Comp430 Assignment #3 Report
Burak Yıldırım 72849

Part 1: Reading

a) Because of their methodology, the writers make this remark. Each email has a spam score that is determined by a linear classifier. Each word in the email is assigned a weight according to the suggested approach. It gives words that are more likely to be used in spam emails greater weight. The weights of each word multiplied by their frequency are added together to create the email's spam score. As a result, the presence of another word has no impact on the score for a certain word.

b) At first, for each word in the emails spam scores are calculated using a linear classifier. The payload is then an email composed of words with low spam scores. Words in this email that have a high spam score can be introduced sparingly. Because the words with high spam scores will have less of an impact on the email's total spam score, the classifier will categorize this email as non-spam. Some words with high spam scores may be present in an email that is categorized as non-spam by maximizing the frequency of words with low spam ratings.

c) The Dynamic Threshold defense checks whether the overall spam score exceeds beyond a designated threshold value or not. The overall spam score is computed by the spam filter which calculates the spam score of each word then it adds up these values to finalize the overall spam score. The overall spam score is higher than a designated threshold value for spam emails. The threshold value is dynamically changed depending on the spam scores of all words in the email. The threshold is raised if words tend to have higher values which makes the classifier more difficult to label as spam. The threshold is lowered if the words tend to have lower values which makes the classifier easier to label as spam. Basically, the threshold value determines the sensitivity of detection. It is an effective defense strategy since it takes relationship among all words into account. Using this defense, it is much more difficult to have an email that is classified as non-spam containing words with high spam scores.

d) When compared to ordinary emails, spam emails tend to contain more words with high spam scores. Outlier detection can help identify these emails. Few words with high spam scores should be in the email's payload, whereas more words with low spam scores should be included. Even though this email has a lower total spam score than typical emails, it nevertheless contains more words with high spam ratings. The spam filter can determine the rate of words in each email that have a high spam score in order to identify this attack. When compared to the normal distribution of emails, it may then be used as a feature to identify possibly spam emails. If the rate of words with high spam scores is remarkably high, it will be marked as a potential spam email.

e) This optimization problem aims to produce a updated malware file (x) that can evade detection by a malware detector (F(.)) that employs machine learning while preserving the original malicious functionality (S). s denotes for the original malicious file. x is created by applying the function f(.) to s. x is obtained by applying the xor ($\oplus$) operation which flips some bits to the s. To determine the optimal settings for a detector's chance of finding x and its ability to distinguish between flipped bits in x and s, the hyperparameter (λ) is utilized. C(.) is the constraint function which provides x has the needed functionality. In order to satisfy the requirements of λ and a threshold value (T) for the quantity of flipped bits, the objective of this optimization problem is to minimize the likelihood (q) that x is detected by a detector. The threshold value represents the greatest separation between x and s in regards to the number of flipped bits. The original functionality is preserved when using a threshold value.

f) The amount of queries made to the malware detector in order to find a convenient perturbation that can evade detection is known as query-efficiency. A query-efficient attack can reduce effort and use fewer queries, which is important for the attack. Maintaining the original harmful functionality in a malware file's updated version is known as functionality-preserving. They are desirable and important because they offer effectiveness, practicality, and preserve original functioning. Since they rely on injecting content that has a purpose of evasion, i.e., content that is captured from benign samples, the attacks are query-efficient in this context. By only injecting content into the malicious program by using vulnerabilities in the file extension used to store programs on disk, they are functionality-preserving. This means that they don't modify the execution traces of the program.

g) The detection rate illustrates the proportion of changed malware files that are picked up by a malware detector that employs a machine learning approach. The amount of changed malware files developed and tested during each attack is the attack size. The number of queries represents the quantity of requests to the malware scanner. As seen in Figure 4, the detection rate declines as the number of queries rises. It is an intuitive conclusion given that additional queries gives space to the attacker to analyze more potential perturbations, which results in better perturbations that are harder to detect and evade. When the attack size increases, the detection rate decreases because produce smaller but detectable samples are produced due to features becoming sparse.

h) It leverages the concept of transferability because transferability is checking if adversarial samples can still be effective against other models. In this paper, it is tested that the attacks transfers to the other commercial antivirus software and they can evade more than 12 antivirus engines. The study states that commercial products can be evaded with a transfer attack.

Part 2: Implementation
Q1 –

Model Type	n	Accuracy
DT	0.05	0.9628571428571426
DT	0.1	0.9395918367346938
DT	0.2	0.883877551020408
DT	0.4	0.655510204081633
LR	0.05	0.9642857142857142
LR	0.1	0.9387755102040817
LR	0.2	0.8973469387755096
LR	0.4	0.7408163265306125
SVC	0.05	0.9891836734693882
SVC	0.1	0.9861224489795921
SVC	0.2	0.971836734693878
SVC	0.4	0.7971428571428569

When n increases, accuracy decreases because more samples will be flipped. When n is equal SVC seem to be more robust since it obtained higher accuracy values.

Q2 –

t	Recall
0.99	0.2
0.98	0.26
0.96	0.31
0.8	0.41
0.7	0.47
0.5	0.5

In this attack, recall shows our capability to detect positive samples which is forest fires. As it can be seen from the table, when t decreases recall increases. It is due to the when threshold is lower it is easier to predict it as positive.

Q3 –

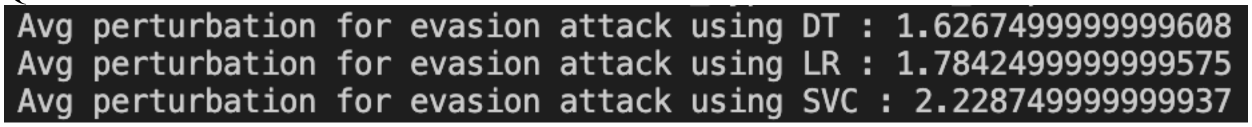
Model Type	Num Samples	Success Rate
DT	0	0
DT	1	0.94
DT	3	0.94
DT	5	0.94
DT	10	1
LR	0	0
LR	1	0.58
LR	3	0.64
LR	5	0.84
LR	10	1
SVC	0	0
SVC	1	0.44
SVC	3	0.5
SVC	5	0.6
SVC	10	0.82

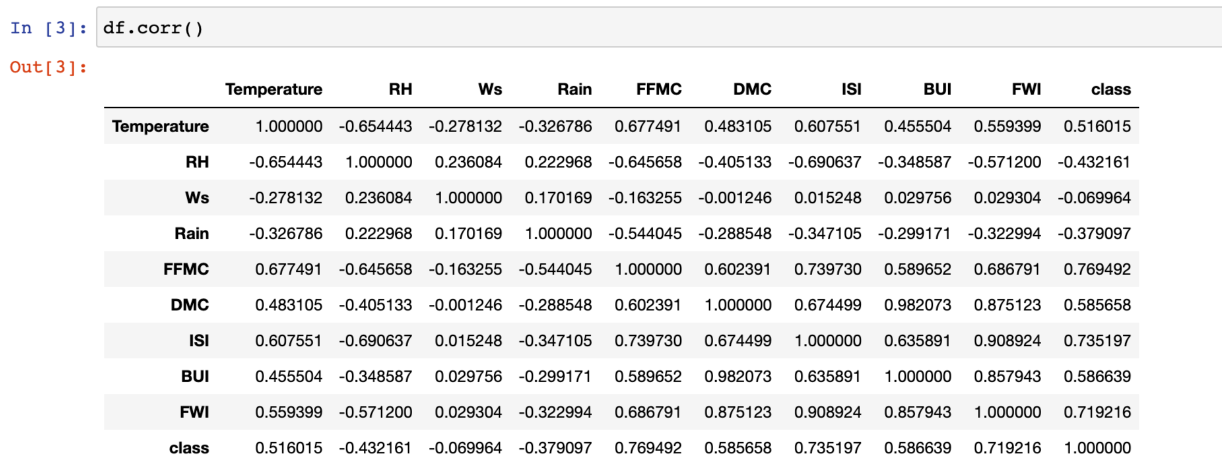
My trigger pattern is making the first feature which is the temperature to 70. According to the number of samples parameter, I selected unique random indexes with class 0 in the X_train dataset then I set their temperature feature to 70 and label to 1. Then I concatenate (inject) this newly created entry to the existing X_train_temp and y_train_temp lists.

Success Rate = $\frac{\sum_{x \in x_{test}} 1(M(x))}{N}$ where M is the model, M(x) is the prediction of the model, N is the total number of experiments, 1() is the one function that returns the total number of ones.

I designed my experiment by creating an X_test list and set it with zeros of each has length of features. I set the number of experiments as n = 50. Using unique n random indexes I take n random samples with class 0 from X_train change their first feature with 70 and group them as X_test. Lastly, I returned the success rate as (the total number of samples in X_test labeled as 1 by the model containing trigger pattern) / (the total number of samples in X_test contains trigger pattern).

Q4 –


```
Avg perturbation for evasion attack using DT : 1.6267499999999608
Avg perturbation for evasion attack using LR : 1.7842499999999575
Avg perturbation for evasion attack using SVC : 2.228749999999937
```



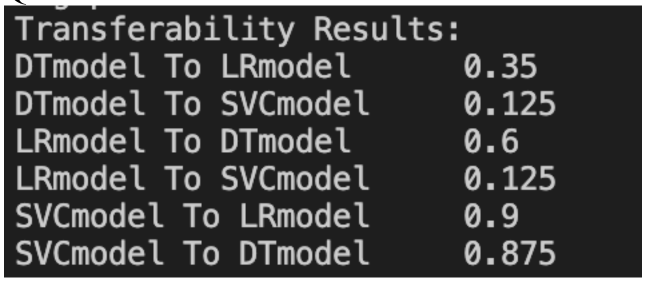
```
In [3]: df.corr()
```

Out[3]:

	Temperature	RH	Ws	Rain	FFMC	DMC	ISI	BUI	FWI	class
Temperature	1.000000	-0.654443	-0.278132	-0.326786	0.677491	0.483105	0.607551	0.455504	0.559399	0.516015
RH	-0.654443	1.000000	0.236084	0.222968	-0.645658	-0.405133	-0.690637	-0.348587	-0.571200	-0.432161
Ws	-0.278132	0.236084	1.000000	0.170169	-0.163255	-0.001246	0.015248	0.029756	0.029304	-0.069964
Rain	-0.326786	0.222968	0.170169	1.000000	-0.544045	-0.288548	-0.347105	-0.299171	-0.322994	-0.379097
FFMC	0.677491	-0.645658	-0.163255	-0.544045	1.000000	0.602391	0.739730	0.589652	0.686791	0.769492
DMC	0.483105	-0.405133	-0.001246	-0.288548	0.602391	1.000000	0.674499	0.982073	0.875123	0.585658
ISI	0.607551	-0.690637	0.015248	-0.347105	0.739730	0.674499	1.000000	0.635891	0.908924	0.735197
BUI	0.455504	-0.348587	0.029756	-0.299171	0.589652	0.982073	0.635891	1.000000	0.857943	0.586639
FWI	0.559399	-0.571200	0.029304	-0.322994	0.686791	0.875123	0.908924	0.857943	1.000000	0.719216
class	0.516015	-0.432161	-0.069964	-0.379097	0.769492	0.585658	0.735197	0.586639	0.719216	1.000000

For the evasion attack, I examined the correlation matrix of the dataset. I found that FFMC and ISI are the mostly related features to the class. Then, I found their minimum and maximum values from the dataset. If the class of a sample is 1 and its FFMC and ISI values are greater than the minimum FFMC and ISI values, I decreased the FFMC and ISI values by 0.1 each time until the class is changed. If the class of a sample is 0 and its FFMC and ISI values are less than the maximum FFMC and ISI values, I increased the FFMC and ISI values by 0.1 until the class is changed.

Q5 –



```
Transferability Results:
DTmodel To LRmodel      0.35
DTmodel To SVCmodel     0.125
LRmodel To DTmodel      0.6
LRmodel To SVCmodel     0.125
SVCmodel To LRmodel     0.9
SVCmodel To DTmodel     0.875
```

Based on these results, I can say that my evasion attack has a low cross-model transferability in general. Although, SVC model can be transferred to LR and DT models with high rates like 0.9 and 0.875 respectively.

Q6 –

Model Type	Num of Queries Used	Accuracy
DT	8	0.9591836734693877
LR	8	0.9183673469387755
SVC	8	0.6938775510204082
DT	12	0.9795918367346939
LR	12	0.8775510204081632
SVC	12	0.7755102040816326
DT	16	0.9795918367346939
LR	16	0.8979591836734694
SVC	16	0.9183673469387755
DT	20	0.9795918367346939
LR	20	0.9795918367346939
SVC	20	0.9591836734693877
DT	24	0.9795918367346939
LR	24	1.0
SVC	24	0.9795918367346939

As expected when the number of queries increases the accuracy increases as the number of examples increases our stolen model will more likely to perform better and give a better accuracy result. Using less number of queries such as 8 and 12, SVC performed poorly compared to the other models. As the number of queries increases the accuracy results of the models became more similar to each other.