Classify URL Reputation

By: Randy Grant

Abstract

This attempts to classify a URL's reputation such that the result comes back as "benign" or "not benign". The hypothesis is that a URL will have enough features to where a classification model can recognize it as those 2 options. This is based on the known factors of a full URI such as length, subdomain count, constant count, etc.

Design

The classification models chosen were set as a baseline first, and then via GridSearchCV were multiple parameters run against the training data to where the best set of hyperparameters were chosen. Based on the best models, those were then combined via VotingClassifier into an ensemble model.

Data

The original dataset was a list of known URLs that had a reputation in another column (2 in column in total). The dataset was chosen from Kaggle and inspected for its source (AV vendors) prior to feature engineering in many other columns. The final columns used were:

```
df.columns
```

✓ 0.1s

```
Index(['url', 'type', 'domain', 'dir_1', 'dir_2', 'dir_3', 'dir_4', 'dir_5',
       'dir_6', 'dir_7', 'dir_8', 'dir_9', 'dir_10', 'count_of_dirs',
       'subdomain_count', 'domain_len', 'url_len', 'dir_1_len', 'dir_2_len',
       'dir_3_len', 'dir_4_len', 'dir_5_len', 'dir_6_len', 'dir_7_len',
       'dir_8_len', 'dir_9_len', 'dir_10_len', 'domain_vowel_count',
       'domain_consonant_count', 'domain_number_count', 'domain_hyphen_count'],
      dtype='object')
```

Algorithms

Feature Engineering

1. Split URIs into “dirs” based on a “/” delimiter
2. Computed length of multiple columns after split
3. Created counts of constants, vowels, numbers, hyphens

Models

Logistic regression, KNN, decision tree, random forest, and ensemble based on the best hyperparameters of those 4. The dataset was split into a 80/20 train-test-split. The training data was then balanced out to where there were 10k 1s and 10k 0s. The test set was left alone.

Model Scores

Models	Accuracy Score	Precision Score	Recall Score	F1 Score
Decision Tree (criterion="entropy", max_depth=17)	0.925539864006575	0.995879686856201	0.924929402760550	0.959094174391954
Decision Tree (default settings)	0.925539864006575	0.995879686856201	0.924929402760550	0.959094174391954
KNN (5 neighbors)	0.925539864006575	0.995879686856201	0.924929402760550	0.959094174391954
KNN (default settings)	0.942115619318040	0.993931229863348	0.944432714892718	0.968549969551390
Logistic Regression (C=0.01, penalty="L1", solver="LibLinear")	0.925539864006575	0.995879686856201	0.924929402760550	0.959094174391954
Logistic Regression (default settings)	0.913160975366758	0.980623891147339	0.926288564566783	0.952682115834832
Random Forest (default settings)	0.981269770106353	0.995847797062750	0.984257475389934	0.990018714909544
Random Forest (estimators=300)	0.925539864006575	0.995879686856201	0.924929402760550	0.959094174391954

Tools

Python, Pandas for data manipulation, Scikit-learn for models and scores, Matplotlib and Seaborn for plots and graphs, Excel for putting the scores into one format.

Communication

These slides and visualizations were presented, and this was uploaded to github.