



Team Esh: Chris Muro, Andrew Greensweight, Marc DiGeronimo

 **Leaderboard**

[Raw Data](#) [Refresh](#)

YOUR RECENT SUBMISSION


























 **kaggle_submission_ensemble.csv**
Submitted by midigeron · Submitted 14 hours ago

Score: 0.851
Private score:

[Jump to your leaderboard position](#)

[Public](#) [Private](#)

This leaderboard is calculated with approximately 60% of the test data. The final results will be based on the other 40%, so the final standings may be different.

#	Team	Members	Score	Entries	Last	Join
1	Eric Liotta		0.902	24	9d	
2	Anthony Paolantonio		0.876	10	9d	
3	Shashankk Shekar Chaturvedi		0.874	9	7d	
4	Derick Miller		0.870	36	11h	
5	Dingxin Hu		0.870	15	9d	
6	TEAM ROCKET	  	0.870	10	8d	
7	jdawg		0.869	47	9d	
8	Brandon Tai		0.865	54	11h	
9	esh	  	0.857	118	14h	
<div> Your Best Entry! Your submission scored 0.851, which is not an improvement of your previous score. Keep trying!</div>						
10	Apex	  	0.857	12	8d	
11	N-R Team	 	0.856	33	9d	
12	Endurance	  	0.497	2	1mo	
13	lohithburra01	  	0.496	2	1mo	

Current Best Score = .857 Total Successful Submissions = 118

For HW10, the prediction vectors from kaggle needed to be redownloaded and renamed to be used for the code. A folder was created called ensemble_submissions which contains all of the prediction vectors used in the ensemble method. The vectors come from kaggle downloaded as a csv in the proper submission format. It is downloaded to the ensemble submission folder and also renamed to add the accuracy score at the end of the file. When downloading submissions

for the first test, 8 submissions were used. Each submission was using different methods throughout the semester such as weighted features, matrix factorization, logistic regression, decision trees, random forest classifier, gradient boosted tree classifier and multi-layer perceptron classifier. To make sure that none of the submissions were duplicates, each had a different accuracy value in the range .766 to .857. The first part of the code is as follows

```

9 accuracy = []
10 # corresponding filename for accuracy scores
11 files = []
12 # holds the prediction vectors from each submission
13 S = []
14 STx = []
15 ratings = []
16 num_prediction_vectors = 0
17
18 # make a few folder under home directory named ensemble_submissions
19 directory = os.fsencode("/home/ensemble_submissions")
20
21 set_ratings = 0

```

After the necessary libraries were imported some variables are initialized. Accuracy list holds the accuracy values from the prediction vectors. Files is a list that holds the filenames. It is useful for testing since the filenames have the accuracy scores in it so it is beneficial for checking that the correct accuracy scores for the corresponding file. S is a list that contains the prediction vectors from the submission files. STx contains the S transpose * x vector from the formula

$$s_{\text{ensemble}} = a_1 s_1 + a_2 s_2 + \dots + a_K s_K = S \cdot a_{1:K} = S (S^T S)^{-1} S^T x$$

With S described earlier being the other component. Ratings will become a dataframe of one of the submissions later in the code which will be updated and used for the ensemble predictions submission since it will already be in the correct kaggle submission format.

Num_prediction_vectors contains the total number of prediction vectors in the ensemble, this is later used for more descriptive submission messages. Directory is the path to the ensemble submissions folder on colab. The reason for encoding it will be seen shortly. Set_ratings is a flag used so the ratings variable is not constantly reassigned for optimization. The next part of the code is as follows.

```

23 for file in os.listdir(directory):
24     filename = os.fsdecode(file)
25     if filename[-4:] == ".csv":
26         num_prediction_vectors += 1
27         files.append(filename)
28         accuracy_score = int(filename[-7:][:3])/1000
29         accuracy.append(accuracy_score)
30         df = pd.read_csv("/home/ensemble_submissions/" + filename)
31         S.append(np.array(df["Predictor"]))
32         N = len(df["Predictor"]) # should be 120000
33         Pi = accuracy_score
34         stx = 2*Pi-1
35         STx.append(stx)
36         if set_ratings == 0: # only do once
37             ratings = df
38             set_ratings +=1

```

Directory variables needed to be encoded in order to properly index through the contents of the folder for processing. The corresponding file must be decoded to be used for other functions. Since the python notebook file creates hidden folders for the checkpoints an if statement is used to make sure the code is only reading in csv files. If a csv file is read then num_prediction_vectors is incremented and the file name is added to the files list. Here is an example filename "output3_848.csv" ; the 848 represents the accuracy score from kaggle. To retrieve it the string filename is broken up 2 times. The first will only look at the last 7 characters since that would be "848.csv" then the first 3 characters of that string are taken so "848" and cast to an integer and divided by 1000 to get the correct accuracy value. It is then added to the accuracy list at the same index as filename in the files list. Now the csv file is read as a dataframe and the predictions vector is appended to the S list. The next steps are based on the following formula on canvas.

$$S^T \mathbf{x} = \begin{bmatrix} N(2P_1 - 1) \\ N(2P_2 - 1) \\ \vdots \\ N(2P_K - 1) \end{bmatrix}$$

N is the total number of predictions/length of the Predictor column of the dataframe. Pi is the accuracy score which was calculated previously and stx for that dataframe is 2*accuracy_score -1 as given. The result is appended to the STx list. Now the flag is used to check if the ratings variable has been set. If not, the ratings variable is set to the current dataframe and flag is set. If it has been set then there is no need to reset it since the TrackID column will be the same and the predictions will be changed later in the code.

```

39
40 for i in range(len(S)):
41     for j in range(len(S[0])):
42         if S[i][j] == 0:
43             S[i][j] = -1
44

```

This code shows the process to remap all of the 0 predictions to -1.
The next piece of code is used for the formula mentioned earlier

$$\mathbf{S}_{\text{ensemble}} = a_1 \mathbf{s}_1 + a_2 \mathbf{s}_2 + \dots + a_K \mathbf{s}_K = \mathbf{S} \cdot \mathbf{a}_{LS} = \mathbf{S} (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{x}$$

```

45 S = np.array(S)
46 STx = np.array(STx)
47
48 esh = np.matmul(S, np.transpose(S))
49 iesh = np.linalg.inv(esh)
50 sesh = np.matmul(np.transpose(S), iesh)
51 Sensemble = np.matmul(sesh, STx)

```

In the code S and STx are flipped from the formula because of the way numpy arrays are structured. If this was MATLAB for example S and STx would be done in the same way as the formula.

Next code is used for determining the updated ratings.

```

54 for i in range(len(Sensemble)):
55     if (i+1) % 6 == 0: # 0-5 hold ratings for first user, do matrix calculation
56         S_sub_ensemble = Sensemble[i-5:i+1]
57         S_sub_ensemble_sort_max = list(reversed(sorted(S_sub_ensemble)))[0:3]
58         index = 0
59         for j in range(i-5, i):
60             if S_sub_ensemble[index] in S_sub_ensemble_sort_max:
61                 ratings.at[j, "Predictor"] = 1
62             else:
63                 ratings.at[j, "Predictor"] = 0
64             index += 1
65

```

Sensemble holds the decimal values between -1 and 1 that are used to make the predictions. The ratings are done in groups of 6 values since those represent the ratings for a single user. For each user S_sub_ensemble list is the group of 6 weights for the users ratings. The list needs to be sorted and take the top 3 highest values. Now the index is set as 0 and used to check if the weight is in the top 3. This is done because S_sub_ensemble is consistently in range 0-5 whereas inside the for loop j is used since that needs to be in the range of the entirety of the ratings dataframe to be able to change any index. So if the weight is in the top 3 the corresponding prediction in the final ensembled data frame the prediction is set to 1. If it is not it is set to 0.

The ratings dataframe is now updated and converted to a csv file for kaggle submissions.

On colab the code also contains automated kaggle submissions. To do this, kaggle python library is installed and user kaggle.json file is uploaded to colab in the .kaggle folder and the permissions are changed for it to work. The following commands are then used.

```
70  
71 !kaggle competitions submit -c aai627-spring2024 -f kaggle_submission_ensemble.csv -m "Ensemble Method num_prediction_vectors = $num_prediction_vectors"  
72 !kaggle competitions submissions -c aai627-spring2024
```

First command submits the file with a message that contains the total number of prediction vectors used to add more description. The results are also shown.

The top ensemble submission was .857 which is not an improvement from the previous best of .857. When adding more predictions the score decreases. This is possibly due to the increasing amount of lower accuracy submissions resulting in more wrong predictions being used. All of the submissions cannot be used together since many of the solutions are identical because many of the submissions were changes to weights and hyperparameters and scores either remained the same, increased slightly or decreased significantly. Therefore we did not find ensemble method to increase our scores based on our kaggle submissions.