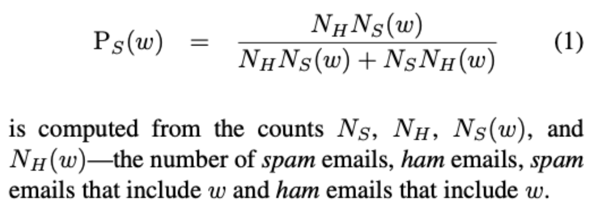
COMP430 DATA PRIVACY AND SECURITY PROJECT3 REPORT, FALL-2022

Name – Surname: Barış KAPLAN
KU ID Number: 0069054 (69054)
KU Email Address: bkaplan18@ku.edu.tr
KU Login Name: bkaplan18
Homework Number: Homework #3
Course Code: COMP430
Course Name: Data Privacy and Security

Part1 (Reading Part) Answers

a-) The authors arrive at this observation by defining, understanding, and analyzing the assumption of the method of Robinson. In the Section 2.3, the assumption of the method of Robinsion states that the spam score of each token in an email is independent of the absence or presence situations of the other words which exist in that email. The assumption of the Robinson is supported by the Fisher's and Graham's works about dealing with the spam-filtering process and determining/evaluating the significance level of each token in an email. While determining/evaluating the significance level of each token in an email, Graham and Fisher applied several significance tests which are independent of each other to the each token which exist in an email. In the Section 2.3 which explains the learning method called "SpamBayes", the authors further supported the independence of the spam scores of the email tokens by providing the below equation in the Page 2 Section 2.3 of the article named "Exploiting Machine Learning to Subvert Your Spam Filter" (Page 2):



$$\mathrm{P}_S(w) \quad = \quad \frac{N_H N_S(w)}{N_H N_S(w) + N_S N_H(w)} \qquad (1)$$

is computed from the counts N_S, N_H, $N_S(w)$, and $N_H(w)$—the number of *spam* emails, *ham* emails, *spam* emails that include w and *ham* emails that include w.

Figure-1: The proposed formula which further supports the independence of the spam scores of tokens (Screenshot taken from the Page2 Section2.3 of the article named "Exploiting Machine Learning to Subvert Your Spam Filter").

Explanation about the above equation: From the above equation, we can understand/infer that the probability of including the specific word "w" is based on the total number of ham emails which involve the provided word w, the total number of the spam emails which contain the given token w, the total number of the spam emails, and the total number of the ham emails. We can conclude from this situation that the probability of observing a specific token w is dependent only on the provided word w. We can also conclude that this probability is not conditioned or dependent on the other words/tokens which exist in an email (the above formula does not include any conditional probabilities or dependence to the words other than w).

b-) The attacks mentioned in this text are included under the attack category of "Causative Availability". The "Causative Availability" attacks primarily aims/tends to change the training data of the spam filter and increase the total number of false positive spam-filtering attempts. This article states two types of "Causative Availability" attacks which are "Dictionary Attacks" and "Focused Attacks". In the "Dictionary Attack", the main goal is to increase the number of false positive classifications of the spam filters. In other words, the main goal of the dictionary attacks applied to the spam filters is to increase the number of legitimate emails which are classified by the spam filters incorrectly as spam emails. In the "Focused Attacks", the goal is not to disrupt the true functionality of the spam filter but to block the victims from obtaining specific legitimate emails. The above observation is used by combining the common ideas of the "Dictionary Attacks" and "Focused Attacks", and by defining and explaining the best-performing/optimal/ideal attacks under the following maximization purpose of the attacker:

- ⇨ **Attack Email =** a => **subsequent legitimate email =** m
- ⇨ **Expected Spam Score of an Email =** Ia => **distribution =** p

The main goal of this maximization purpose of the attacker is to maximize the expected spam score of the emails which are sent to victims while performing a specific attack.

Mathematical Representation of this Maximization Purpose:

As the author of the provided “Exploiting Machine Learning to Subvert Your Spam Filter” article says;

$\max_{a} E_{m\sim p} [I_{a} (m)]$

(Page 4, formula taken from the Page 4 of the provided “Exploiting Machine Learning to Subvert Your Spam Filter” article).

c-) By using the collected data instead of the fixed/static/constant threshold selections (picks) of the SpamBayes, the defense called “Dynamic Threshold” dynamically adjusts the classification thresholds of the spam filter. By using dynamic and adaptive thresholds, the Dynamic Threshold defense technique decreases/reduces the effectiveness, performance, and consequences of the attacks (or prevents the attacks) which try to change all of the spam scores, such as distribution-based attacks. Because, with the usage of the adaptive and dynamic thresholds, the rankings which are assigned by the Dynamic Threshold defense are not affected by these attempts of changing spam scores (are resistant/robust/invariant to the attempts of changing spam scores). Here, the “rankings” mean (refer to) the activity or procedure of assigning the scores/ranks to the activities coming to a software/system by the Dynamic Threshold defense technique according to the probability that whether these activities are adversarial/malicious. The Dynamic Threshold defense is effective since it can easily adapt to the constantly/regularly varying malicious/adversarial activities by using a mechanism in which the classification thresholds of the spam filter dynamically change. This adaptation strategy brings “up-to-dateness” to the Dynamic Threshold defense about understanding & analyzing the recent attack techniques, and taking suitable measures against those attacks. In addition, the “Dynamic Threshold” efense technique reduces the consequences of the dictionary attacks. By considering these, we can say that the Dynamic Threshold defense is effective in preventing several attacks such as “Distribution-based attacks” “Focused Attacks”, and “Dictionary Attacks”, and in making the softwares/systems more robust to (protecting the systems/softwares against) the adversarial processes/activies like spam.

d-) The authors admit that the feature (or aspect) of “the presence and/or absence of some specific word or word group (phrase) instances within the attack payload” make it possible to identify that specific attack. For detecting the dictionary attacks which are discussed in this paper, the outlier detection defense mechanism can be used for detecting, determining and analyzing the incoming emails which involve an abundance (lot) of words from the list of phrases and/or words which was predefined and commonly used by the attacker. After the outlier detection defense detects these kinds of emails, it can mark those emails as outliers and high probably spam/malicious. This outlier detection technique can also be used for the email words/phrases which are neither in the list of attacker nor in the usual/typical legitimate emails. For defending against the focused attacks, the outlier detection defense mechanism can be initially used for determining and analyzing the emails which continuously come from the attacker side to the victims and which contain unusual/strange words and/or phrases that are not commonly used within the legitimate emails. After the outlier detection defense detects these kinds of emails, it can classify those emails as “malicious/adversarial” and can take suitable action which will be applied to those emails.

e-) This optimization problem is trying to achieve the minimization of the possibility/probability of classifying the recently inputted malware as malicious. This minimization process performed after applying the alteration (transformation) process to the each of the inputted malware. However, while trying to achieve this minimization purpose, this optimization problem also tries to keep the total number of the bytes injected to the input malware below of a certain level (“it does not enable injecting a lot of bytes to inputs”). The process of minimizatiion is applied along all possible transormation instances which are valid. Furthermore, in this optimization problem, the total number of queries which can be sent to the targeted specific machine learning (ML) model is confined by the total budget of query. Here, q represents the total number of queries which can be sent to the targeted specific machine learning (ML) model. Moreover, T represents the total amount of the query budget.

f-) It is meant by the “query-efficiency” concept of the attacks in this paper that the attacks are based on the injection of the content which specifically aims (tends) to simplify/ease the process of (accomplish/achieve) evasion. While simplifying the process of evasion, the contents to be inserted/injected by the “query-efficient” attacks are obtained/taken from the safe/benign instances (samples) rather than from a randomly created distribution. Therefore, for the “query-efficient” attacks, we can conclude/infer that overcoming the cyberthreat detection models and evading from them are easier and take less time compared to the attacks which are not “query-efficient”. As a result of this, the “query-efficient” attack brings lower rates of the detections of cyberthreats such as malware or virus. By considering all these situations into account (which I mentioned in this question), we can clearly conclude that the “query-efficiency” property is desired and significant for the attackers. In this paper, it is meant by the “functionality-preserving” concept of the attacks that when the malicious (adversarial) softwares/programs are changed by applying a set of different manipulations, than the altered software/program will have exactly the same functionality with the initial (original) program/software. Moreover, according to “the functionality-preserving” property presented in this paper, the set of different manipulations which are applied to the malicious programs only deals with inserting content into these malicious programs rather than changing the functionalities and the executions of them. In this paper, this process of injecting/inserting the content is performed by specifically trying/aiming to simplify and accomplish the evasion. Therefore, we can conclude that the attacks which are “functionality-preserving” are more effective and efficient in comparison to the attacks which are not “functionality-preserving”. The attacks which are not “functionality-preserving” might have to run its malicious cyberthreats within a safe environment called “sandbox”. For the attacks which are not “functionality-preserving”, this situation will make the corresponding attack save its internal functionalities. However, this will also bring more costs to these attackers (attackers who apply the attacks which are “not functionality-preserving”). Therefore, by considering the individual purposes and the cost perspective, I can conclude that the “functionality-preserving” property is desired and important for the attackkers.

g-) In this paper which is named as “Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, the number of queries mean the total number of queries which were sent/asked to the targeted machine learning (ML) model in order to find if the inputs of this targeted machine learning model are “Not Malicious (Safe)” or “Malicious/Advesarial”. The number of queries can also be defined as the total number/amount of times the targeted machine learning model was sent queries (or namely queried). Moreover, the attack size equals to the total number/amount of the bytes which were inserted into the malware cyberthreats which were inputted to the targeted machine learning models to be infected. In addition to that, the detection rate equals to the rate which indicates the total number of true positive classifications **(the samples which are malicious are correctly classified under the malicious category)** of the adversarial/malicious samples (by ML models) divided by the total number of number of samples (Total number of samples include true positives, true negatives, false positives, and false negatives). I would expect that whereas the number of queries increases, the detection rate decreases within the context of the attacks of this paper. In other words, I would expect that the detection rate is **inversely proportional** to the number of queries. Furthermore, I would expect that whereas the attack size increases, the detection rate decreases within the context of the attacks of this paper. In other words, I would expect that the detection rate is **inversely proportional** to the attack size. In this paper which is named as “Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, the authors presented that the increasing the number of queries makes the attack brings the attack payload size and the misclassification confidence at optimal levels (makes the attack balance the attack payload size and misclassification confidence) in better ways (“Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, Page 4). However, higher number/amount of queries brings a cost of higher amount of time complexity. The author also proposes the significant influence of the total number of queries in the context of a genetic optimizer called GAMMA (“Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, Page 3). In this context, it is discussed that the increasing number of queries bring malicious instances with better attack payload size and better malware detection ratio. Moreover, it is also mentioned that the GAMMA can detect higher number of malicious/adversarial solutions which are simultaneously evasive and confidential by using/sending higher number of queries(“Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, Page 3). However, in the context of this paper, it is discussed that finding these kinds of solutions cannot be found soon in the process of adversarial solution seeking which is performed by the GAMMA. These events/actions/situations mentioned in the article significantly supported my intitutions about this question.

h-) Yes, the attacks which are applied on the commercial antivirus software leverage the transferability concept. In this paper which is named as “Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, it is mentioned that the attacks applied on the commercial antivirus softwares could be able to evade to the more than 12 antivirus engines which are commercial (“Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware”, Page 1). Moreover, it is stated by the authors in this text that for each malware instance/sample, the section-injection attack can bypass to more than 12 virus detector software. Due to the fact that the section-injection attack can bypass to more than 12 virus detector software, we can state that the attacks on the commercial antiviruus software leverage the concept of transferability. Because, the definition of the transferability concept states that a cyberattack which is successful on a machine learning (ML) model is highly likely to be successful on the other similar machine learning (ML) models. From this paper, I can give "GDBT Classifier Model" as an example to the machine learning (ML) model on which the section-injection attack was successfully applied. By taking these situations into account, we can conclude that the section-injection attack that was successfully to the "GDBT Classifier Model" was also able to bypass to more than 11 other virus detector software. Therefore, at the end, we can conclude that the attacks (section-injection attacks) on commercial antivirus software leverage the concept of transferability.

Note: More supporting information about the part h of the reading part of this homework can be found in the Page 8 of this paper which is named as "Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware".

Part2 Discussion About Implementation

Question-1)

Label Flipping Experiment Accuracy Results

	n = 0.05	**n = 0.10**	**n = 0.20**	**n = 0.40**
DT	0.9661224489795921	0.9453061224489794	0.9114285714285715	0.8222448979591834
LR	0.9665306122448978	0.9536734693877553	0.9173469387755099	0.8699999999999999
SVC	0.991428571428572	0.988571428571429	0.9763265306122452	0.9379591836734696

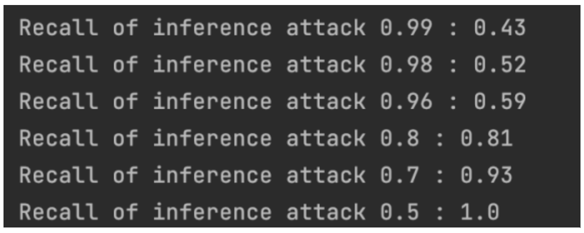
Interpretation and Discussion of the Results: For all of the provided machine learning model types (DL, LR, SVC), I have observed that when the value of n increases, the value of accuracy decreases. In other words, I have observed that the value of accuracy is **inversely proportional** to the value of n. The label flipping attack does not equally affect all of the provided machine learning model types (DL, LR, SVC). For all provided n values (0.05, 0.10, 0.20, 0.40), I have observed that the accuracy value of SVC model is higher than the accuracy values of DT and LR. Since the accuracy value of SVC model is the highest for each n value, I can say that SVC model is more robust to the label flipping attack in comparison to the other provided machine learning models (DL and LR).

NOTE: Please see the above table for the supporting information.

Question-2)

Recall Values where t is in the list [0.99, 0.98, 0.96, 0.8, 0.7, 0.5]

t values	**t = 0.99**	**t = 0.98**	**t = 0.96**	**t = 0.80**	**t = 0.70**	**t = 0.50**
Recall values	0.43	0.52	0.59	0.81	0.93	1.0



```
Recall of inference attack 0.99 : 0.43
Recall of inference attack 0.98 : 0.52
Recall of inference attack 0.96 : 0.59
Recall of inference attack 0.8 : 0.81
Recall of inference attack 0.7 : 0.93
Recall of inference attack 0.5 : 1.0
```

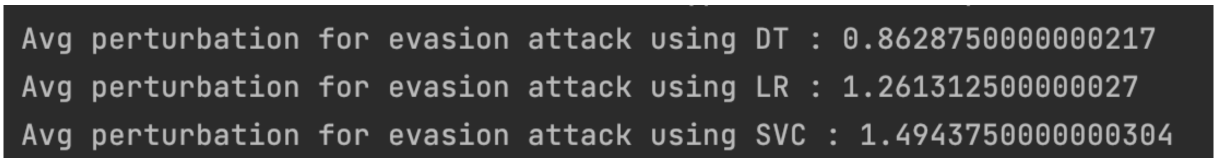
All outputs of my membership inference attack

2a) The recall for a membership inference attack is the proportion (ratio) of the total number of samples that are correctly inferred by the membership inference attack as elements (members) of the training dataset to the total number of correct inference attempts performed by the membership inference attack (Here, the total number of correct inference attempts can include both true positive **(For true positive inference attempts, the membership inference attack correctly infers the samples that these samples are included as elements in the training dataset)** inference attempts and false negative **(For false negative inference attempts, the membership inference attack incorrectly infers (comes to a conclusion) about the samples that these samples are not included as elements in the training dataset)** inference attempts). Here, the total number of samples that are correctly inferred by the membership inference attack as elements (members) of the training dataset refers to **the total number of true positive inferences.** However, the total number of samples that are incorrectly inferred by the membership inference attack as elements (members) of the training dataset refers to the **total number of the false positive inferences.** Therefore, we can also say that recall refers to the total number of the true positive inferences over the summation of the total number of false negative inferences and the total number of true positive inferences. In short, we can also use the following to represent the recall:

Recall = (total number of true positive inferences) / (total number of true positive inferences + total number of false negative inferences)

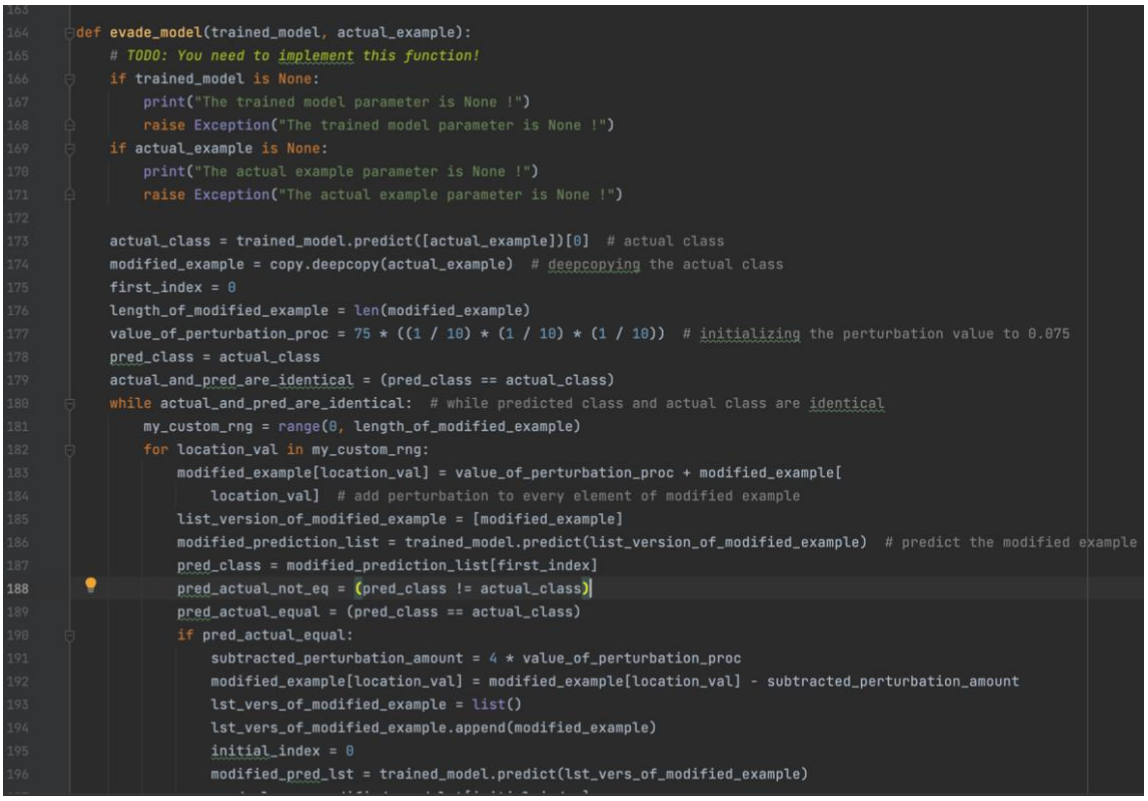
2b) I have observed that when the value of t decreases **(starting from t = 0.99 and ending at t = 0.50, see my below table and screenshot for observing all t values)**, the value of recall of the membership inference attack increases. In other words, I have observed that the value of recall of the membership inference attack is **inversely proportional** to the value of t. I have observed such a trend in the recall values. Because, firstly, the threshold value (the value of t) determines the amounts (or proportions) of the incorrectly inferred/identified/classified elements (members) and the correctly inferred/identified/classified elements (members). The membership inference attack classifies the members/elements which exceed the threshold value as true positives and the members/elements which do not exceed the threshold value (which are equal to or less than the threshold value) as false negatives. By considering this classification strategy, if the threshold value (value of t) is low (if the threshold value decreases), then it is more likely that the membership inference attack correctly classifies members/elements as the real elements/members of the training dataset. Moreover, if the threshold value (value of t) is low (if the threshold value decreases), it is less likely that the membership inference attack incorrectly classifies members/elements as the actual elements/members of the training dataset. Therefore, by considering these, we can say that the number of true positives tends to be higher when the threshold value is lower. In addition, we can say that the number of false negatives tends to be lower when the threshold value is lower. Consequently, higher number of true positives and lower number of false negatives will bring higher recall values. Therefore, at the end, I derived an increasing trend of the recall values while the threshold values are decreasing (starting from threshold = 0.99 and ending at threshold = 0.50).

<u>Question-4)</u>



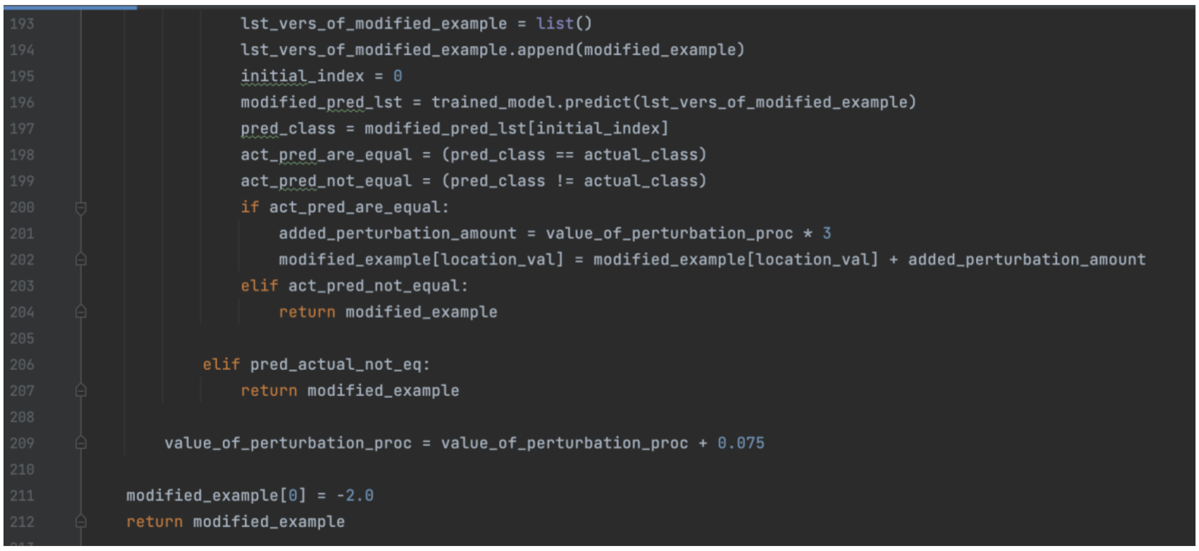
```
Avg perturbation for evasion attack using DT : 0.8628750000000217
Avg perturbation for evasion attack using LR : 1.261312500000027
Avg perturbation for evasion attack using SVC : 1.4943750000000304
```

The average perturbation amounts for all provided machine learning models which are DT, LR, and SVC.



```python
def evade_model(trained_model, actual_example):
    # TODO: You need to implement this function!
    if trained_model is None:
        print("The trained model parameter is None !")
        raise Exception("The trained model parameter is None !")
    if actual_example is None:
        print("The actual example parameter is None !")
        raise Exception("The actual example parameter is None !")

    actual_class = trained_model.predict([actual_example])[0]  # actual class
    modified_example = copy.deepcopy(actual_example)  # deepcopying the actual class
    first_index = 0
    length_of_modified_example = len(modified_example)
    value_of_perturbation_proc = 75 * ((1 / 10) * (1 / 10) * (1 / 10))  # initializing the perturbation value to 0.075
    pred_class = actual_class
    actual_and_pred_are_identical = (pred_class == actual_class)
    while actual_and_pred_are_identical:  # while predicted class and actual class are identical
        my_custom_rng = range(0, length_of_modified_example)
        for location_val in my_custom_rng:
            modified_example[location_val] = value_of_perturbation_proc + modified_example[
                location_val]  # add perturbation to every element of modified example
            list_version_of_modified_example = [modified_example]
            modified_prediction_list = trained_model.predict(list_version_of_modified_example)  # predict the modified example
            pred_class = modified_prediction_list[first_index]
            pred_actual_not_eq = (pred_class != actual_class)
            pred_actual_equal = (pred_class == actual_class)
            if pred_actual_equal:
                subtracted_perturbation_amount = 4 * value_of_perturbation_proc
                modified_example[location_val] = modified_example[location_val] - subtracted_perturbation_amount
                lst_vers_of_modified_example = list()
                lst_vers_of_modified_example.append(modified_example)
                initial_index = 0
                modified_pred_lst = trained_model.predict(lst_vers_of_modified_example)
```

My Evasion Attack First Screenshot



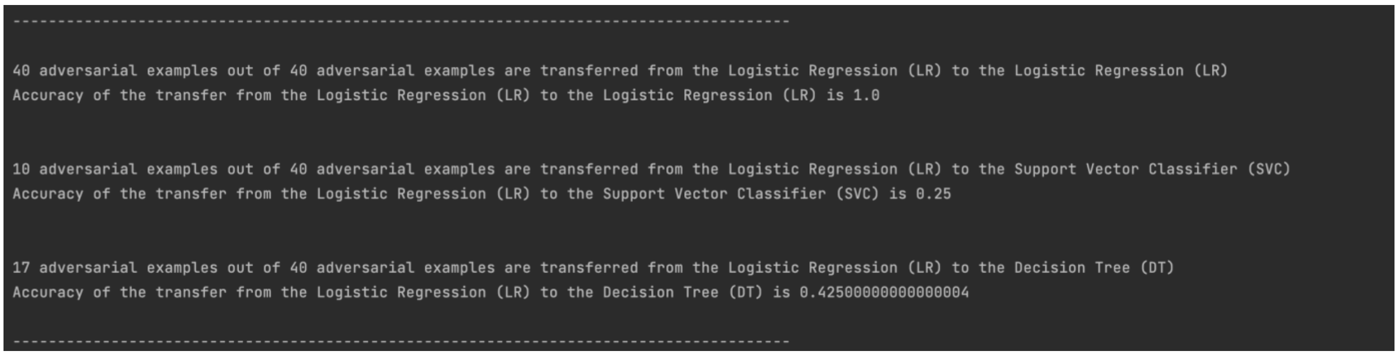
```python
                lst_vers_of_modified_example = list()
                lst_vers_of_modified_example.append(modified_example)
                initial_index = 0
                modified_pred_lst = trained_model.predict(lst_vers_of_modified_example)
                pred_class = modified_pred_lst[initial_index]
                act_pred_are_equal = (pred_class == actual_class)
                act_pred_not_equal = (pred_class != actual_class)
                if act_pred_are_equal:
                    added_perturbation_amount = value_of_perturbation_proc * 3
                    modified_example[location_val] = modified_example[location_val] + added_perturbation_amount
                elif act_pred_not_equal:
                    return modified_example

            elif pred_actual_not_eq:
                return modified_example

        value_of_perturbation_proc = value_of_perturbation_proc + 0.075

    modified_example[0] = -2.0
    return modified_example
```

My Evasion Attack Second Screenshot

My Evasion Attack Strategy: Initially, I have dealed with the cases where the trained_model parameter or the actual_example parameter is None (For handling all possible test scenarios). Then, I have predicted the actual example by calling predict() function with the trained_model. After I predicted the actual example, I obtained the actual class. Next, I have deepcopied the actual example and obtained the modified example. After that, I have initialized the perturbation value to 0.075. I have selected this amount such that the evade_model() function modifies the actual example enough and always finds an adversarial example. At this step, before the while loop, I have initialized the predicted class to the actual class. Then, as it is already provided to us in the hints, I have iterated a while loop whenever the actual class is equal to the predicted class. Inside this while loop, I have gone through all of the index-based locations by using a for loop. After that, inside the for loop, I have added the perturbation amount to each of the members/elements in the modified example list. Subsequently, I have predicted the modified example by calling/using the predict() function with the trained model. Then, I have assigned the single existing element of the prediction of modified example to the predicted class. At this step, I have checked whether the predicted class is equal to the actual class. If the predicted class is not equal to the actual class **(corresponds to one of the outer if statements)**, then this means that the specified ML model (DT, LR or SVC) has found an adverserial example (modified example) which misleads the specified ML model. This situation means that the evasion attack has reached its final goal/aim. Therefore, when the predicted class is not equal to the actual class, I have returned the found modified example. If the predicted class is equal to the actual class **(corresponds to one of the outer if statements)**, I have added a specific multiple of the perturbation amount to the each element of the modified example(i.e., I have modified each element in the modified_example by a specific multiple of the perturbation amount). While determining this multiple (coefficient), I have used a trial & error approach (experimentation approach). Then, I have predicted the modified example by using/calling the predict() function with the trained model. At this step, I have updated/assigned the predicted class (pred_class) to the first element (eg: prediction_result[0]) of the result of this prediction. Next; inside the if statement which checks the equality of actual class and predicted class, since the actual class can be equal to the predicted class due to the modifications applied to the modified example, I have again checked whether the actual class is equal to the predicted class. At this step, if the actual class is equal to the predicted class, then I have added a specific multiple of the perturbation amount to the each of the elements/members in the modified_example (i.e., I have modified each element in the modified_example by a specific multiple of the perturbation amount). While determining this multiple (coefficient), I have used a trial & error approach (experimentation approach). However, if the actual class is not equal to the predicted class, then it means that my evasion attack has founded an adversarial example (modified example) which misleads the specified ML model (LR, DT or SVC). This situation means that the evasion attack has reached its ultimate aim. Consequently; within the if statement checking the equality of predicted class and actual class, if the actual class is not equal to the predicted class, I have returned the found modified example. Inside the while loop checking the equality of the actual class and the predicted class, I have gradually increased the value of the perturbation amount with an amount that I have determined with experimentation (trial and error). While applying this experimentation (trial and error) approach, I have paid attention to **avoid** from the **false positive type of evasions.** In other words, I have determined the amount of increase in the perturbation value such that the perturbation value does not cause (or causes almost no) false positive type **(false positive type of evasions occur for the case where the evasion attack concludes that predicted class is different from the actual_class, although the predicted class is actually equal to the actual class)** of evasions in the results of the evasion attack.

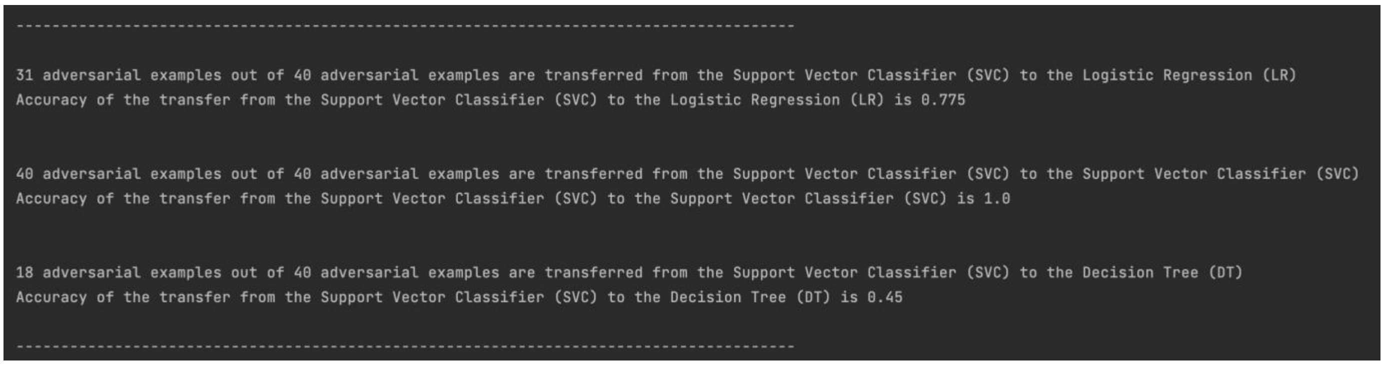
NOTE: For obtaining more supporting information, you can refer to the above screenshots named as “My Evasion Attack First Screenshot” and “My Evasion Attack Second Screenshot”.

Question - 5



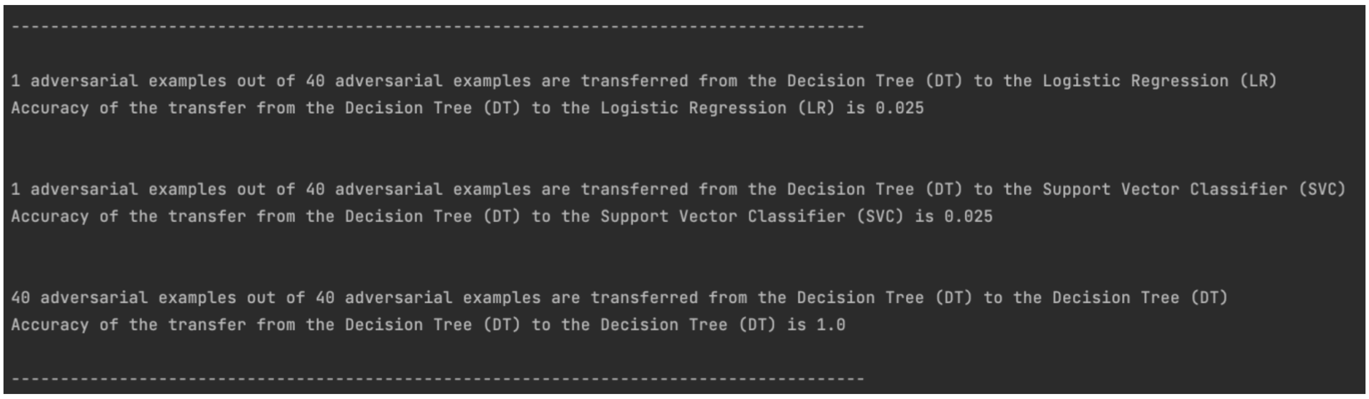
```
-----------------------------------------------------------------------------------

40 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to the Logistic Regression (LR)
Accuracy of the transfer from the Logistic Regression (LR) to the Logistic Regression (LR) is 1.0


10 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to the Support Vector Classifier (SVC)
Accuracy of the transfer from the Logistic Regression (LR) to the Support Vector Classifier (SVC) is 0.25


17 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to the Decision Tree (DT)
Accuracy of the transfer from the Logistic Regression (LR) to the Decision Tree (DT) is 0.42500000000000004

-----------------------------------------------------------------------------------
```

Results of the number of transfers for the LR (Transfer from LR to LR, transfer from LR to SVC, and transfer from LR to DT)



```
-----------------------------------------------------------------------------------

31 adversarial examples out of 40 adversarial examples are transferred from the Support Vector Classifier (SVC) to the Logistic Regression (LR)
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Logistic Regression (LR) is 0.775


40 adversarial examples out of 40 adversarial examples are transferred from the Support Vector Classifier (SVC) to the Support Vector Classifier (SVC)
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Support Vector Classifier (SVC) is 1.0


18 adversarial examples out of 40 adversarial examples are transferred from the Support Vector Classifier (SVC) to the Decision Tree (DT)
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Decision Tree (DT) is 0.45

-----------------------------------------------------------------------------------
```

Results of the transfers for the SVC (Transfer from SVC to LR, transfer from LR to SVC, and transfer from LR to DT)



```
-----------------------------------------------------------------------------------

1 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Logistic Regression (LR)
Accuracy of the transfer from the Decision Tree (DT) to the Logistic Regression (LR) is 0.025


1 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Support Vector Classifier (SVC)
Accuracy of the transfer from the Decision Tree (DT) to the Support Vector Classifier (SVC) is 0.025


40 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Decision Tree (DT)
Accuracy of the transfer from the Decision Tree (DT) to the Decision Tree (DT) is 1.0

-----------------------------------------------------------------------------------
```

Results of the transfers for the DT (Transfer from DT to LR, transfer from DT to SVC, and transfer from DT to DT.

So, as we can see from the above images:

FOR LR:

40 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to the Logistic Regression (LR)
Accuracy of the transfer from the Logistic Regression (LR) to the Logistic Regression (LR) is 1.0

10 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to SVC
Accuracy of the transfer from the Logistic Regression (LR) to the Support Vector Classifier (SVC) is 0.25

17 adversarial examples out of 40 adversarial examples are transferred from the Logistic Regression (LR) to the Decision Tree (DT)
Accuracy of the transfer from the Logistic Regression (LR) to the Decision Tree (DT) is 0.42500000000000004

FOR SVC:

31 adversarial examples out of 40 adversarial examples are transferred from the SVC to the Logistic Regression (LR)
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Logistic Regression (LR) is 0.775

40 adversarial examples out of 40 adversarial examples are transferred from the SVC to SVC.
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Support Vector Classifier (SVC) is 1.0

18 adversarial examples out of 40 adversarial examples are transferred from the Support Vector Classifier (SVC) to the Decision Tree (DT)
Accuracy of the transfer from the Support Vector Classifier (SVC) to the Decision Tree (DT) is 0.45

FOR DT:

1 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Logistic Regression (LR)
Accuracy of the transfer from the Decision Tree (DT) to the Logistic Regression (LR) is 0.025

1 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Support Vector Classifier (SVC)
Accuracy of the transfer from the Decision Tree (DT) to the Support Vector Classifier (SVC) is 0.025

40 adversarial examples out of 40 adversarial examples are transferred from the Decision Tree (DT) to the Decision Tree (DT)
Accuracy of the transfer from the Decision Tree (DT) to the Decision Tree (DT) is 1.0

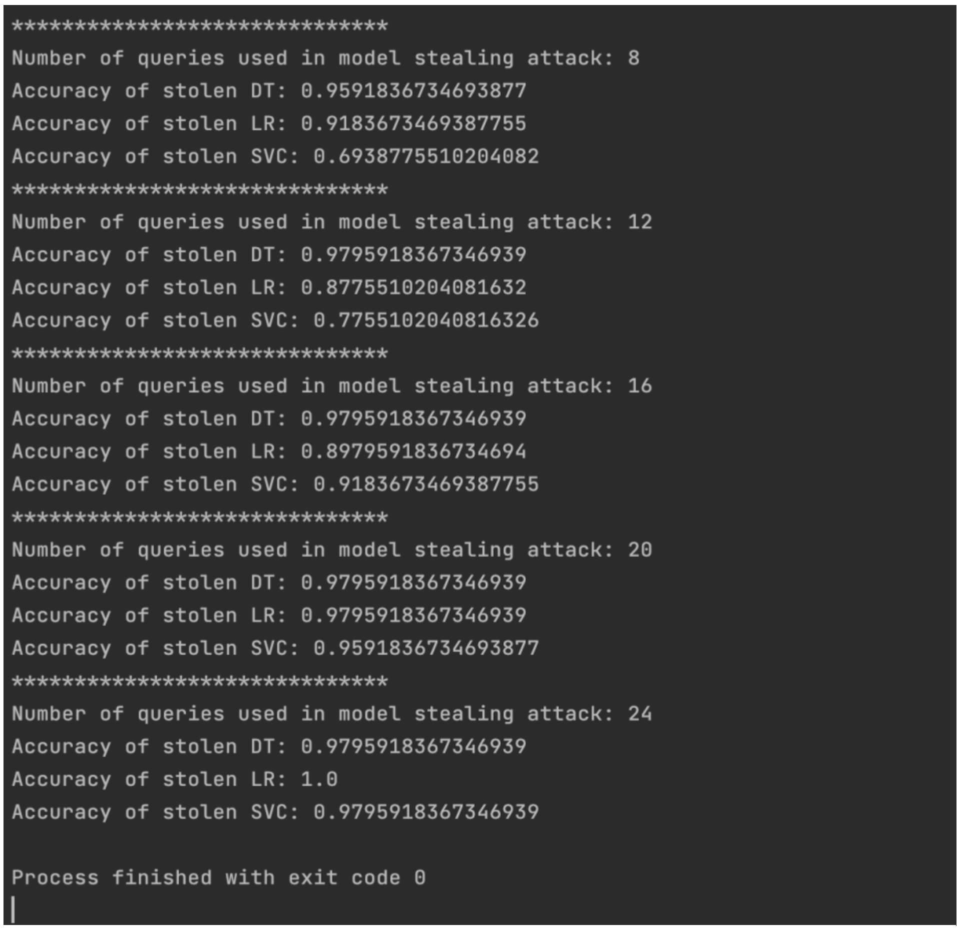
A Brief Note: SVC refers to the Support Vector Classifier model, DT refers to the Decision Tree model, and LR refers to the Logistic Regression model.

Discussion About the Results of Question-5: For the SVC (Support Vector Classifier) model, I think that my evasion attack has a high level of cross-model transferabiliity. Because, for the SVC model, the accuracies of the transfers are 0.45 and 0.775, which are almost 0.50 or higher than that. Moreover, the SVC transfer accuracies which are almost equal to or higher than the 0.50 indicate a high level of cross-model transferability of my evasion attack for the SVC (Support Vector Classifier) model. For the LR (Logistic Regression) model, I think that my evasion attack has a moderate level of cross-model transferability. Because, for the LR model, the accuracies of the transfers are 0.25 and 0.425, which are between 0 and 0.5. Since the accuracies of the transfers for the LR model are between 0 and 0.5, I can say that my evasion attack has a moderate level of cross-model transferability for the LR model. For the DT (Decision Tree) model, I think that my evasion attack has a low level of cross-model transferability. Because, for the DT model, the accuracies of the transfers are equal to 0.025, which are almost 0. Since the accuracies of the transfers for the DT model are approximately 0, I can say that my evasion attack a low-level of cross-model transferability for the DT model. For the LR model, the accuracies of transfers to the SVC model and to the DT model are 0.25 and 0.42500000000000004 respectively. For the SVC model, the accuracies of transfers to the LR model and to the DT model are 0.775 and 0.45. For the DT model, the accuracies of transfers to the LR model and to the SVC model are 0.025 and 0.025 respectively. Therefore, there exists 5 transfer accuracies **(0.25, 0.025, 0.025, 0.425, 0.45)** which are below 0.50, and 1 transfer accuracy **(0.775)** which is above 0.50. This means that approximately %83 percent **(5 / 6)** of the transfer accuracies are below 0.50, and %17 percent **(1 / 6)** of the transfer accuracies are above 0.50. Since the percentage of the number of transfer accuracies below 0.50 **(Percentage = %83)** is higher than the percentage of the number of transfer accuracies above 0.50 **(Percentage = %17)**, I can infer/conclude that my evasion attack **does not have a high (has a low)** cross-model transferability overall.

QUESTION - 6

	DT	LR	SVC
Number of Queries = 8	0.9591836734693877	0.9183673469387755	0.6938775510204082
Number of Queries = 12	0.9795918367346939	0.8775510204081632	0.7755102040816326
Number of Queries = 16	0.9795918367346939	0.8979591836734694	0.9183673469387755
Number of Queries = 20	0.9795918367346939	0.9795918367346939	0.9591836734693877
Number of Queries = 24	0.9795918367346939	1.0	0.9795918367346939

The table which shows the accuracies of the stolen machine learning models (DT, LR, SVC) for different number of queries used in the process of model stealing. (See the table I provided above this description)



```
******************************
Number of queries used in model stealing attack: 8
Accuracy of stolen DT: 0.9591836734693877
Accuracy of stolen LR: 0.9183673469387755
Accuracy of stolen SVC: 0.6938775510204082
******************************
Number of queries used in model stealing attack: 12
Accuracy of stolen DT: 0.9795918367346939
Accuracy of stolen LR: 0.8775510204081632
Accuracy of stolen SVC: 0.7755102040816326
******************************
Number of queries used in model stealing attack: 16
Accuracy of stolen DT: 0.9795918367346939
Accuracy of stolen LR: 0.8979591836734694
Accuracy of stolen SVC: 0.9183673469387755
******************************
Number of queries used in model stealing attack: 20
Accuracy of stolen DT: 0.9795918367346939
Accuracy of stolen LR: 0.9795918367346939
Accuracy of stolen SVC: 0.9591836734693877
******************************
Number of queries used in model stealing attack: 24
Accuracy of stolen DT: 0.9795918367346939
Accuracy of stolen LR: 1.0
Accuracy of stolen SVC: 0.9795918367346939

Process finished with exit code 0
```

The screenshot which shows all of the results I obtained in Question-6 for varying number of queries (8, 12, 16, 20, 24) used.

My discussion and observations about the results I obtained in Question-6: In general, I have observed from the results that when the total number of queries which are used in the model stealing attack increases, the accuracy value of the stolen models (DT, LR, SVC) also increases. In other words, I have generally observed that the accuracy value of each stolen model (DT or LR or SVC) is **directly proportional** to the number of queries which are used in my model stealing attack. However, as the results of my model stealing attack, I have observed some irregular/strange/anomalous results sometimes. For example, for the stolen DT (Decision Tree) model, I have observed that the accuracy of the stolen DT model stay constant (did not change) while the number of queries used in my model stealing attack is gradually increasing from 12 to 24. Moreover, for the stolen LR (Logistic Regression) model, I have observed that the accuracy value of the stolen LR decreases while the number of queries used in my model stealing attack is increasing from 8 to 12. Moreover, for the stolen LR (Logistic Regression) model, I have observed that the accuracy of my model stealing attack decreased when the number of queries used in my model stealing attack increases from 8 to 12. Other than these irregular/strange/anomalous patterns, as I said, I have generally observed that the accuracy value of each stolen model (DT or LR or SVC) is **directly proportional** to the number of queries which are used in my model stealing attack.

NOTE: Please see the above screenshots and table I provided for this question (Question-6).