COMP430 Data Privacy & Security - Project 3

Emir Şahin 72414

Part 1: Poisoning & Evasion Attacks

Question 1: Label Flipping Attack

	p = 0.05	p = 0.1	p = 0.2	p = 0.4
DT	0.9692	0.9567	0.9399	0.7769
LR	0.9794	0.9753	0.9698	0.9527
SVC	0.9542	0.9505	0.9462	0.7456

Increasing the p value has a predictable outcome on all models. There is a consistent inverse relationship between the p value and the accuracies of the models. The Linear Regression model is significantly more robust against the label flipping attack compared to the other models. The robustness of the Decision Tree Classifier model and the Support Vector Classification model are comparable.

Question 2: Defense Against Label Flipping

To detect outliers, I utilized the KMeans algorithm. I conduct an unsupervised training using the training data and generate binary labels for the same set I trained the KMeans algorithm on. The accuracy of the algorithm is ~60-65% which is expected considering it is unsupervised. I then compare 3 lists, the correct labels list, the attacked labels list and the labels list generated using the unsupervised trained KMeans algorithm. For any given index on these lists, first I checked if the correct label and the potentially attacked label differ (Which means the label on that index was attacked), if they do differ, then I checked if the labels generated by the KMeans algorithm had the correct label, if so, I incremented the counter indicating that the defense has correctly identified a label flip.

The defense is about as effective as the KMeans algorithm is accurate, ~60-65%.

p = 0.05 / 48 flips	25
p = 0.1 / 96 flips	54
p = 0.2 / 192 flips	114
p = 0.4 / 384 flips	227

These were the number of correctly identified flipped bits. The accuracy is 58.3%

Question 3: Evasion Attack

In every run I started by identifying which model the method was going to evade as each model required different strategies to successfully evade. There is an important variable named “deviationStep” which can be thought of as the resolution of the method, the lower the value of this variable the less perturbation is done but the running time increases. The method works by making small changes on the datapoint and preserving the change for the next cycle if the confidence of the model decreases. The change that lowers the confidence the most is preserved until the model incorrectly classifies the datapoint.

For Support Vector Classification, confidence is represented by the distance to the class dividing line. 8 new datapoints are created, since the datapoints have 4 dimensions every dimension is increased and decreased by the deviationStep resulting in 8 new datapoints. An element of randomness is added to the deviation process to ensure the method does not get stuck; this could happen if the values weren’t randomized in cases where 2 datapoints keep recursively calling each other as the next best datapoint with the lowest confidence. If any datapoint finally results in the model being misclassified, the while loop is broken and that datapoint is returned.

For Decision Tree Classifier, since the confidence values are mostly represented as [0, 1] or [1, 0] instead of float values similar to [0.6563, 0.3437], the algorithm determines the 2 most significant features and increments or decrements them by deviationStep resulting in 2 modified datapoints. These datapoints are continuously stretched in the 2 directions (2 most significant features represented as (0, 1) the stretching is done in the 2 configurations: (-, -), (+, +)). When the label predicted label is finally flipped the modified datapoint is returned.

Evasion algorithm for Linear Regression is similar to the evasion algorithm for Support Vector Classification. 8 datapoints are generated by increasing and decreasing each data dimension by the deviationStep value, the randomization is kept as it works, I am not sure if removing it would cause issues. For each of the 8 datapoints the distance between the 2 confidence values for the 2 classes are calculated, the lowest difference means the least confidence, the datapoint resulting in the least distance between the 2 confidence values is copied to the next cycle. When the label finally flips the while loop is broken and that datapoint is returned.

Average Perturbation for DT	1.048
Average Perturbation for LR	0.869
Average Perturbation for SVC	0.843

Question 4: Evasion Attack Transferability

Out of 40 adversarial examples crafted to evade Decision Tree Classifier:

> 18 of them transfer to Linear Regression.

> 20 of them transfer to Support Vector Classification.

Out of 40 adversarial examples crafted to evade Linear Regression:

> 23 of them transfer to Decision Tree Classifier.

> 19 of them transfer to Support Vector Classification.

Out of 40 adversarial examples crafted to evade Support Vector Classification:

> 32 of them transfer to Decision Tree Classifier.

> 29 of them transfer to Linear Regression.

I have no point of reference regarding the transferability of my evasion attack algorithms besides the one provided in the handout. The transferability of the Decision Tree Classifier evasion algorithm is slightly less than the results given in the handout, the transferability of the Linear Regression evasion algorithm is slightly more than the one provided in the handout and the transferability of the Support Vector Classification evasion algorithm is considerably more than the one provided in the handout.

Part 2: Backdoor Attacks In NLP

Question 1 & Question 2:

For these questions the implementation is intuitive. I completed these parts using pythonic methods/mechanisms and my implementation is heavily reliant on pandas methods and data structures.

Question 3:

The first step of my defense is tokenizing every “text” row in the dataframe into tokens. I classified these tokens into groups to filter them all out:

> Actual English words also present in the dictionary (look, joke, etc.)

> Actual English words that are not present in the dictionary (looked, jokes, etc.)

> Correctly used punctuation marks (., /, <, etc.)

> Incorrectly used punctuation marks (.., ..., /<, etc.)

> Numbers and digits

> Contractions (n’t, ‘re, ‘m, ‘s, ‘ve, ‘ll, ‘d)

And finally, there is the backdoor trigger word that we want to filter out which is its own class. After tokenizing the words, I calculate their frequencies since the backdoor trigger word is likely to be repeated many times. The total number of tokens is reduced to ~500 most frequent tokens. The number of most frequent tokens taken into consideration before filtering is dependent on the threshold_factor, I've set this value to 9.5 as it filters out the most non-frequent words while also consistently keeping the backdoor trigger word inside the most frequent tokens.

After the frequency analysis is complete, I apply filtering, which is done in many steps. I start by filtering out actual English words that are present in the dictionary by removing all words present in the word_list (word_list is the local list of nltk words) both uppercase and lowercase. Then I remove punctuation marks and incorrectly used punctuation marks, then I remove the contractions, then I remove all the numbers. This filtering process removes the bulk of the words that we want to keep in the dataset. I then filter out actual English words that are not present in the dictionary by removing common suffixes from the tokens and checking if the token is in the dictionary when the suffix is removed. After this whole filtering process is complete, I end up with a list of ~5-10 words that are to be removed from the entire dataset.

Finally, I return the dataset with all the undesired words removed. I tested by writing the poisoned dataframe and the sanitized dataframe into .csv files and checking the number of times the identified backdoor trigger word is repeated in both .csv files. In all my test cases all instances of the backdoor trigger word(s) were removed in the sanitized dataset. The filtering process also removes some words that are actual English words that are not in the dictionary, these are not taken into account as they're extreme corner cases (children, tries, etc.).

Part 2 - Conclusions:

There are some unexpected results, I ran the notebook on my own device on a virtual machine, each module took around 6 minutes to complete, the unexpected results could be caused by this. Especially in the word level backdoor attack without defense. The sentence level attack consistently reduces the accuracy of the models. The word level attack also reduces the accuracy but for high poison rates and high word counts the accuracy increases in some cases.