category for each example. I used Letter Recognition dataset obtained from UCI Machine Learning Repository to train and test 19999 records across 26 categories which also makes it a problem of multinomial classification.

II. DATASET ATTRIBUTES

The dataset obtained above had 16 attributes and an attribute indicating the category of letter (A-Z). The attributes are:

A. The horizontal position, counting pixels from the left edge of the image, of the center of the smallest rectangular box that can be drawn with all "on" pixels inside the box.

B. The vertical position, counting pixels from the bottom, of the above box.

C. The width, in pixels, of the box.

D. The height, in pixels, of the box.

E. The total number of "on" pixels in the character image.

F. The mean horizontal position of all "on" pixels relative to the center of the box and divided by the width of the box. This feature has a negative value if the image is "leftheavy" as would be the case for the letter L.

G. The mean vertical position of all "on" pixels relative to the center of the box and divided by the height of the box.

H. The mean squared value of the horizontal pixel distances as measured in 6 above. This attribute will have a higher value for images whose pixels are more widely separated in the horizontal direction as would be the case for the letters W or M.

I. The mean squared value of the vertical pixel distances as measured in 7 above.

J. The mean product of the horizontal and vertical distances for each "on" pixel as measured in 6 and 7 above. This attribute has a positive value for diagonal lines that run from bottom left to top right and a negative value for diagonal lines from top left to bottom right.

K. The mean value of the squared horizontal distance times the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.

L. The mean value of the squared vertical distance times the horizontal distance for each "on" pixel. This measures the correlation of the vertical variance with the horizontal position.

M. The mean number of edges (an "on" pixel immediately to the right of either an "off" pixel or the image boundary) encountered when making systematic scans from left.

III. EXPLORATORY DATA ANALYSIS

Before doing anything I analyzed the data on hand. Our dataset contains 19999 records and 17 attributes which is described in previous section.

A. Missing value imputation

Missing values are important to impute before proceeding Therefore I checked for any missing values in the given dataset and found there are no missing values in the given dataset.

B. Basic statistics

Since all the attributes excluding the target variable are numerical in nature, therefore I checked the basic statistics namely count, mean, standard deviation, min, 1st quartile, median, 3rd quartile and max of each of the 16 numerical attributes.

From the statistics in “Fig 1”, it is observed that the minimum and maximum of all the attributes are 0 and 15 respectively which indicates the attributes has been scaled from 0-15.

	count	mean	std	min	25%	50%	75%	max
x-box	19999.0	4.023651	1.913206	0.0	3.0	4.0	5.0	15.0
y-box	19999.0	7.035452	3.304631	0.0	5.0	7.0	9.0	15.0
width	19999.0	5.121956	2.014568	0.0	4.0	5.0	6.0	15.0
height	19999.0	5.372469	2.261445	0.0	4.0	6.0	7.0	15.0
onpix	19999.0	3.505975	2.190441	0.0	2.0	3.0	5.0	15.0
x-bar	19999.0	6.897545	2.026071	0.0	6.0	7.0	8.0	15.0
y-bar	19999.0	7.500175	2.325087	0.0	6.0	7.0	9.0	15.0
x2bar	19999.0	4.628831	2.699837	0.0	3.0	4.0	6.0	15.0
y2bar	19999.0	5.178609	2.380875	0.0	4.0	5.0	7.0	15.0
xybar	19999.0	8.282164	2.488485	0.0	7.0	8.0	10.0	15.0
x2ybar	19999.0	6.453823	2.631016	0.0	5.0	6.0	8.0	15.0
xy2br	19999.0	7.928996	2.080671	0.0	7.0	8.0	9.0	15.0
x-ege	19999.0	3.046252	2.332500	0.0	1.0	3.0	4.0	15.0
xegvy	19999.0	8.338867	1.546759	0.0	8.0	8.0	9.0	15.0
y-ege	19999.0	3.691935	2.567004	0.0	2.0	3.0	5.0	15.0
yegvx	19999.0	7.801190	1.617510	0.0	7.0	8.0	9.0	15.0

Fig 1. Basic statistics of attributes.

C. *Correlation of attributes*

Correlation of variables tells a lot and in general variables which are not at all correlated should not be included in the model. I plotted a confusion matrix “Fig 2” using Seaborn’s heatmap.

It is observed that x-box and width as well as y-box and height are highly correlated positively.

Also x-bar and y-bar are highly correlated negatively.

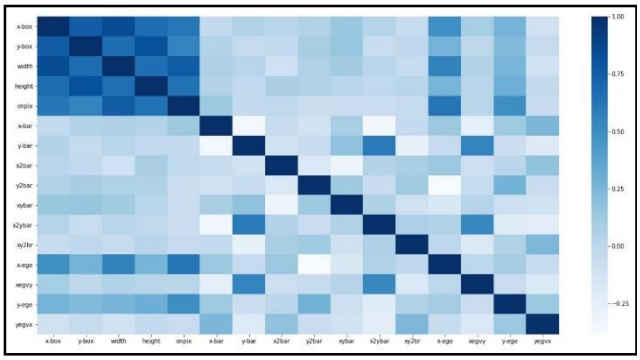

Fig 2. Correlation matrix

IV. Model Building

Before building the model the data is split into training set and test set so that the model can be trained on training set and can be checked on unseen data that is test set. Also the target attribute and all other attributes are kept in different variables as in scikit learn they both are to be passed differently.

A. *Decision Tree*

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

I trained the training set to just see how Decision tree performs and it gave me an accuracy of 87.45% using default hyperparameters.

B. *Random Forests*

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.

In my study, I first built a Random forests model using default parameters and found the testing accuracy to be 94.2% which is a big increment from decision tree.

Then I tuned the hyperparameters of Random forests using GridSearchCV(). I tuned the hyperparamters to build model using the parameters I provided and fetch me the best parameter. I provided n_estimators value from 10 to 100 in a step of 5 and min_samples_leaf value from 1 to 50 in steps of 1 and I also used k-fold cross validation with k = 10. Giving all these parameters means the model will now run for 18x50x10 = 9000 times.

After the model ran for 90000 times, I got the best parameters as n_estimators = 95 and min_samples_leaf = 1.

I build a final Random Forests model using the best paramaters and got a testing accuracy of 97.1% which is almost 3% better than the Random Forests model with default hyperparameters.

V. Report

A. *Classification Report*

A classification report “Fig 3” is a report consisting of classification metrics such as precision, recall, f1 score for each class. It also includes micro average, macro average and weighted average.

	precision	recall	f1-score	support
A	0.99	1.00	1.00	172
B	0.96	0.95	0.95	146
C	0.99	0.96	0.98	163
D	0.94	0.97	0.96	170
E	0.98	0.96	0.97	167
F	0.97	0.97	0.97	147
G	0.97	0.99	0.98	147
H	0.94	0.89	0.92	152
I	0.96	0.97	0.97	162
J	0.98	0.93	0.95	149
K	0.93	0.94	0.93	143
L	0.99	0.98	0.99	156
M	0.95	0.98	0.97	150
N	0.99	0.97	0.98	147
O	0.96	0.98	0.97	123
P	0.97	0.98	0.97	168
Q	0.97	0.97	0.97	177
R	0.92	0.99	0.95	143
S	0.96	0.98	0.97	154
T	0.99	0.99	0.99	166
U	0.99	0.98	0.99	153
V	0.96	0.97	0.97	147
W	0.99	0.99	0.99	155
X	0.98	0.97	0.98	152
Y	1.00	0.99	0.99	153
Z	1.00	0.99	0.99	138
micro avg	0.97	0.97	0.97	4000
macro avg	0.97	0.97	0.97	4000
weighted avg	0.97	0.97	0.97	4000

Fig 3. Classification Report

B. *Confusion Matrix*

A confusion matrix in case binary classification is a matrix consisting of True Positives, False Positives, False Negatives and True Negatives. But confusion matrix "Fig 4" in case of multiclass classification, has True Positives and rest all False Positives, False Negatives and True Negatives are considered as errors.

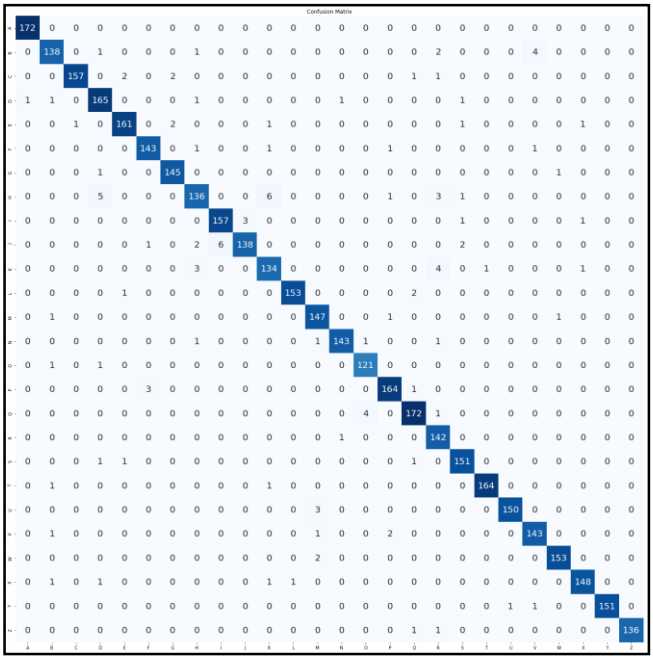

Fig 4. Confusion Matrix

VI. CONCLUSION

After creating both the models –"Decision Tree" and "Random Forests", it is evident from their accuracy that Random Forests clearly outperforms Decision Tree. Also Random Forests model without hyperparamter tuning is 3% less accurate than Random Forests with tuned hyperparameters which concludes that we should always tune our model's hyperparameters for better result. One point that I would like to mention here is that hypeparameter tuning takes lot of time, depending on your computer processing capabilities.

The final model is able to classify the capital English alphabets with 97.1% accuracy.

REFERENCES

[1] Peter W. Frey and David J. Slate, "Letter recognition using holland-style adaptive classifiers", in Kluwer Academic Publishers, Boston, 1991.