Classification Algorithms

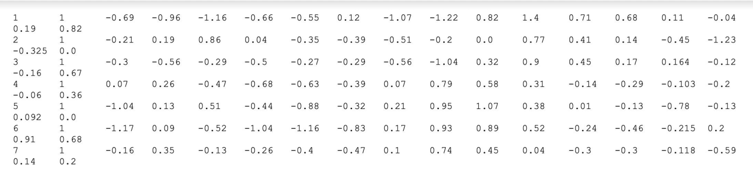
CSE347

Lauren Bright
Sophie Makaridi
Yanqi Liu

Choice of Datasets and Algorithms

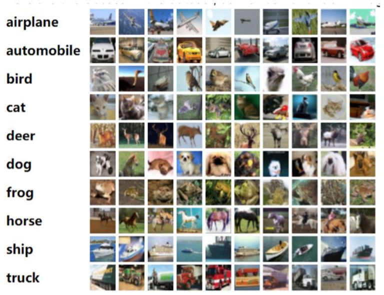
Datasets:

Cho



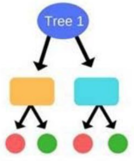
```
1       1       -0.69   -0.96   -1.16   -0.66   -0.55   0.12    -1.07   -1.22   0.82    1.4     0.71    0.68    0.11    -0.04
0.19    0.82
2       1       -0.21   0.19    0.86    0.04    -0.35   -0.39   -0.51   -0.2    0.0     0.77    0.41    0.14    -0.45   -1.23
-0.325  0.0
3       1       -0.3    -0.56   -0.29   -0.5    -0.27   -0.29   -0.56   -1.04   0.32    0.9     0.45    0.17    0.164   -0.12
-0.16   0.67
4       1       0.07    0.26    -0.47   -0.68   -0.63   -0.39   0.07    0.79    0.58    0.31    -0.14   -0.29   -0.103  -0.2
-0.06   0.36
5       1       -1.04   0.13    0.51    -0.44   -0.88   -0.32   0.21    0.95    1.07    0.38    0.01    -0.13   -0.78   -0.13
0.092   0.0
6       1       -1.17   0.09    -0.52   -1.04   -1.16   -0.83   0.17    0.93    0.89    0.52    -0.24   -0.46   -0.215  0.2
0.91    0.68
7       1       -0.16   0.35    -0.13   -0.26   -0.4    -0.47   0.1     0.74    0.45    0.04    -0.3    -0.3    -0.118  -0.59
0.14    0.2
```

Cifar10

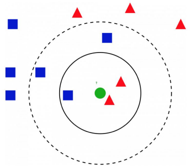


Algorithms:

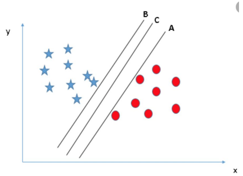
Random Forest Classifier



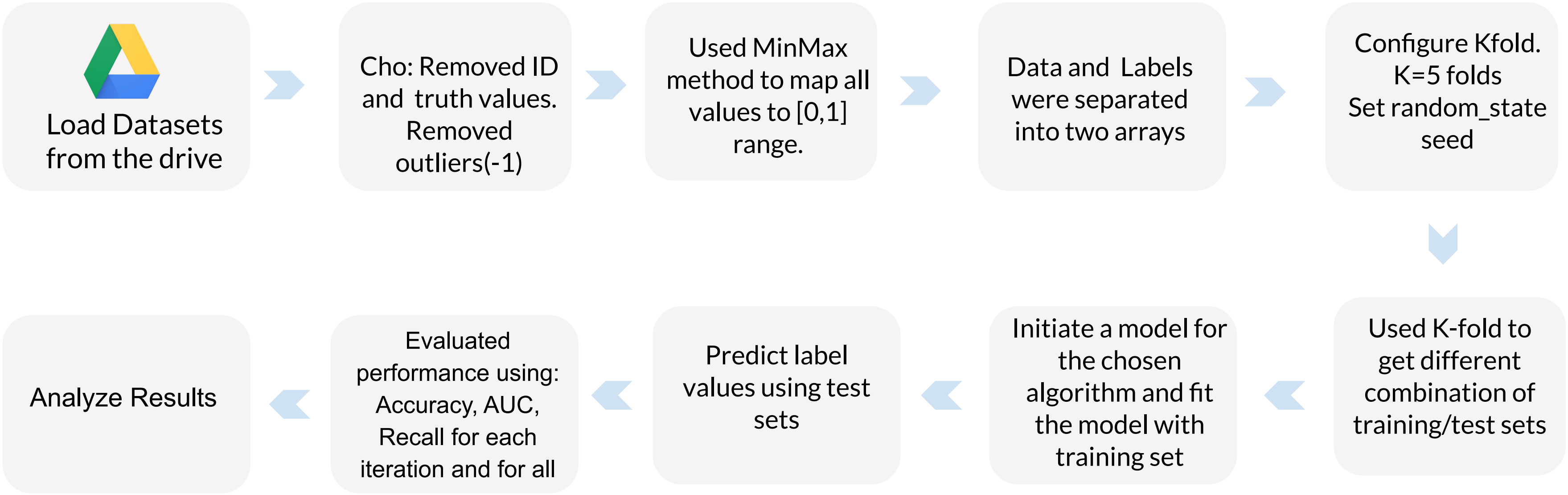
K-nearest neighbors



Support Vector Machine



Data Processing and Classifying



Load Datasets from the drive

Cho: Removed ID and truth values. Removed outliers(-1)

Used MinMax method to map all values to [0,1] range.

Data and Labels were separated into two arrays

Configure Kfold.
K=5 folds
Set random_state seed

Used K-fold to get different combination of training/test sets

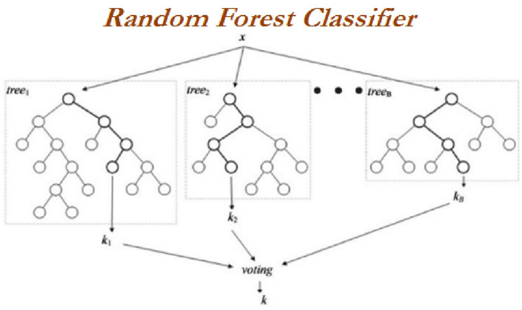
Initiate a model for the chosen algorithm and fit the model with training set

Predict label values using test sets

Evaluated performance using: Accuracy, AUC, Recall for each iteration and for all

Analyze Results

Algorithm Overview - Random Forest



How it works:

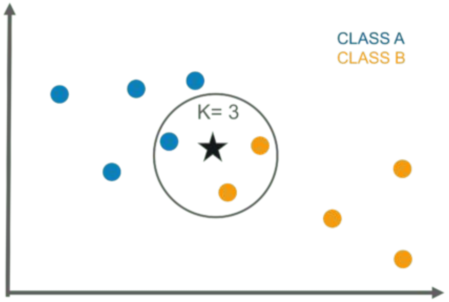
- Random forest is an ensemble, tree-based algorithm
- The RF classifier generates a set of classification trees that each classify an object and “vote” for the class of the object
  - In this implementation, each tree outputted a probabilistic prediction, so not just a vote for one class, but different weights that could contribute to several classes
- The overall forest then considers all votes and chooses a final classification for the object
- Having several trees cuts down on variance and overfitting that can occur with just one tree
- The main parameter is **n_estimators**, which determines the number of trees in the forest
  - n_estimators = 100 was chosen with the same seed for each run
  - Choosing a higher value for n_estimators could increase accuracy, but greatly increases run time

Image from: https://www.mygreatlearning.com/blog/random-forest-algorithm/

Algorithm Overview - K-Nearest Neighbor

How it works:

- The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm.
- Assumes that similar things exist in close proximity.
- Stores all available cases and classifies new cases based on a similarity measure.
  - Distance function - Euclidean distance
- A case is classified by a vote of its neighbors.
  - Majority voting vs. weighted voting



Algorithm Overview - Support Vector Machine

How it Works:

- ❖ find a hyperplane in an N-dimensional space to separate classes
- ❖ Out of all possible hyperplanes maximize the margin.
- ❖ Uses Lagrangian to solve the optimization problem
- ❖ When not linearly separable, use Kernel functions : linear, poly, rbg, sigmoid, precomputed, callable

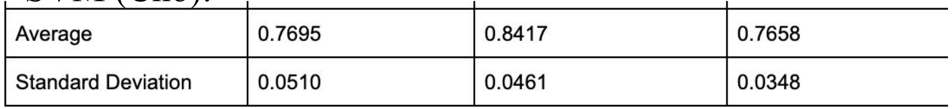
- ❖ I used the Radial Basis Function (RBF).
  - ➢ Creates additional features to increase dimensions
  - ➢ Points that were hard to classify linearly, become easily separable in higher dimensions.

Setbacks - difficulties

- K-nearest neighbors and Support vector machine are time inefficient on large datasets
  - Cifar10 took hours to finish
  - Made it hard to test with different parameters
  - A way to go around was to test on a smaller part of Cifar
- Working with datasets
  - Datasets were fairly different (Image vs. numerical)
  - Understanding datasets, features, type, size
- Detecting Outliers/noise
  - Hard to see for cho
  - Easier for Cifar since images are easier to visualize

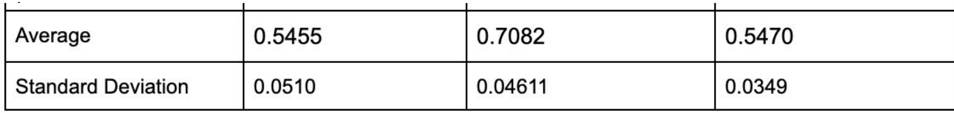
Comparing Results

★ SVM (Cho):



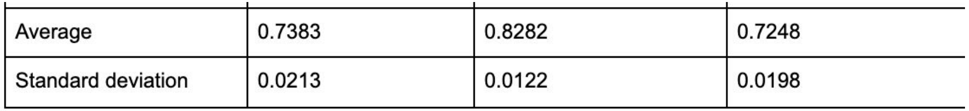
Average	0.7695	0.8417	0.7658
Standard Deviation	0.0510	0.0461	0.0348

★ SVM (CIFAR):



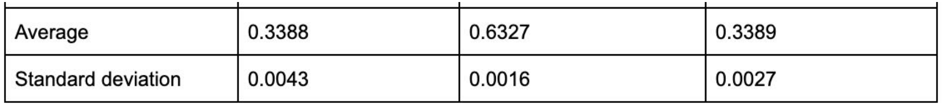
Average	0.5455	0.7082	0.5470
Standard Deviation	0.0510	0.04611	0.0349

KNN (Cho):



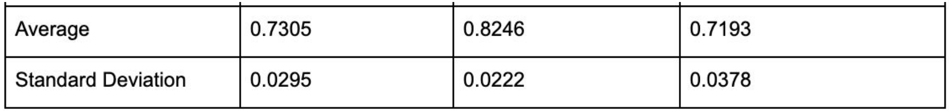
Average	0.7383	0.8282	0.7248
Standard deviation	0.0213	0.0122	0.0198

KNN (CIFAR):



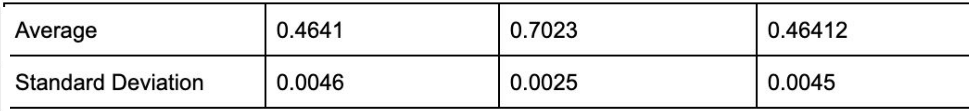
Average	0.3388	0.6327	0.3389
Standard deviation	0.0043	0.0016	0.0027

Random Forest (Cho):



Average	0.7305	0.8246	0.7193
Standard Deviation	0.0295	0.0222	0.0378

Random Forest (CIFAR):



Average	0.4641	0.7023	0.46412
Standard Deviation	0.0046	0.0025	0.0045

Observations and takeaways

Observations:

- Large vs. small datasets with different algorithms
- Different types of data perform differently
  - Image vs. numerical values
  - Question of quality of data

Takeaways:

- Parameter tuning is difficult and time consuming
  - Many parameters to consider
- Different algorithms work better for certain tasks