1. (i) Coin classification system

Since a statistical model was derived to classify the coins, there is no change in the way that the machine classifies the coins, ie. there is no learning happening. The classification is just based on the statistical model and boundaries given by the characteristics of coins.

(ii) Coin classification w/out info about coins

Since the input does have labeled output and the algorithm is determining the criteria, this is an example of supervised learning. The machine learning algorithm tries different hypotheses for which ones correctly classify the coins, since data on the characteristics of the coins is not given.

(iii) Tic-Tac-Toe strategy adjustment

This learning is reinforcement learning, because a reward system is being used to adjust the computer's strategy in each iteration. The output is given a good or bad rating that allows the computer to change strategy accordingly. This is not supervised learning because there is no labeled output data, and it is not unsupervised learning because it is not merely finding patterns in data, but rather by receiving feedback on different strategies.

final answer: [d]

2. Machine learning problems require 3 things:
① there is a pattern
② it cannot be pinned down mathematically
③ we have data on it

i) classifying prime numbers ✗
while there is extensive data on prime numbers, fulfilling ③, it can definitely be determined mathematically classifying primes and non-primes, which contradicts ②. So, this problem is not suited for ML.

ii) fraud in credit cards ✓
There is lots of data on fraudulent credit card charges and it definitely cannot be pinned down mathematically, because it is unfortunately common but does involve many non-numerical factors. There is a pattern for what credit card charges may be fraudulent, so this problem is suited for ML.

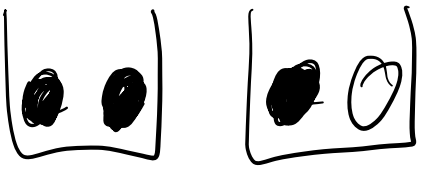
iii) time for object to hit ground ✗
There is lots of data on timing falling objects hitting the ground and there is a clear pattern, but that is because it can be determined mathematically from simple laws of physics. So, this

problem is not well-suited for ML.

iv) optimal cycle of traffic lights ✓

There is lots of traffic data and there is a pattern based on traffic patterns, but this cannot be modelled mathematically. So, this problem fullfills all 3 conditions so it is well-suited for ML.

final answer: [a]

3. 

probability that ball picked is black: $\frac{3}{4}$

For second ball to be black, you had to pick the bag with 2 black balls.

probability that bag w/2 black balls picked: $\frac{1}{2}$

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} \Rightarrow \frac{P(\text{first bag and first ball black})}{P(\text{first ball black})}$$

$$= \frac{\frac{1}{2}}{\frac{3}{4}} = \frac{1}{2} \cdot \frac{4}{3} = \boxed{\frac{2}{3}}$$

final answer: [d]

4. $\mu = 0.55 = P(\text{red})$ $P(\text{green}) = 0.45$

probability of drawing no red marbles when 1 sample is drawn

If 1 sample is drawn, the probability that they are all not red is the probability of choosing a green marble 10 times:

$(0.45)^{10} \approx 3.4 \times 10^{-4} \Rightarrow$ [b]

5. ① The probability of 10 green marbles in one sample: $(0.45)^{10}$ ↳ $\nu = 0$

② The probability of at least 1 green marble: $1-(0.45)^{10}$ ↳ $\nu \neq 0$

③ Probability that all 1000 samples have $\nu \neq 0$: $(1-0.45^{10})^{1000}$

④ Probability that all 1000 samples have $\nu = 0$: $1-(1-0.45^{10})^{1000} = 0.289$

final answer: [C]

6. 8 possible target functions
because there are 3 unknown points, and 2 possible outputs for each point

how many target functions agree/w each hypothesis on # of points

target. func 1: $f(101)=0$ $f(110)=0$ $f(111)=0$

2: $f(101)=0$ $f(110)=0$ $f(111)=1$

3: $f(101)=0$ $f(110)=1$ $f(111)=0$

4: $f(101)=0$ $f(110)=1$ $f(111)=1$

5: $f(101)=1$ $f(110)=0$ $f(111)=0$

6: $f(101)=1$ $f(110)=1$ $f(111)=0$

7: $f(101)=1$ $f(110)=0$ $f(111)=1$

8: $f(101)=1$ $f(110)=1$ $f(111)=1$

hypothesis: returns 1 for all 3 points:

$$1 \cdot 3 + (3) \cdot 2 + (3) \cdot 1 = 12$$

1 t.f. 3 points; 3 t.f. 2 points; 3 t.f. 1 point

hypothesis: returns 0 for all 3 points:

$$1 \cdot 3 + 3 \cdot 2 + 3 \cdot 1 = 12$$

hypothesis: XOR function

t.f. 1: 2 points t.f. 2: 3 points

t.f. 3: 1 point t.f. 4: 2 points

t.f. 5: 1 point t.f 6: 0 points

t.f. 7: 2 points t.f. 8: 1 point

total: $1 \cdot 3 + 3 \cdot 2 + 3 \cdot 1 = 12$

hypothesis: opposite of XOR function

=> 3 - # of points from option [c]

1 point, 0 points, 2 points, 1 point, 2 points,

3 points, 1 point, 2 points

$3 \cdot 1 + 2 \cdot 3 + 3 \cdot 1 = 12$

All 4 hypotheses have the same score, so the final answer is [e].

```
[1] import random
    import numpy as np

    #above the line is + 1 and below is - 1

    #target function generation and classifying function

    def rand_func():
      x1 = random.uniform(-1.0, 1.0)
      x2 = random.uniform(-1.0, 1.0)
      y1 = random.uniform(-1.0, 1.0)
      y2 = random.uniform(-1.0, 1.0)
      m = (y2 - y1) / (x2 - x1)
      b = y1 - m * x1
      return m, b



    def classify_point(point, m, b):
        x, y = point
        if y > m * x + b:
          return 1
        if y == m * x + b:
          return 0
        return -1


    def PLA(weights, X, Y):
      iterations = 0
      while iterations < 10000:
        misclassified = []
        for i, point in enumerate(X):
```

```
    def classify_point(point, m, b):
        x, y = point
        if y > m * x + b:
          return 1
        if y == m * x + b:
          return 0
        return -1


    def PLA(weights, X, Y):
      iterations = 0
      while iterations < 10000:
        misclassified = []
        for i, point in enumerate(X):
          x_vals = np.array([point[0], point[1], 1])
          if np.sign(np.dot(weights, x_vals)) != Y[i]:
            misclassified.append(i)

        #print(misclassified)
        if len(misclassified) == 0:
          break

        #print(misclassified)
        idx = np.random.choice(misclassified)
        x_val1 = np.array([X[idx][0], X[idx][1], 1])
        weights += Y[idx] * x_val1
        iterations += 1
      return iterations, weights
```

```python
N = 10

iterations = 0
max_iter = 10000
weights = np.zeros(3)
weights[2] = 1

disagreement = 0
avg_iter = 0

for _ in range(1000):
  m, b = rand_func()
  X = []
  Y = []
  for _ in range(N):
    x = random.uniform(-1.0, 1.0)
    y = random.uniform(-1.0, 1.0)
    X.append([x, y])
    Y.append(classify_point([x, y], m, b))
  iterations, weights = PLA(weights, X, Y)
  avg_iter += iterations
  point = np.random.uniform(-1, 1, 2)
  f_label = classify_point(point, m, b)
  g_label = np.sign(np.dot(weights, np.array([point[0], point[1], 1])))
  if f_label != g_label:
      disagreement += 1

print("Disagreement for N = 10: ", disagreement / 1000)
print("Average iterations for N = 10: ", avg_iter / 1000)
```

```
Disagreement for N = 10:  0.091
Average iterations for N = 10:  7.025
```

```python
N = 100

iterations = 0
max_iter = 10000
weights = np.zeros(3)
weights[2] = 1

disagreement = 0
avg_iter = 0

for _ in range(1000):
  m, b = rand_func()
  X = []
  Y = []
  for _ in range(N):
    x = random.uniform(-1.0, 1.0)
    y = random.uniform(-1.0, 1.0)
    X.append([x, y])
    Y.append(classify_point([x, y], m, b))
  iterations, weights = PLA(weights, X, Y)
  avg_iter += iterations
  point = np.random.uniform(-1, 1, 2)
  f_label = classify_point(point, m, b)
  g_label = np.sign(np.dot(weights, np.array([point[0], point[1], 1])))
  if f_label != g_label:
      disagreement += 1

print("Disagreement for N = 100: ", disagreement / 1000)
print("Average iterations for N = 100: ", avg_iter / 1000)
```

```
Disagreement for N = 100:  0.02
Average iterations for N = 100:  96.439
```

7. According to my code, it takes ~15 iteration $\Rightarrow$ [b]

↳ screenshots in previous page

8. $P[f(x) \neq g(x)]$ based on random sample of 1000, closest to 0.1 $\Rightarrow$ [c]

9. for $N = 100$, it takes ~100 iterations

[b]

10. $P[f(x) \neq g(x)] \approx 0.01$ based on random sample of 1000 [b]