3 (a)

We denote with $C(h)$ the set of literals in the conjunction $h \in H$.
Let E be the set of all boolean variables which can be used.
We define the refinement operator $\rho : H \to 2^H$ by $\rho(h) = \cup_{e \in E \setminus C(h)} \{e \wedge (\bigwedge_{x \in C(h)} x)\}$.
This operator creates from a given conjunction a set of extended conjunctions that contain one new literal from our set E.

Local finiteness: YES
One can see in the definition of the operator that $|\rho(h)| \leq |E \setminus C(h)|$.

Completeness: YES
We will use induction for showing completeness. In the induction step we want to prove that if all conjunctions of k literals with $k < n$ can be generated by finite operator application, we can also generate all conjunctions with $k' \leq n$ literals for $n \leq |E|$.
In the induction start we set n = 1. The empty conjunction is already given as starting hypothesis. Applying the operator to the empty conjunction gives us the set E as result (all 1-literal-conjunctions). Therefore, we proved the induction step for n=1.
Now we assume that all k-literal-conjunctions with $k < n$ were generated by finite application of the operator. For creating all conjunctions with k' literals for $k' \leq n$ we just have to generate all n-literal-conjunctions since the rest is already given by the assumption. This can be done by applying the operator to all (n-1)-literal-conjunctions because the operator creates from one such conjunction the set of all possible extensions involving one new literal. Since there is a finite number of (n-1)-literal-conjunctions, the operator has to be applied a finite number of times. So we are ready with the induction step now.
So we proved that for $n \leq |E|$ we can create all conjunctions of $\leq n$ literals. For $n = |E|$ we obtain the complete hypothesis space.

Properness: YES
We also have properness. Lets do a proof by contradiction and assume that there are $h, h' \in H$ with $h' \in \rho(h)$ and $h' \equiv h$. This means that h and h' must be conjunctions of the same boolean variables from E. But since $h' \in \rho(h)$ this cannot be the case per definition of the operator, because the operator only adds new variables from E that are not already part of the input conjunction.
There is only one special case: In the case that the operator is applied to a conjunction g using all variables from E (most specific hypothesis), no new values will be added. But since $E \setminus C(g) = \emptyset$, the operator generates an empty set in that case. So even in this case the assumption cannot be true.

Optimality: NO
Lets say we have two boolean variables a and b and let "." denote the empty conjunction:
1. $a, b \in \rho(.)$
2. $a \wedge b \in \rho(a)$ and also $a \wedge b \in \rho(b)$
Therefore, $a \wedge b$ can be created starting at the empty conjunction and then either using a or b. So there are multiple ways to generate the same hypothesis.

3 (b)

We wrote a small Python program to perform the BFS. The program output will be our result. It shows for each hypothesis (on the left side) the value of function q (on the right side, pointed by an arrow). A value of "-1" is chosen by us on purpose as "abnormal value" to indicate that there were no rows for the corresponding hypothesis ($|D(X)| = 0$). The BFS-steps appear from top to bottom in their order of occurence and are enumerated. Furthermore, the "currently" best two solutions can be seen after each update of the solution set. Each solution is a tuple consisting of the hypothesis as first value and the q-value as second value. The resulting solution set can be seen at the end.

The result:

```
0." EMPTY CONJUNCTION " ---> Q: 0.0    //ADD TO SOLUTION SET!
CURRENT SOLUTION SET:{('" EMPTY CONJUNCTION "', 0.0)}
1." a=1 " ---> Q: 0.4618802153517006    //ADD TO SOLUTION SET!
CURRENT SOLUTION SET:{('" EMPTY CONJUNCTION "', 0.0), ('" a=1 "', 0.4618802153517006)}
2." b=1 " ---> Q: 0.11547005383792512    //ADD TO SOLUTION SET! (KICK OUT WORST HYPOTHESIS FROM SOLUTION SET)
CURRENT SOLUTION SET:{('" b=1 "', 0.11547005383792512), ('" a=1 "', 0.4618802153517006)}
3." c=1 " ---> Q: 0.11547005383792512
4." d=1 " ---> Q: 0.11547005383792512
5." a=1  b=1 " ---> Q: 0.6    //ADD TO SOLUTION SET! (KICK OUT WORST HYPOTHESIS FROM SOLUTION SET)
CURRENT SOLUTION SET:{('" a=1  b=1 "', 0.6), ('" a=1 "', 0.4618802153517006)}
6." a=1  c=1 " ---> Q: 0.6    //ADD TO SOLUTION SET! (KICK OUT WORST HYPOTHESIS FROM SOLUTION SET)
CURRENT SOLUTION SET:{('" a=1  b=1 "', 0.6), ('" a=1  c=1 "', 0.6)}
7." a=1  d=1 " ---> Q: 0.14142135623730948
8." b=1  c=1 " ---> Q: 0.5656854249492381
9." b=1  d=1 " ---> Q: 0.14142135623730948
10." c=1  d=1 " ---> Q: 0.4
11." a=1  b=1  c=1 " ---> Q: -1
12." a=1  b=1  d=1 " ---> Q: 0.6
13." a=1  c=1  d=1 " ---> Q: -1
14." b=1  c=1  d=1 " ---> Q: 0.4
15." a=1  b=1  c=1  d=1 " ---> Q: -1
BEST TWO HYPOTHESES:{('" a=1  b=1 "', 0.6), ('" a=1  c=1 "', 0.6)}
```

The exact details of how this output was generated can be found in the Python script below, just in case that this is also required to be submitted for formal completeness of our solution.

```python
from math import *
from itertools import *


"""Set up data from task and define useful functions"""

row1 = {"tid": 1, "a": 1, "b": 0, "c": 0,"d": 1,"label": 1}
row2 = {"tid": 2, "a": 1, "b": 1, "c": 0,"d": 1,"label": 0}
row3 = {"tid": 3, "a": 0, "b": 1, "c": 1,"d": 1,"label": 1}
row4 = {"tid": 4, "a": 1, "b": 0, "c": 1,"d": 0,"label": 0}
row5 = {"tid": 5, "a": 0, "b": 1, "c": 1,"d": 0,"label": 1}

attributes = ["a","b","c","d"]

table = [row1, row2, row3, row4, row5]

"""function D(X) to find all rows matching with query/hypothesis X"""
def D(X): #X = {attr: required val}
    output = []
    for row in table:
        if X.items() <= row.items(): #checks if X is "sub-dictionary" of row
            output.append(row)
    return output

"""function q"""
def q(X):
    DX = D(X)
    X_1 = X.copy()
    X_1["label"] = 1
    DX_1 = D(X_1)
    D_1 = D({"label":1})
    if len(DX) == 0:
        return -1
    return sqrt(len(DX)) * abs( len(DX_1)/len(DX) - len(D_1)/len(table) )


"""helper function to print hypothesis in a better readable way"""
def print_hypothesis(X):
    used_attributes = [a for a in attributes if a in X.keys()]
    output_str = ""
    if len(used_attributes) == 0:
        return "\" EMPTY CONJUNCTION \""
    output_str += "\""
    for a in used_attributes:
        output_str += f" {a}={X[a]} "
    output_str += "\""
    return output_str
```

Figure 1: part 1/2 of script

```python
"""perform and print the BFS search"""

bfs_queries = [{}] #will be list with all queries X for D[X] in required BFS order (without redundant repetition of queries)
for i in [1,2,3,4]:
    for attr_i_subset in list(combinations(attributes , i)): #iterate over all i-subsets of attribute set
        X = {}
        for attr in attr_i_subset:
            X[attr] = 1
        bfs_queries.append(X)


best_two_subgroups = set([])
for (j,X) in enumerate(bfs_queries):
    qX = q(X)
    X = print_hypothesis(X)
    if len(best_two_subgroups)<2:
        print(f"{j}.{X} ---> Q: {qX}    //ADD TO SOLUTION SET!")
        best_two_subgroups.add((X, qX))
        print(f"CURRENT SOLUTION SET:{best_two_subgroups}")
        continue
    worst_solution = min(best_two_subgroups, key=lambda z: z[1])
    if qX > worst_solution[1]:
        print(f"{j}.{X} ---> Q: {qX}    //ADD TO SOLUTION SET! (KICK OUT WORST HYPOTHESIS FROM SOLUTION SET)")
        best_two_subgroups.remove(worst_solution)
        best_two_subgroups.add((X, qX))
        print(f"CURRENT SOLUTION SET:{best_two_subgroups}")
        continue
    print(f"{j}.{X} ---> Q: {qX}")
print(f"BEST TWO HYPOTHESES:{best_two_subgroups}")
```


Figure 2: part 2/2 of script

3 (c)

Here we work with estimates of $Pr(L = +)$. Let p be our estimate of $Pr(L = +)$ for the whole sample set: $p = \frac{|D^+|}{|D|} = \frac{3}{5}$.

Let p_1 be our estimate of $Pr(L = +)$ for the subgroup D[a=1,b=1]. We obtain $p_1 = \frac{|D^+[a=1,b=1]|}{|D[a=1,b=1]|} = 0$.

Let p_2 be our estimate of $Pr(L = +)$ for the subgroup D[a=1,c=1]. We obtain $p_1 = \frac{|D^+[a=1,c=1]|}{|D[a=1,c=1]|} = 0$.

Let $V_1 = \frac{p_1 - p}{\sqrt{p(1-p)/5}} = \frac{-p}{\sqrt{p(1-p)/5}} = -2.7386$

Let $V_2 = \frac{p_2 - p}{\sqrt{p(1-p)/5}} = \frac{-p}{\sqrt{p(1-p)/5}} = -2.7386$

According to the table, the null hypothesis can be rejected since 2.7386 is bigger than 2.58. Therefore, the subgroups can be seen as unusual w.r.t. $\alpha = 0.01$.