Problem 1

(a) For a sample x to output $Y=0$. x_1, x_2, x_3 has to be equal to 0. There will be 2^{n-3} cases where that is the case. The case where $Y=1$ would then be equal to $2^n - 2^{n-3}$ and since $(2^n - 2^{n-3}) > 2^{n-3}$, $Y=1$ would be the output for any input. Hence, the mistakes would be 2^{n-3}.

$$\frac{2^{n-3}}{2^n} = \frac{\cancel{2^n} \cdot 2^{-3}}{\cancel{2^n}} = 2^{-3} = 1/8 \text{ of the time}$$

(b) No. Even with a split on x_i where $i \geq 4$ will always predict 1 on any branch and thus stay the same.

With a split on x_j s.t. $1 \leq j \leq 3$ will still predict 1 on all branches.

$\therefore$ This will make the same amount of errors as the single-leaf decision tree.

(c) $H[Y] = -P_1 \log_2(P_1) - P_0 \log_2(P_2)$

where $P_1 = \frac{2^{n-3}}{2^n} = \frac{1}{8}$

$P_2 = 1 - \frac{1}{8} = 7/8$

$H[Y] = -\ 7/8 \log_2(7/8) - \frac{1}{8} \log_2(1/8)$

$\simeq 0.543$

(d) Splitting at x_i s.t i is an arbitrary value $1 \leq i \leq 3$.

$H(Y \mid x_i) = 1/2\,(0) + 1/2\left(-3/4 \log_2(3/4) - \frac{1}{4} \log_2(\frac{1}{4})\right)$

$= 0.406.$

Problem 2

(a) $H(S) = B\left(\frac{p}{p+n}\right) = -\frac{p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{p}{p+n}\right)$

min entropy : $H(S)=0$

- This is when $p=0$ or $n=0$ where all examples are either negative or positive.

max entropy : $H(S)=1$

- This is when S is mixed perfectly.
- $H(S)=1$ if $p=n$

General case

- for any p and n, the entropy $H(S)$ will be between 0 and 1 because the function $H(q)$ reaches max at $q=0.5$ (where $p=n$) and decreases as q moves away from 0.5 to either 0 or 1

(b) $H(S) = B\left(\frac{p}{p+n}\right)$

Let $p = \sum_k p_k$ and $n = \sum_k n_k$

$\therefore \frac{p_k}{p_k+n_k} = \frac{p}{p+n}$

Given that

$H(S_k) = B\left(\frac{p_k}{p_k+n_k}\right) = B\left(\frac{p}{p+n}\right)$

$\because$ Information gain is

$$H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k)$$

$$H(S_k) = B\left(\frac{p_k}{p_k+n_k}\right)$$

$$\text{Gain} = H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k) \quad \text{—— (1)}$$

we can see that

$$\sum_k \frac{|S_k|}{|S|} H(S_k) = \sum_k \frac{p_k+n_k}{p+n} \left[B\left(\frac{p}{p+n}\right)\right]$$

$$= \frac{p+n}{p+n} B\left(\frac{p}{p+n}\right)$$

$$= (1) B\left(\frac{p}{p+n}\right)$$

Going back to equation (1)

$$H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k) = B\left(\frac{p}{p+n}\right) - B\left(\frac{p}{p+n}\right)$$

$$\boxed{= 0}$$

$\therefore$ Information gain is 0

Problem 3

(a) $k=1$ will minimize the training set error as it achieves a perfect classification on its own - and classify with itself, thus having a training set error of 0.

The training set error is not a reasonable estimate of test set error especially if $k=1$ is it may lead to overfitting where the model "memorizes" the training data and won't be able to find a general solution.

(b) $\boxed{k=5}$ would minimize the error.
The error would be $\boxed{4/14}$.

Cross-validation is a better measure of test-set performance because each data point is tested on a model trained by all other points. Thus, reducing overfitting.

(c)
- With the lowest k being k=1
- If k=1 the error would be $4/14$ where all the asterisks are labelled as a circle or it could be $5/14$ or $6/16$.

- With the highest k being k=n s.t n=14 (for all 14 points)

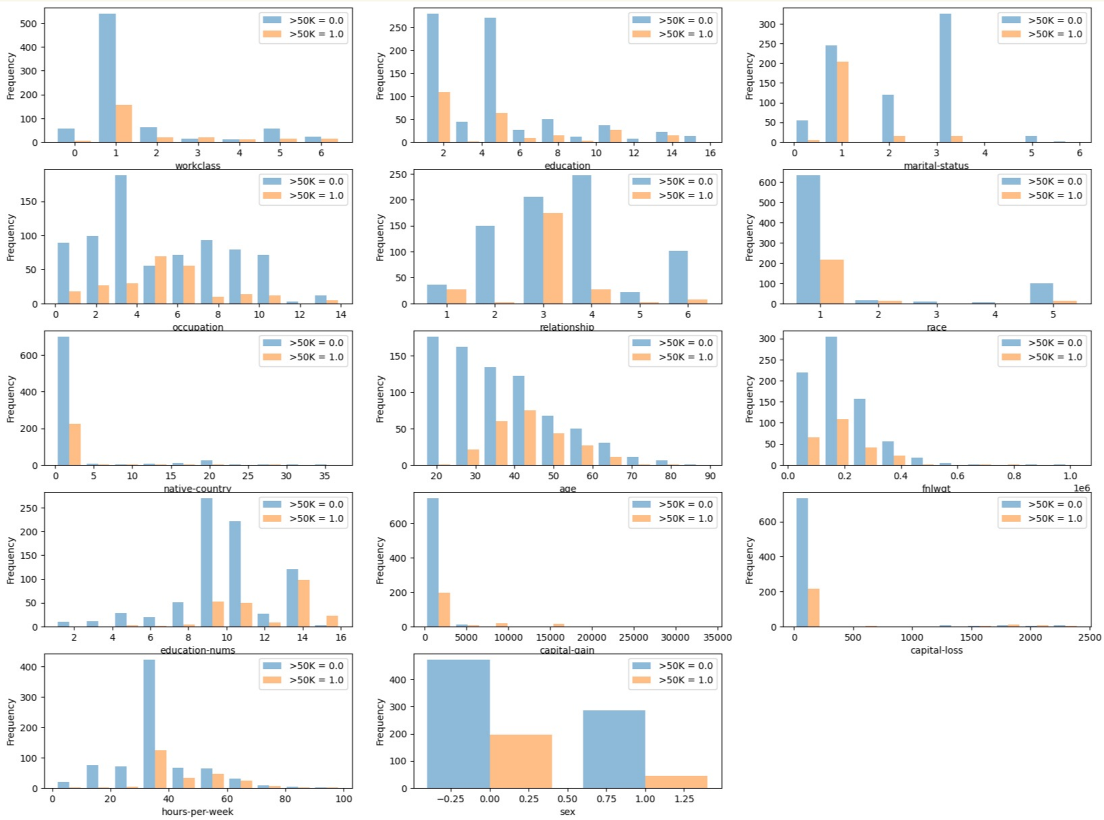
- This means the label or choice will always be the global majority in this case it will be the circles.
  Thus, giving an error of $\frac{4}{14}$

- Too small kvalues like k=1 can cause overfitting which makes the model sensitive.

- Whereas, too large k values over-smooths the points and cause for underfitting.

Problem 4

(a)



(1) Workclass - Most people earn less than 50k where majority of people working in workclass 1 makes less than 50k. People earning >50k are more concentrated in certain work classes.

(2) Education - Higher education levels have a higher number of people that make >50k

(3) Marital-Status - One group of people earn >50k while the others have a majority of making <50k. I believe this to be divorced people.

(4) Occupation - The majority for most of the occupations are people that make <50k. whereas, one occupation has a majority of people making >50k.

(5) Relationship - Majority of all are people making <50k and most people making >50k are concentrated in one category

(6) Race - most races have people making <50k. One race has a majority of >50k earners.

(7) native-country - majority of people are in one native-country, which people making <50k dominant.

(8) Age - Younger people tend to make <50k.
40-50 year olds have the highest probability in making >50k

(9) fnlgwt - Distribution looks similar among all classes.
There are more people making <50k

(10) education-nums - Most people have 8-12 years of education.
most people making >50k are 14-16 years of education.

(11) Capital-gain - One class is dominated by people making <50k where the others are balanced or dominated by people making >50k.

(12) Capital-loss - One class dominates most people and that is dominated by people making <50K.
The rest are balanced.

(13) hours-per-week - Most people work 40-50 hours per week and is dominated by people making <50k.

(14) sex - Both sexes are dominated by people making <50k.

(b)

```
Classifying using Random...
        -- training error: 0.374
```

(c)

```
Classifying using Decision Tree...
        -- training error: 0.000
```

(d)

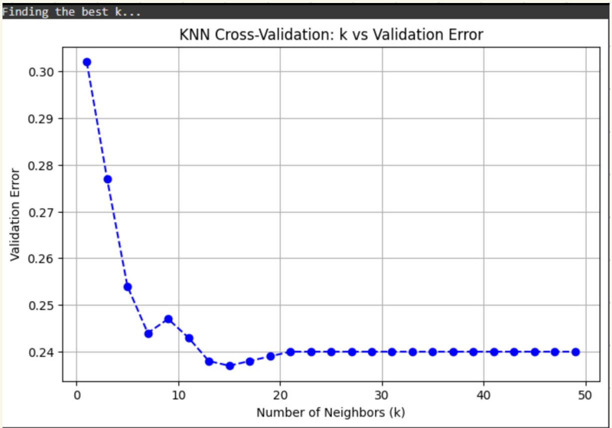
```
Classifying using k-Nearest Neighbors...
        -- training error for k=3: 0.153
        -- training error for k=5: 0.195
        -- training error for k=7: 0.213
```

(e)

```
Investigating various classifiers...
Majority: train error  = 0.240000000000000
Majority: test error   = 0.240000000000000
Majority: F1 score     = 0.760000000000000

Random: train error  = 0.374775000000000
Random: test error   = 0.382000000000000
Random: F1 score     = 0.618000000000000

Decision Tree: train error  = 0.148862500000000
Decision Tree: test error   = 0.182000000000000
Decision Tree: F1 score     = 0.818000000000000

KNN: train error  = 0.201675000000000
KNN: test error   = 0.259150000000000
KNN: F1 score     = 0.740850000000000
```

(f)

```
Finding the best k...
```



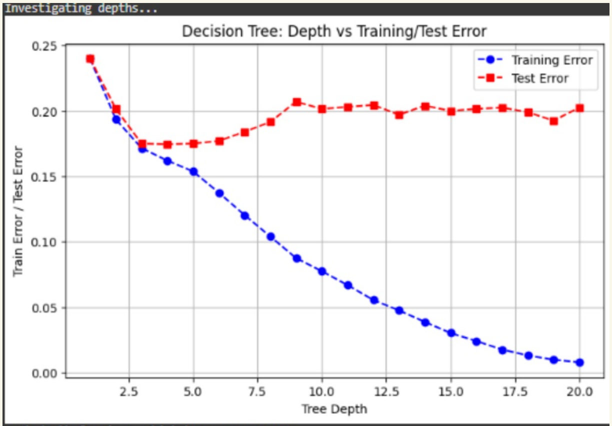

```
Best value of k: 15
```

As k decreased it had a local min, then increased and decreased to the global min of 15, then flattened after that

$\boxed{k=15}$

(g)

```
Investigating depths...
```




```
Best depth found: 4, which has error: 0.175
Depth to Test Error Mapping:
Depth 1: Test Error = 0.240
Depth 2: Test Error = 0.202
Depth 3: Test Error = 0.175
Depth 4: Test Error = 0.175
Depth 5: Test Error = 0.175
Depth 6: Test Error = 0.177
Depth 7: Test Error = 0.184
Depth 8: Test Error = 0.191
Depth 9: Test Error = 0.207
Depth 10: Test Error = 0.202
Depth 11: Test Error = 0.203
Depth 12: Test Error = 0.204
Depth 13: Test Error = 0.197
Depth 14: Test Error = 0.204
Depth 15: Test Error = 0.200
Depth 16: Test Error = 0.202
Depth 17: Test Error = 0.202
Depth 18: Test Error = 0.199
Depth 19: Test Error = 0.193
Depth 20: Test Error = 0.202
```

Best depth is 4, as that is the global min for the test data. with a testing error of 0.175.

(h)

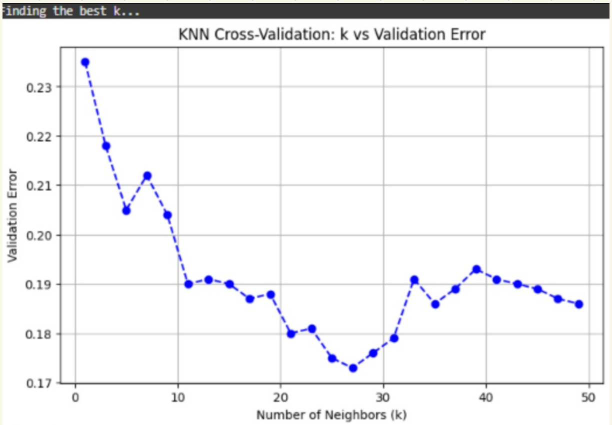
The decision tree shows some slight overfitting but stabalized with more data. Knn maintains a decreasing but more stable test error, which indicates better generalization.

(i)

```
Classifying using Random...
        -- training error: 0.374
Classifying using Decision Tree...
        -- training error: 0.000
Classifying using k-Nearest Neighbors...
        -- training error for k=3: 0.114
        -- training error for k=5: 0.129
        -- training error for k=7: 0.152
Investigating various classifiers...
Majority: train error  = 0.240000000000000
Majority: test error   = 0.240000000000000
Majority: F1 score     = 0.760000000000000

Random: train error  = 0.374775000000000
Random: test error   = 0.382000000000000
Random: F1 score     = 0.618000000000000

Decision Tree: train error  = 0.148862500000000
Decision Tree: test error   = 0.182150000000000
Decision Tree: F1 score     = 0.817850000000000

KNN: train error  = 0.132650000000000
KNN: test error   = 0.209000000000000
KNN: F1 score     = 0.791000000000000
```

• We can see that everything stays the same except KNN's performance increases and training error decreases.

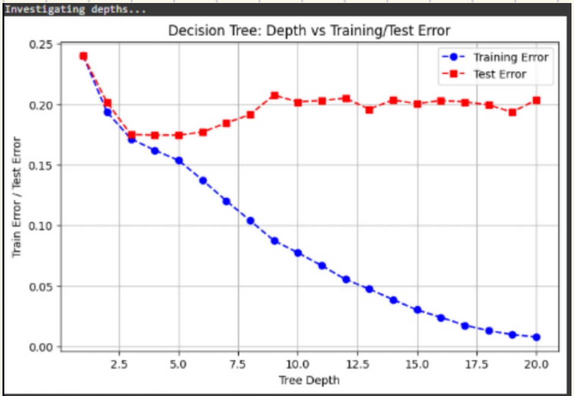
```
Finding the best k...
```




```
Best value of k: 27
```

• The best value for k is now 27 after the normalization happens.

```
Investigating depths...
```




```
Best depth found: 4, which has error: 0.175
Depth to Test Error Mapping:
Depth 1: Test Error = 0.240
Depth 2: Test Error = 0.202
Depth 3: Test Error = 0.175
Depth 4: Test Error = 0.175
Depth 5: Test Error = 0.175
Depth 6: Test Error = 0.177
Depth 7: Test Error = 0.185
Depth 8: Test Error = 0.191
Depth 9: Test Error = 0.208
Depth 10: Test Error = 0.202
Depth 11: Test Error = 0.203
Depth 12: Test Error = 0.205
Depth 13: Test Error = 0.196
Depth 14: Test Error = 0.204
Depth 15: Test Error = 0.200
Depth 16: Test Error = 0.203
Depth 17: Test Error = 0.202
Depth 18: Test Error = 0.200
Depth 19: Test Error = 0.194
Depth 20: Test Error = 0.203
```

- Decision Tree stays the same with best depth being 4.