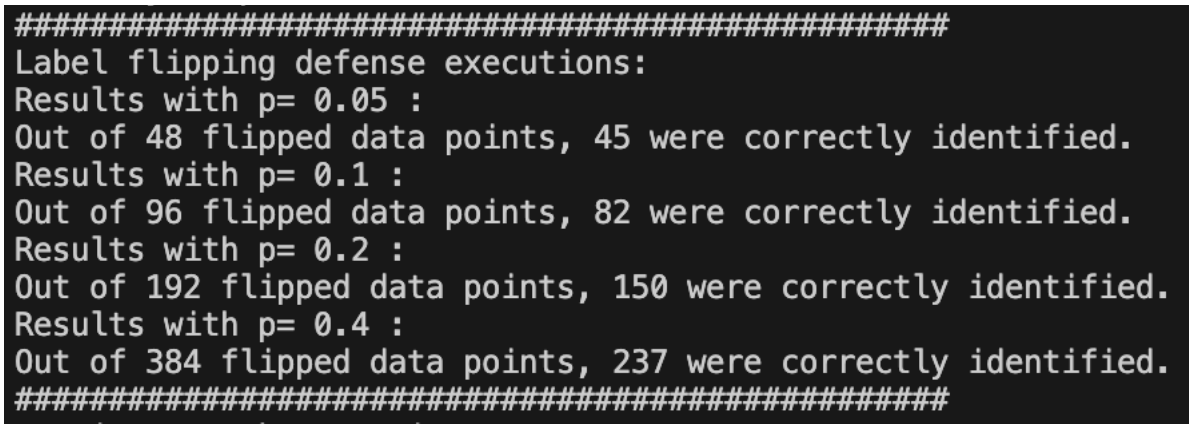
PART1

Question 1

	DT	LR	SVC
0.05	0.9665	0.9785	0.9537
0.1	0.9574	0.9738	0.9485
0.2	0.9313	0.9713	0.9458
0.4	0.7896	0.9557	0.7508

“p” value increase affects all models in negative way. As we can see from table, LR is the most robust model that it stays in high accuracy against the label flipping attack. SVC is the weakest one but near to DT. So, if we make an order robust to weak, it would be “LR>DT>SVC”.

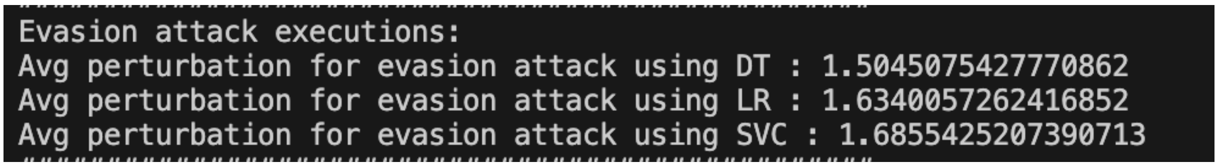
Question 2


```
##################################################
Label flipping defense executions:
Results with p= 0.05 :
Out of 48 flipped data points, 45 were correctly identified.
Results with p= 0.1 :
Out of 96 flipped data points, 82 were correctly identified.
Results with p= 0.2 :
Out of 192 flipped data points, 150 were correctly identified.
Results with p= 0.4 :
Out of 384 flipped data points, 237 were correctly identified.
##################################################
```

(Effectiveness is evaluated by identified/flipped points)	Effectiveness of defense
p=0.05	%94
p=0.1	%85
p=0.2	%78
p=0.4	%62

First, I performed a label flipping attack using the ″p″ value. I combined the flipped version of y_train and X_train into a single dataset and identified points that do not match the local density of their neighbors (outliers) using LOF. I searched for n_neigbors and I found that taking it as “20” is kind a standard and found this by trying, 30 was too large for this dataset. It can be changed according to the complexity and largeness of the dataset. I think my defense is effective for p=0.05 and p=0.1. But for p=0.2 and p=0.4, it could be improved.

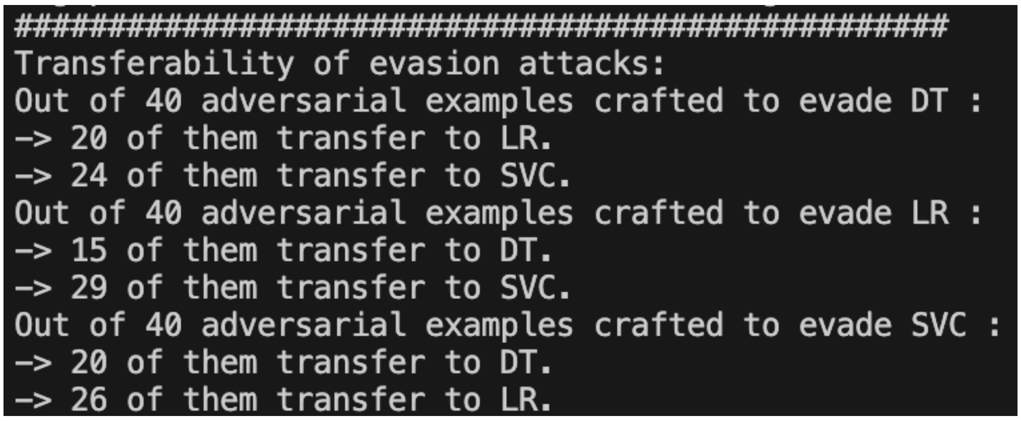
Question 3



```
Evasion attack executions:
Avg perturbation for evasion attack using DT : 1.5045075427770862
Avg perturbation for evasion attack using LR : 1.6340057262416852
Avg perturbation for evasion attack using SVC : 1.6855425207390713
```

In a loop with the condition copy_class == actual_class, (copy_class is used here as predicted class), I randomly selected a feature and took its value randomly from a normal distribution with mean actual feature value and variance (which I set to “1.5”). All perturbation amounts are lower than 3 and successful evasion is achieved for all examples.

Question 4



```
####################################################
Transferability of evasion attacks:
Out of 40 adversarial examples crafted to evade DT :
-> 20 of them transfer to LR.
-> 24 of them transfer to SVC.
Out of 40 adversarial examples crafted to evade LR :
-> 15 of them transfer to DT.
-> 29 of them transfer to SVC.
Out of 40 adversarial examples crafted to evade SVC :
-> 20 of them transfer to DT.
-> 26 of them transfer to LR.
```

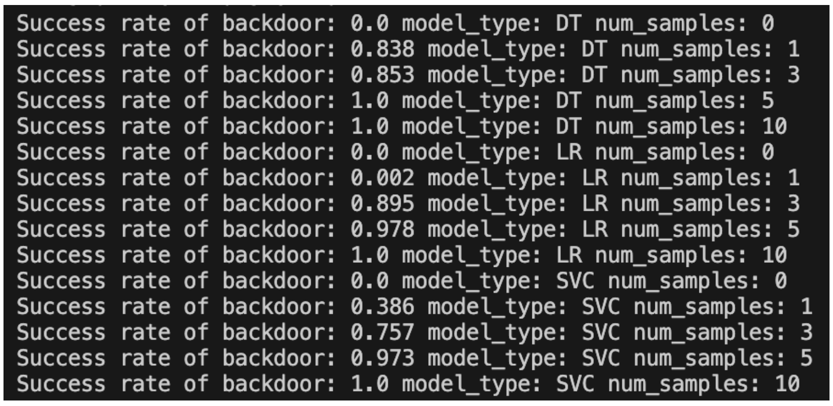
We can comment on this by looking pairs in terms of percentage like:

- DT to LR transferability is %50
- DT to SVC transferability is %60
- LR to DT transferability is %37,5
- LR to SVC transferability is %72,5
- SVC to DT transferability is %65
- SVC to LR transferability is %50

As we can see from these, LR to SRC has most cross-model transferability and LR to DT has least cross-model transferability.

Although these results do not necessarily show that the evasion attack has a high cross-model transferability, they do show that it can be differently effective on different models. In particular, the transferability rates ranging from 50% to 73% indicate that these attacks target common weaknesses.

Question 5



```
Success rate of backdoor: 0.0 model_type: DT num_samples: 0
Success rate of backdoor: 0.838 model_type: DT num_samples: 1
Success rate of backdoor: 0.853 model_type: DT num_samples: 3
Success rate of backdoor: 1.0 model_type: DT num_samples: 5
Success rate of backdoor: 1.0 model_type: DT num_samples: 10
Success rate of backdoor: 0.0 model_type: LR num_samples: 0
Success rate of backdoor: 0.002 model_type: LR num_samples: 1
Success rate of backdoor: 0.895 model_type: LR num_samples: 3
Success rate of backdoor: 0.978 model_type: LR num_samples: 5
Success rate of backdoor: 1.0 model_type: LR num_samples: 10
Success rate of backdoor: 0.0 model_type: SVC num_samples: 0
Success rate of backdoor: 0.386 model_type: SVC num_samples: 1
Success rate of backdoor: 0.757 model_type: SVC num_samples: 3
Success rate of backdoor: 0.973 model_type: SVC num_samples: 5
Success rate of backdoor: 1.0 model_type: SVC num_samples: 10
```

[I also made a table to make it more visible]

	num_samples=0	1	3	5	10
DT	0.0	0.838	0.853	1.0	1.0
LR	0.0	0.002	0.895	0.978	1.0
SVC	0.0	0.386	0.757	0.973	1.0

I found optimal values by experimenting to create backdoor examples and defined these fixed values as follows: TRIGGER_FLAG refers to the fixed value that some features of a backdoor example will take, and I fixed it to 1000. N_FLAGGED_FEATURES indicates the number of features that will have a trigger flag. I marked the first two features to be able to make a secret but still effective attack. I generated the other features that were not triggered from a normal distribution with a variance of 5.0 determined by the RAND_NOISE_VAR parameter. I also fixed the number of samples I will use during the test as N_TEST_SAMPLES to 1000. The trigger pattern in my attack is by setting the first two features to a fixed value and generating the other features from random noise.

To measure the success rate of the experiment, labels for the 1000 backdoor examples I created were predicted using the previously trained backdoored model. I measured the percentage of backdoor examples that the backdoor model predicted would have 1 label as the success rate. Success Rate = (Number of backdoor instances predicted by the backdoor model as 1) / (Total number of backdoored instances)

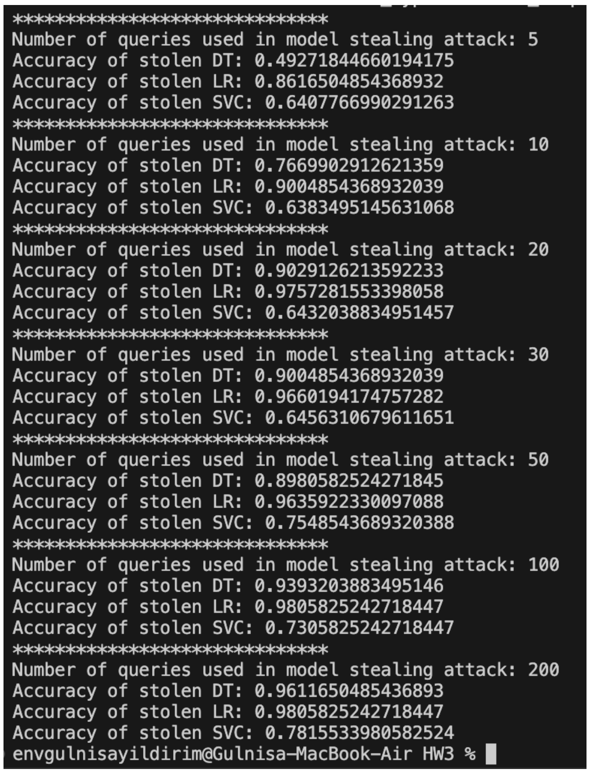
Additionally, according to my experiment results [I run my code for few times it always like that]:

For num_samples=1   most robust one=LR

For num_samples=3   most robust one=SVC

For num_samples=5   most robust one=SVC

Question 6



```
******************************
Number of queries used in model stealing attack: 5
Accuracy of stolen DT: 0.49271844660194175
Accuracy of stolen LR: 0.8616504854368932
Accuracy of stolen SVC: 0.6407766990291263
******************************
Number of queries used in model stealing attack: 10
Accuracy of stolen DT: 0.7669902912621359
Accuracy of stolen LR: 0.9004854368932039
Accuracy of stolen SVC: 0.6383495145631068
******************************
Number of queries used in model stealing attack: 20
Accuracy of stolen DT: 0.9029126213592233
Accuracy of stolen LR: 0.9757281553398058
Accuracy of stolen SVC: 0.6432038834951457
******************************
Number of queries used in model stealing attack: 30
Accuracy of stolen DT: 0.9004854368932039
Accuracy of stolen LR: 0.9660194174757282
Accuracy of stolen SVC: 0.6456310679611651
******************************
Number of queries used in model stealing attack: 50
Accuracy of stolen DT: 0.8980582524271845
Accuracy of stolen LR: 0.9635922330097088
Accuracy of stolen SVC: 0.7548543689320388
******************************
Number of queries used in model stealing attack: 100
Accuracy of stolen DT: 0.9393203883495146
Accuracy of stolen LR: 0.9805825242718447
Accuracy of stolen SVC: 0.7305825242718447
******************************
Number of queries used in model stealing attack: 200
Accuracy of stolen DT: 0.9611650485436893
Accuracy of stolen LR: 0.9805825242718447
Accuracy of stolen SVC: 0.7815533980582524
envgulnisayildirim@Gulnisa-MacBook-Air HW3 %
```

	num_queries=5	10	20	30	50	100	200
DT	0.493	0.767	0.903	0.900	0.898	0.939	0.961
LR	0.862	0.900	0.976	0.966	0.964	0.981	0.981
SVC	0.641	0.638	0.643	0.646	0.755	0.731	0.782

As we can see from table, LR is the easiest model to steal. Even number of queries equal to 5, it can be stolen with high accuracy (0.862). After LR, DT is the second easiest one to steal. For num_queries=5 it can be seen like DT has lower accuracy than SVC, so SVC is the second easiest, but this is not the case. As num_queries value increases DT has always higher accuracy rather than SVC so that is why DT is the second model. And lastly, SVC is the hardest model to steal, it does not even reach 0.8 accuracy as number of queries increases. I mean if we compare them in num_queries=200 column it is obvious to state that SVC is the hardest one to steal.

PART2

a) SpamBayes' learning method assumes that the spam scores of words are independent of each other. This independence can be clearly observed in Equation (2), since the spam score f(w) of each word is based on the frequency of occurrence of that word in spam emails and raw emails. In Equations, the overall spam score of an email is calculated by combining the individual spam scores of the words using the Fisher method. In this process, adding a word has no effect on the spam scores of other words. This mathematical model allows each word to be treated as a separate unit and reinforces the principle of independence. As a result, the authors reached this observation by examining the equations.

b) Attackers consider that the spam scores of words are independent of each other, allowing each word to be optimized as a separate target. During dictionary attacks, a large pool of words is created, and these words are marked as spam, which results in filters being fooled. On the other hand, in targeted attacks, words that are suitable for a specific target email content are selected and their spam scores are increased. This strategy allows the attacker to act effectively even if the attacker has limited information.

c) Dynamic Threshold defense provides effective protection against attacks by using flexibly adjustable thresholds instead of fixed threshold values.

In this method, the training data is divided into two parts: one part is used for training the model, and the other part is used to determine the appropriate threshold values. These thresholds play a critical role in distinguishing spam and raw emails and are optimized by an auxiliary function called g(t). The advantage of dynamic thresholds is to minimize misclassifications by balancing the overall shifts in word scores during attacks. This method provides effective protection, especially in attacks where all scores increase or decrease, since the rankings of the messages are not affected by these shifts. As a result, spam and raw messages are classified more accurately, while the impact of the attack is reduced.

d) The outlier detection we learned in class can be used as an effective method to identify and reduce the damage done by attacks to the training set. As stated in the article, the fake emails added to the training set by the attackers usually show significant deviations from the training data. Although these emails are few compared to the total messages, the number of tokens they contain is quite high. In addition, the word distribution of the attack messages can be very different compared to the normal email distribution. Such anomalies can be detected by outlier detection algorithms and the negative effects of the attack can be reduced by discarding the attack data from the training set. Sudden increases in the spam scores of the target words can be easily detected by outlier detection.