CSC411 A1

Casey Juanxi Li - 998816973

September 2018

1 Nearest Neighbours and the Curse of Dimensionality.

a) Expectation and variance of $Z = (X - Y)^2$, where $X, Y \sim U(0, 1)$:

Expectation: Expected value of iid random variable equals that variable multiplied by its PDF, integrated over all values of the variable. We assume X, Y are iid and P(X,Y) = P(X)P(Y).

$$\begin{aligned} E[Z] &= \int Z \cdot P(Z)\, dZ \\ &= \int_0^1 \int_0^1 (X - Y)^2 \cdot P(Y)P(X)\, dY\, dX \\ &= \int_0^1 \int_0^1 (X - Y)^2 \cdot 1 \cdot 1\, dY\, dX \qquad (X, Y \sim U(0, 1)) \\ &= \int_0^1 \int_0^1 (X - Y)^2, dY\, dX \end{aligned}$$

Numerically:

```
def z(y, x):
  return (x-y) ** 2

# Expected Z
E_Z, E_Z_err = scipy.integrate.dblquad(
  z, # dYdX
  0, 1, # outside integral
  lambda x: 0, lambda x: 1, # inside integral
)
print("E_Z: {}".format(E_Z))
```

Result: `E_Z: 0.16666666666666666`, or 1/6

Variance:

$$\begin{aligned} Var[Z] &= E[Z^2] - (E[Z])^2 \\ &= \int Z \cdot P(Z)\, dZ - (E[Z])^2 \\ &= \int_0^1 \int_0^1 (X - Y)^4 \cdot P(Y)P(X)\, dY\, dX - (E[Z])^2 \\ &= \int_0^1 \int_0^1 (X - Y)^4 \cdot 1 \cdot 1\, dY\, dX - (E[Z])^2 \\ &= \int_0^1 \int_0^1 (X - Y)^4, dY\, dX - (E[Z])^2 \end{aligned}$$

Numerically:

```
# Variance definition: E[Z^2] - (E[Z])^2
exp_Z_sq, exp_Z_sq_err = scipy.integrate.dblquad(
  z_sq, # dYdX
  0, 1,
  lambda x: 0,
  lambda x: 1, # inside integral
  )

Var_Z = exp_Z_sq - (E_Z) ** 2

print("Var_Z: {}".format(Var_Z))
```

Result: `Var_Z: 0.0388888888888889`, or 7/180

b) Expectation and variance of $R = Z_1 + Z_2 + ... + Z_d$:

Expectation: Given linearity of expectation, $E[A + B] = E[A] + E[B]$:

$$E[R] = E[Z_1] + E[Z_2] + ... + E[Z_d]$$

$Z_1, Z_2, ...Z_d$ are all iid and have same distribution as Z, therefore:

$$E[R] = d \cdot E[Z]$$

Variance: Given $Var(X + Y) = Var(X) + Var(Y) + 2Cov(X, Y)$, and $Cov(Z_i, Z_j) = 0$ since all Z's are iid:

$$\begin{aligned} Var[R] &= Var[Z_1] + Var[Z_2] + ... + Var[Z_d] \\ &= d \cdot Var[Z] \end{aligned}$$

Run some tests in Python to verify that the above is correct, for say, d=2:

```
total = 0
var_total = 0
trials = 1000000
d = 2

E_Z = 1/6
Var_Z = 7/180

for i in range(0, trials):
  for j in range(0, d):
    x = random.uniform(0,1)
    y = random.uniform(0,1)

    z = (x - y) ** 2
    total += z
    var_total += (z - E_Z) ** 2

print("Mean: {:0.5f} | Theoretical: {:0.5f} | delta: {:0.5f}".format(
  total / trials,
  E_Z * d,
  total / trials - E_Z * d))
print("Variance: {:0.5f} | Theoretical: {:0.5f} | delta: {:0.5f}".format(
  var_total / trials,
  Var_Z*d,
  var_total / trials - Var_Z*d))
```

Result:

```
Mean:  0.33374 | Theoretical:  0.33333 | delta:  0.00040
Variance:  0.07783 | Theoretical:  0.07778 | delta:  0.00005
```

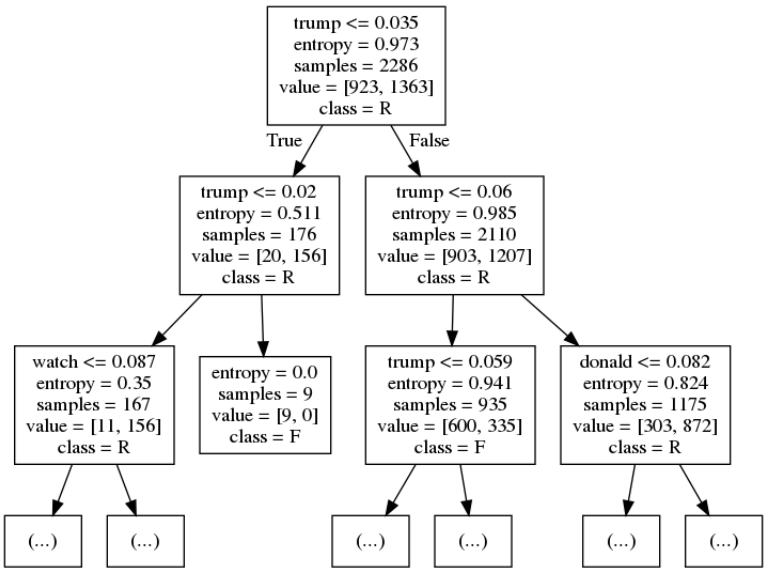
2 Decision Trees.

a) Done in `hw1_code.py`

b) `select_model()` output:

```
max_depth: 1  | criterion: gini    | score: 0.5898
max_depth: 1  | criterion: entropy | score: 0.7143
max_depth: 2  | criterion: gini    | score: 0.6102
max_depth: 2  | criterion: entropy | score: 0.5980
max_depth: 3  | criterion: gini    | score: 0.6878
max_depth: 3  | criterion: entropy | score: 0.6878
max_depth: 4  | criterion: gini    | score: 0.6755
max_depth: 4  | criterion: entropy | score: 0.6592
max_depth: 5  | criterion: gini    | score: 0.6653
max_depth: 5  | criterion: entropy | score: 0.7469
max_depth: 6  | criterion: gini    | score: 0.7204
max_depth: 6  | criterion: entropy | score: 0.7429
max_depth: 7  | criterion: gini    | score: 0.6878
max_depth: 7  | criterion: entropy | score: 0.7388
max_depth: 8  | criterion: gini    | score: 0.8041
max_depth: 8  | criterion: entropy | score: 0.8163
max_depth: 9  | criterion: gini    | score: 0.8122
max_depth: 9  | criterion: entropy | score: 0.7367
max_depth: 10 | criterion: gini    | score: 0.7510
max_depth: 10 | criterion: entropy | score: 0.8245
max_depth: 11 | criterion: gini    | score: 0.7510
max_depth: 11 | criterion: entropy | score: 0.7367
max_depth: 12 | criterion: gini    | score: 0.7571
max_depth: 12 | criterion: entropy | score: 0.7612
max_depth: 13 | criterion: gini    | score: 0.7694
max_depth: 13 | criterion: entropy | score: 0.8224
max_depth: 14 | criterion: gini    | score: 0.7510
max_depth: 14 | criterion: entropy | score: 0.8327
max_depth: 15 | criterion: gini    | score: 0.8347
max_depth: 15 | criterion: entropy | score: 0.8224
max_depth: 16 | criterion: gini    | score: 0.8306
max_depth: 16 | criterion: entropy | score: 0.8347
max_depth: 17 | criterion: gini    | score: 0.8245
max_depth: 17 | criterion: entropy | score: 0.8286
max_depth: 18 | criterion: gini    | score: 0.8184
max_depth: 18 | criterion: entropy | score: 0.8490
max_depth: 19 | criterion: gini    | score: 0.8061
max_depth: 19 | criterion: entropy | score: 0.8306
max_depth: 20 | criterion: gini    | score: 0.8265
max_depth: 20 | criterion: entropy | score: 0.8163
---
Best hyperparameters: max_depth = 18, criterion = entropy, score = 0.8490
```

c) Visualization:



d) Caveat: This question was worded somewhat ambiguously so here is my interpretation based on Piazza and TA input (@126 and @131):

- Function parameters should be the training data and label matrices, plus the keyword and criterion used for splitting
- Criterion used for splitting should be the same as what was used in my tree (i.e. Tfidf vectorizer value being above a certain threshold)
- "for several other keywords": I assume that we are always doing a top-level split of the tree, not any conditional splits based on a given first split

Here are my outputs of `compute_information_gain()` for various keywords and `Tfidf` values. These correspond to initial splits in the other two images below.

Note that IG values are slightly off from what you'd calculate from the numbers in the diagram. Thresholds are manually passed to the function, and are likely rounded in the `graphviz` picture.

```
KEYWORD     | Tfidf THRESHOLD | IG
trump       | 0.035           | 0.0264
donald      | 0.071           | 0.0527
hillary     | 0.045           | 0.0353
```