CS412 - Machine Learning - 2020

Homework 1

100 pts

Goal

The goal of this homework is three-fold:

- Introduction to the machine learning experimental set up
- Gain experience with Decision tree approache
- Gain experience with the Scikit library

Dataset

MNIST is a collection of 28x28 grayscale images of digits (0-9); hence each pixel is a gray-level from 0-255.

Download the data from Keras. You must use a 20% of the training data for validation (no need for cross-validation as you have plenty of data) and **use the official test data (10,000 samples) only for testing.**

Task

Build a decision tree classifier with the scikit library function calls to classify digits in the MNIST dataset.

Software: You may find the necessary function references here:

http://scikit-learn.org/stable/supervised_learning.html

Submission:

Fill this notebook and submit this document with a link to #your Colab notebook (make sure to include the link obtained from the #share link on top right)

1) Initialize

- First make a copy of the notebook given to you as a starter.
- Make sure you choose Connect form upper right.

2) Load training dataset

- Read from Keras library.

```
# Load the Pandas libraries with alias 'pd'
import pandas as pd

# Read data
import keras
from keras.datasets import mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
11493376/11490434 [==============================] - 0s 0us/step
```

3) Understanding the dataset

There are alot of functions that can be used to know more about this dataset

- What is the shape of the training set (num of samples X number of attributes) ***[shape function can be used]***
- Display attribute names ***[columns function can be used]***
- Display the first 5 rows from training dataset ***[head or sample functions can be used]***

..

```
import numpy as np

x_train_flattened = x_train.reshape(x_train.shape[0],x_train.shape[1]*x_train.shape[2]) #prep
train_dataframe = pd.DataFrame(x_train_flattened) #taken from TA's e-mail
train_dataframe['label'] = y_train

# print shape
print('Data Dimensionality: ')
print(x_train_flattened.shape)

print('Attributes: ')
print(train_dataframe.columns)

# print first 5 rows in your dataset
print('Head of Data: ')
train_dataframe.head()
```

```
Data Dimensionality:
(60000, 784)
Attributes:
Index([      0,       1,       2,       3,       4,       5,       6,       7,
             8,       9,
       ...
           775,     776,     777,     778,     779,     780,     781,     782,
           783, 'label'],
      dtype='object', length=785)
Head of Data:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	2
0	0	0	0	0	0	0	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	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	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	0	0	0	0	0	

4) Shuffle and Split TRAINING data as train (also called development) (80%) and validation (20%)

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

# Shuffle the training data
train_dataframe = shuffle(train_dataframe, random_state=42)
print("Training data shuffled.")

# Split 80-20
x_dev, x_val, y_dev, y_val = train_test_split(x_train_flattened, y_train, test_size = 0.2, ra

print("Train data shape:", x_dev.shape, "and train label shape:", y_dev.shape)
print("Validation data shape:", x_val.shape, "and validation label shape:", y_val.shape)
```

```
Training data shuffled.
Train data shape: (48000, 784) and train label shape: (48000,)
Validation data shape: (12000, 784) and validation label shape: (12000,)
```

5) Train a decision tree classifier on development/train data and do model selection using the validation data

- Train 3 decision tree classifiers with different values of "min_samples_split" which is the minimum number of samples required to split an internal node: min_samples_split = [default =

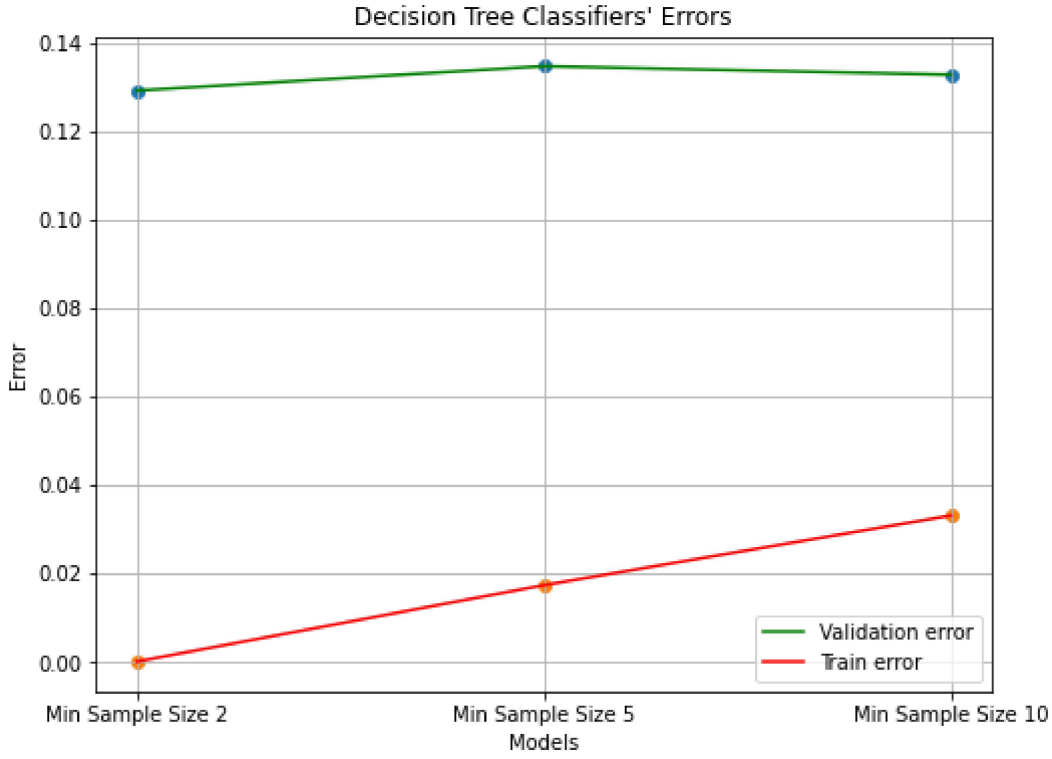
2, 5, 10].

- Test the 3 models on validation set and choose the best one.
- Plot the train and validation set errors for those 3 settings - on one plot.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Train decision tree classifiers
tree1 = DecisionTreeClassifier(min_samples_split=2)
tree2 = DecisionTreeClassifier(min_samples_split=5)
tree3 = DecisionTreeClassifier(min_samples_split=10)

tree1.fit(x_dev,y_dev) #fitting the trees into train data
tree2.fit(x_dev,y_dev)
tree3.fit(x_dev,y_dev)

y1_train = tree1.predict(x_dev) #predicting train labels
y2_train = tree2.predict(x_dev)
y3_train = tree3.predict(x_dev)

accuracytrain1 = accuracy_score(y_dev,y1_train) #train data accuracies
accuracytrain2 = accuracy_score(y_dev,y2_train)
accuracytrain3 = accuracy_score(y_dev,y3_train)

print("Training accuracy for min sample size = 2:", accuracytrain1)
print("Training accuracy for min sample size = 5:", accuracytrain2)
print("Training accuracy for min sample size = 10:", accuracytrain3)

# Evaluate on validation set
y1 = tree1.predict(x_val) #predictin validation labels
y2 = tree2.predict(x_val)
y3 = tree3.predict(x_val)

accuracyvalid1 = accuracy_score(y_val,y1) #validation data accuracies
accuracyvalid2 = accuracy_score(y_val,y2)
accuracyvalid3 = accuracy_score(y_val,y3)

print("Validation set accuracy for min sample size = 2:", accuracyvalid1)
print("Validation set accuracy for min sample size = 5:", accuracyvalid2)
print("Validation set accuracy for min sample size = 10:", accuracyvalid3)

# Plot errors
error_train = [(1 - accuracytrain1), (1 - accuracytrain2), (1- accuracytrain3)] #train errors
error_val = [(1 - accuracyvalid1),(1 - accuracyvalid2),(1 - accuracyvalid3)] #validation erro
plt.figure(figsize=(8,6))
x_axis = ['Min Sample Size 2', 'Min Sample Size 5', 'Min Sample Size 10'] #taken from Berrin
plt.scatter(x_axis, error_val)
plt.scatter(x_axis, error_train)
plt.plot(x_axis, error_val, color='green', label = "Validation error")
```

```
plt.plot(x_axis, error_train, color='red', label = 'Train error')
plt.xlabel('Models')
plt.ylabel('Error')
plt.title("Decision Tree Classifiers' Errors")
plt.grid(True)
plt.legend()
plt.show()
```

```
Training accuracy for min sample size = 2: 1.0
Training accuracy for min sample size = 5: 0.98275
Training accuracy for min sample size = 10: 0.967
Validation set accuracy for min sample size = 2: 0.8709166666666667
Validation set accuracy for min sample size = 5: 0.8654166666666666
Validation set accuracy for min sample size = 10: 0.8673333333333333
```



7) Test your CHOSEN classifier on Test set

- Load test data
- Apply same pre-processing as training data (probably none)
- Predict the labels of testing data **using the best chosen SINGLE model out of the models that you have tried from step 6 (you have selected your model according to your validation results)** and report the accuracy.

```
# Load test data
x_test_flattened = x_test.reshape(x_test.shape[0],x_test.shape[1]*x_test.shape[2]) #same prep

# test prediction using a decision tree with all default parameters and ..... min-split value
ytestpred = tree1.predict(x_test_flattened) #tree1 with min sample size 2 was chosen
```

```
ytestpred = tree1.predict(x_test_flattened) #tree1 with min sample size 2 was chosen

# Report your accuracy
print("Test set accuracy for min sample size = 2:", accuracy_score(ytestpred, y_test))
```

```
Test set accuracy for min sample size = 2: 0.8751
```

8) Notebook & Report

Notebook: We may just look at your notebook results; so make sure each cell is run and outputs are there.

Report: Write an at most 1/2 page summary of your approach to this problem at the end of your notebook; this should be like an abstract of a paper or the executive summary (you aim for clarity and passing on information, not going to details about known facts such as what dec. trees are or what MNIST is, assuming they are known to people in your research area).

Must include statements such as:

(Include the problem definition: 1-2 lines)

(Talk about train/val/test sets, size and how split.)

(Talk about any preprocessing you do.)

(Give the validation accuracies for different approaches, parameters **in a table** and state which one you selected)

(State what your test results are with the chosen method, parameters: e.g. "We have obtained the best results with the ….. classifier (parameters=....) , giving classification accuracy of …% on test data…."

(Comment on the speed of the algorithms and anything else that you deem important/interesting (e.g. confusion matrix)).

You will get full points from here as long as you have a good (enough) summary of your work, regardless of your best performance or what you have decided to talk about in the last few lines.

Report:

The aim of this homework was to classify digits of MINST dataset by training Decision Tree Classifiers on training data and choose the best model according to accurracy scores on the validation data and observe its accuracy on the test set.

MINST dataset itself consists of training part which has 60,000 samples and test part which has 10,000 samples. The training set was first shuffled and then separated into 2 parts, 80% of it (48,000 samples) for training/development and the rest 20% (12,000 samples) for validation part of

the training. In terms of preproccesing, data downloaded was in a form of numpy arrays but it was flattened to 2-dimensions by keeping the number of data points same for further operations, it was also converted to pandas dataframe structure for analysis purposes. This procedure was done for both training and test part of the dataset separately.

3 different decision trees which use classification method were modelled for this homework with different minimum sample sizes, 2, 5 and 10 respectively, with all default parameters for other values. The decision trees were fit into the training data and their accurracy scores were observed in both training and validation sets. Validation accuracies for different approaches can be observed in the table below:

	Min Sample Size = 2	Min Sample Size = 5	Min Sample Size = 10
Validation Set Accuracies	0.87091	0.86541	0.86733

Although the validation set accuracies were really close in all 3 trees, when the minimum sample size was 2, the best accuracy score (least error) was obtained, that's why it was chosen. Minimum sample size 2 tree gave 87.51% classification accuracy on test data.

Most time consuming part is training decision tree classifiers and testing on validation and test sets. What was interesting to me was when I first saw accuracy of 100% with minimum sample size 2 tree on training data, I had a preconception that it could be an overfit, I thought that it would not generalize well. That's why I was really suprised when I saw that it performed the best on validation set and definitely learned not to decide just by looking at the train dataset.

9) Submission

Please submit your "**share link" INLINE in Sucourse submissions**. That is we should be able to click on the link and go there and run (and possibly also modify) your code.

For us to be able to modify, in case of errors etc, **you should get your "share link" as **share with anyone in edit mode**

Also submit your notebook as pdf as attachment, choose print and save as PDF, save with hw1-lastname-firstname.pdf to facilitate grading.

Questions?

You can and should ask all your Google Colab related questions under Forums and feel free to answer/share your answer regarding Colab.

You can also ask/answer about which functions to use and what libraries...

However you should **not ask** about the core parts, that is what is validation/test, which one shd. have higher performance, what are your scores etc.