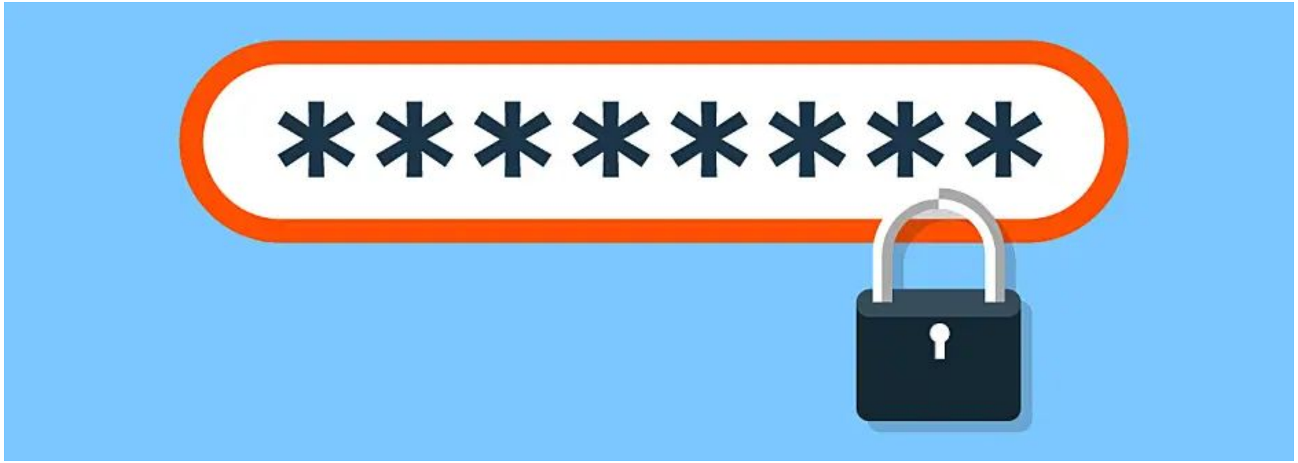
Data Science for Cybersecurity - Password Strength Meter

Dwi Gustin Nurdialit

Problem

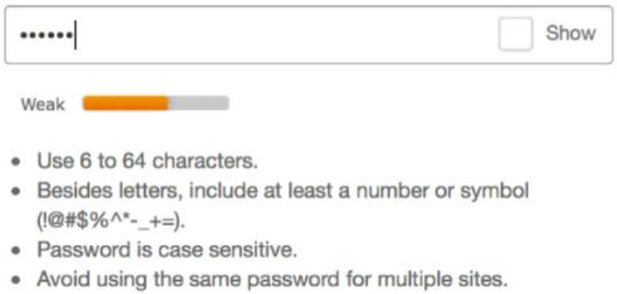
Based on Wikipedia, **computer security**, **cybersecurity**, or **information technology security** (**IT security**) is the protection of computer systems and networks from information disclosure, theft of or damage to their hardware, software, or electronic data, as well as from the disruption or misdirection of the services they provide.

Password

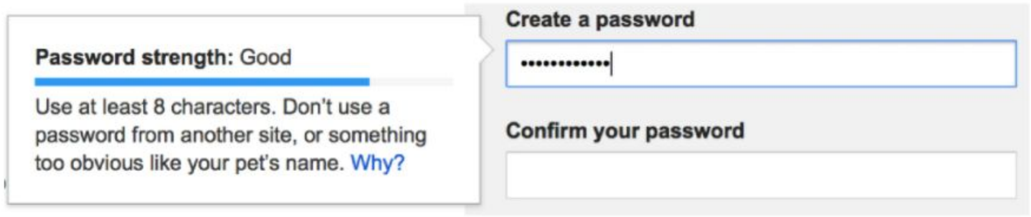


Passwords are a vital component of system security.

To Create Password Strength Meter



- Use 6 to 64 characters.
- Besides letters, include at least a number or symbol (!@#$%^*-_+=).
- Password is case sensitive.
- Avoid using the same password for multiple sites.

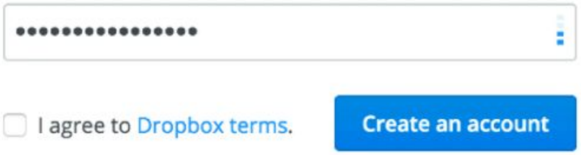


Password strength: Good

Use at least 8 characters. Don't use a password from another site, or something too obvious like your pet's name. Why?

Create a password

Confirm your password



□ I agree to Dropbox terms.

Traditional Ways:

- Based on rules specified by a human programmer→ conditional if, else, etc; Regular Expression
- Requires exact parameters for all condition
- Focused on analysis of past data

Machine Learning:

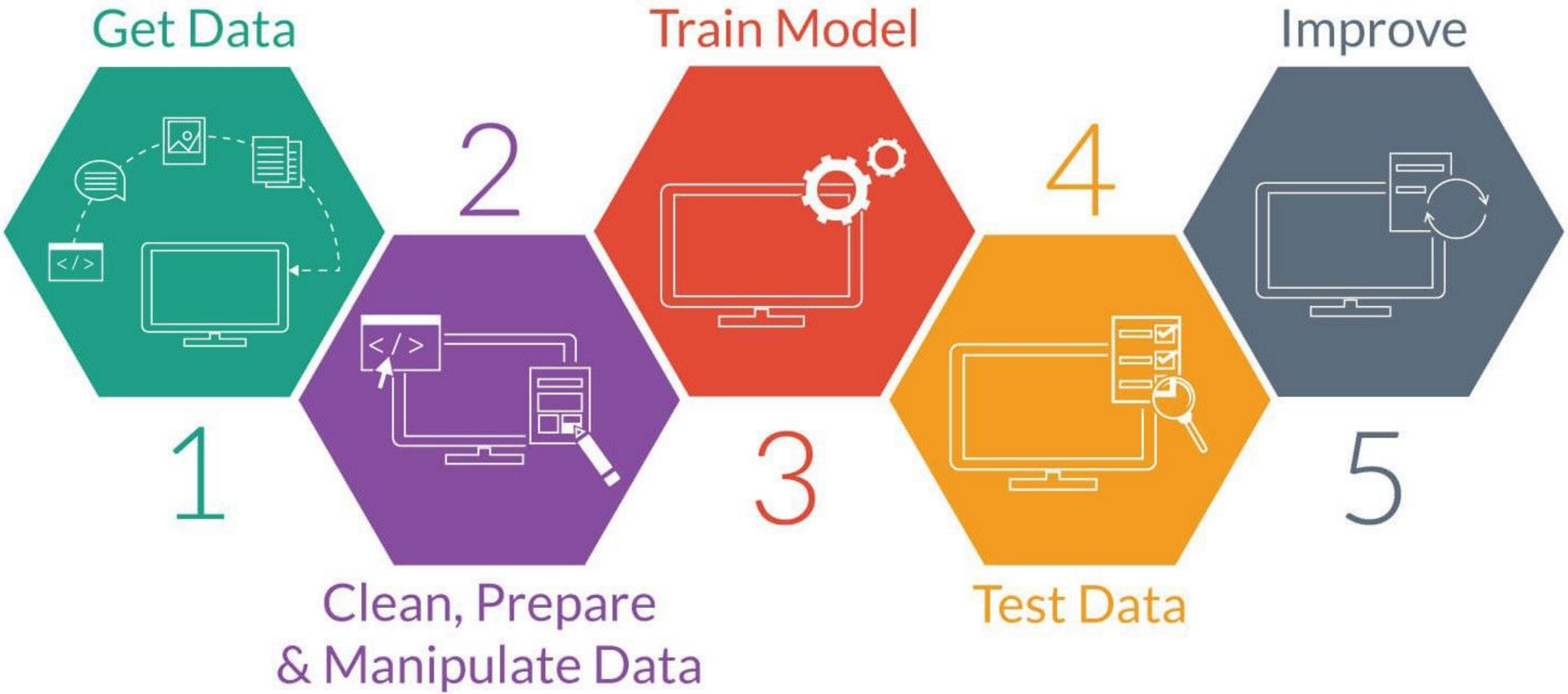
- Based on function originally written by human programmer
- Uses fuzzy logic to approximate decision making
- Performance on a given task improves over time
- Focused on making prediction with new data

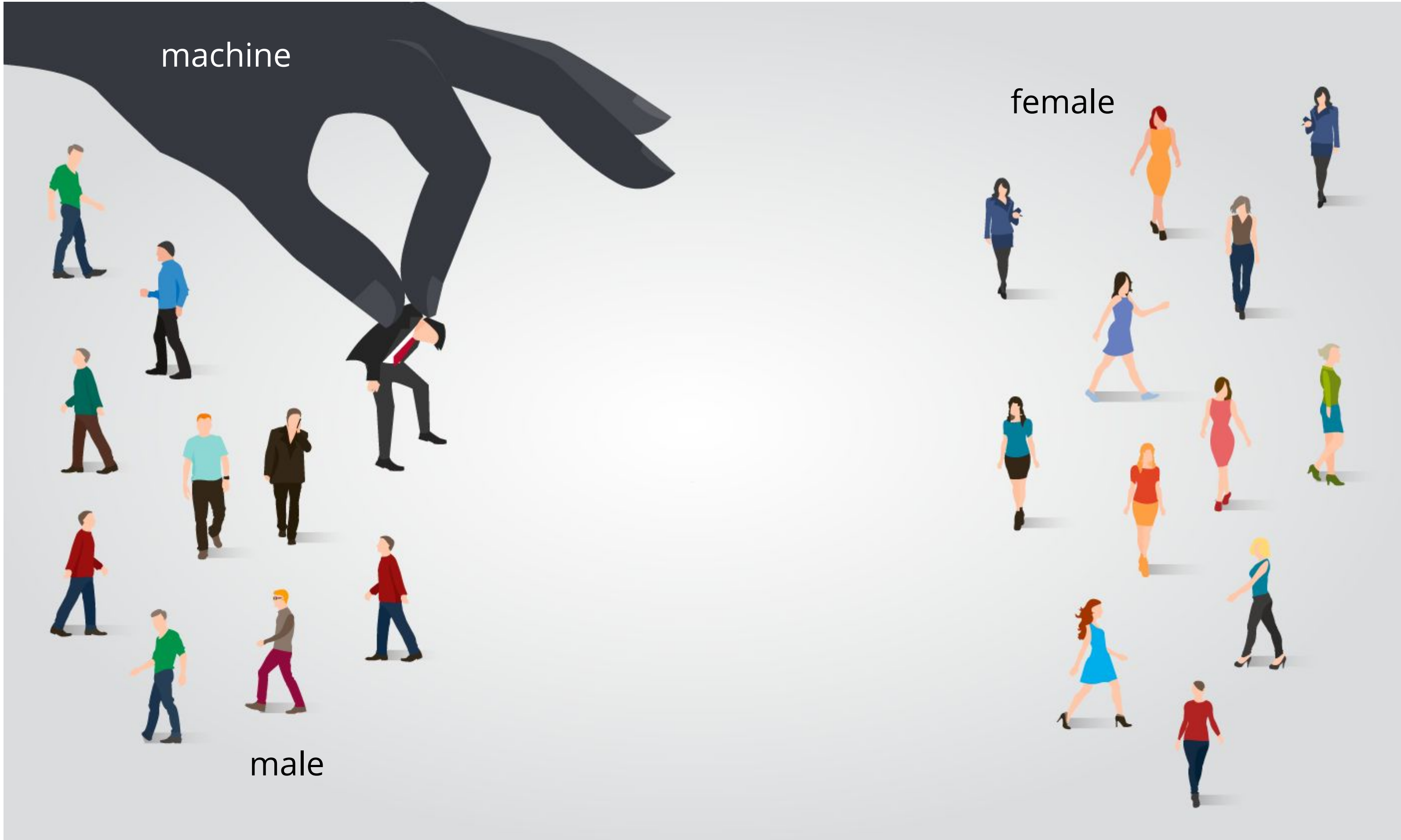
Machine Learning for Cybersecurity

Create password strength meter using predictive model as a classification task.

- What is needed:
  - Dataset
  - Python IDE
  - Machine Learning and Statistical Packages

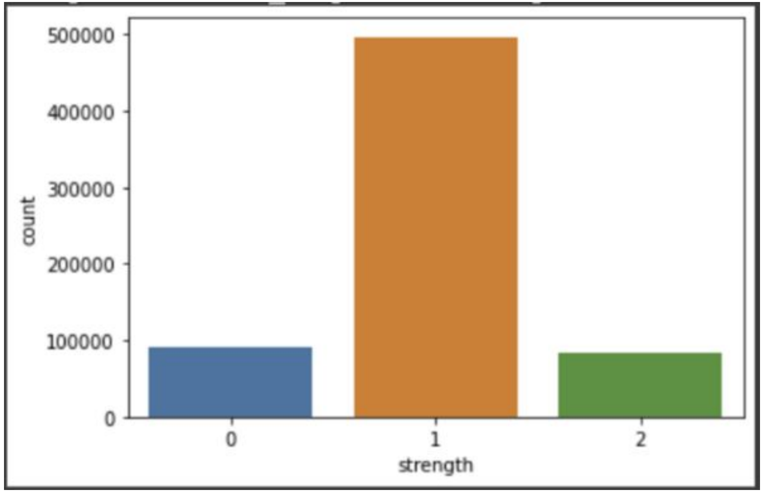
Machine Learning Flow




machine
female
male

Data Preparation

- Get the data: Dataset from Kaggle
- Dataset consist of 670.000 unique values for password collected online
- It has 3-level (0,1,2) password strength, 0 for weak, 1 for medium, 2 for strong
- The strength of passwords, according to
  - its length (=number of characters)
  - the ratio of the upper case to all characters
  - the ratio of a number to all characters
  - the ratio of special characters(“;” , “,”, “[“, Etc...) to all characters



Convert Into the Format of NumPy Array

- Developing machine learning models in Python often requires the use of NumPy arrays
- Codes:

```
password_tuple = np.array(data)
```

Feature Extraction

- Extract our dependent and independent feature
- Codes:

```
x = [labels[0] for labels in password_tuple]
y = [labels[1] for labels in password_tuple]
```

TF-IDF (Term Frequency - Inverse Document)

- TF-IDF is one of the most popular term-weighting schemes today.
- A numerical statistic that is intended to reflect how important a word is to document in a collection or corpus.
- The TF-IDF value increases proportionally to the number of times a word appears in the document.

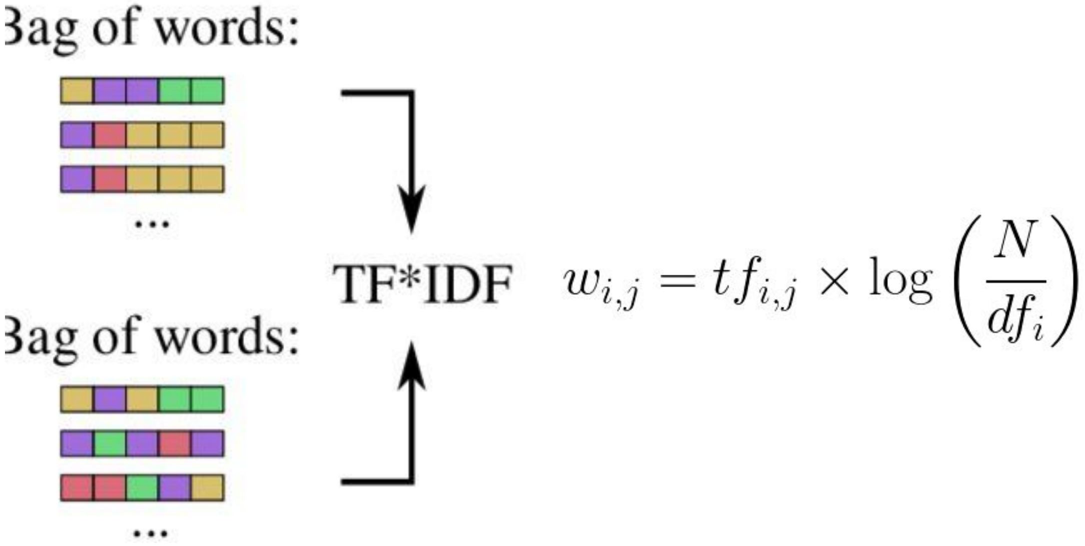
How is TF-IDF Calculated?

- Term Frequency (TF)

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

- Inverse Document Frequency (IDF)

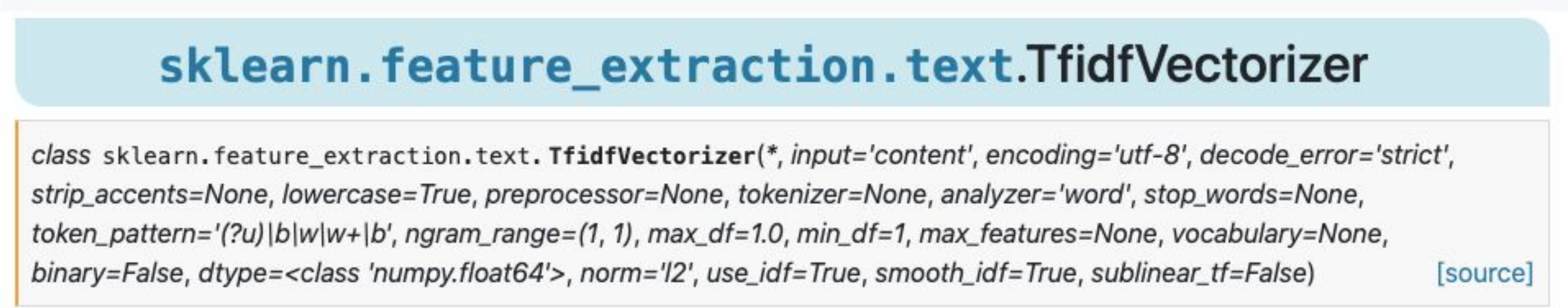
$$idf(w) = log(\frac{N}{df_t})$$



$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Apply TF-IDF on the Data

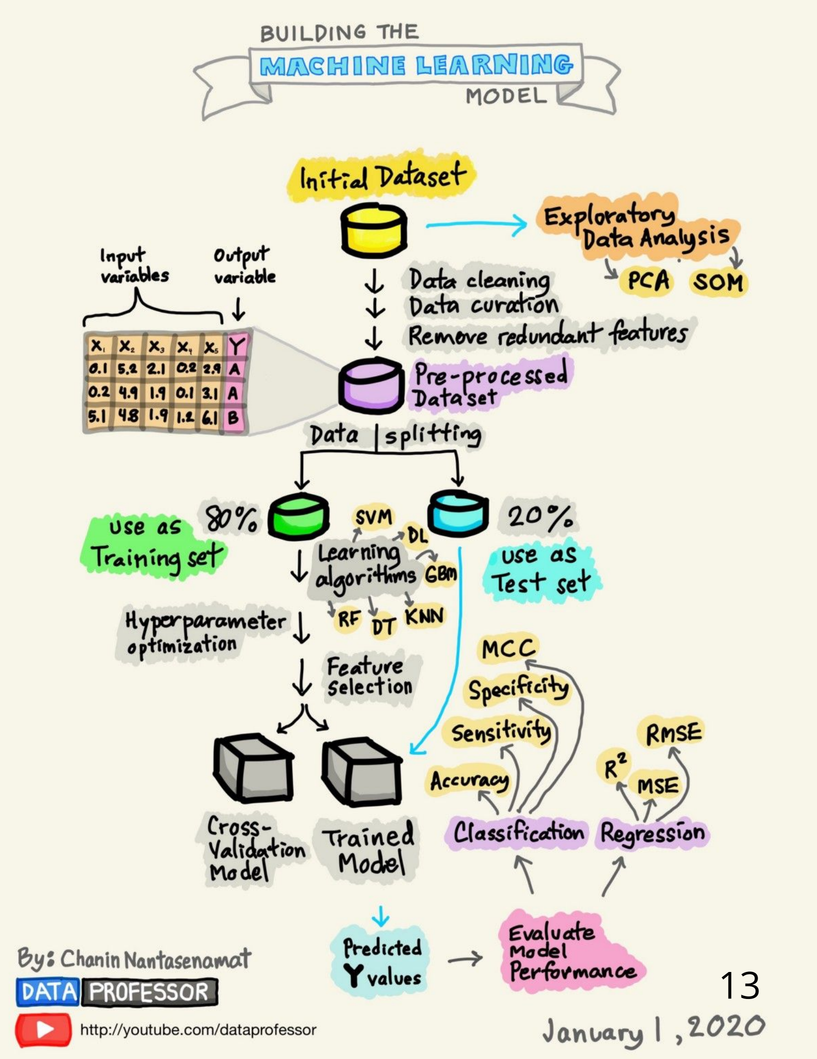
- Define function to split the parameter into character
- Applying TF-IDF using TfidfVectorizer from sklearn package



sklearn.feature_extraction.text.TfidfVectorizer

```
class sklearn.feature_extraction.text.TfidfVectorizer(*, input='content', encoding='utf-8', decode_error='strict',
strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None,
token_pattern='(?u)|b|w|w+|b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None,
binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False)        [source]
```

Build Classification Machine Learning Model



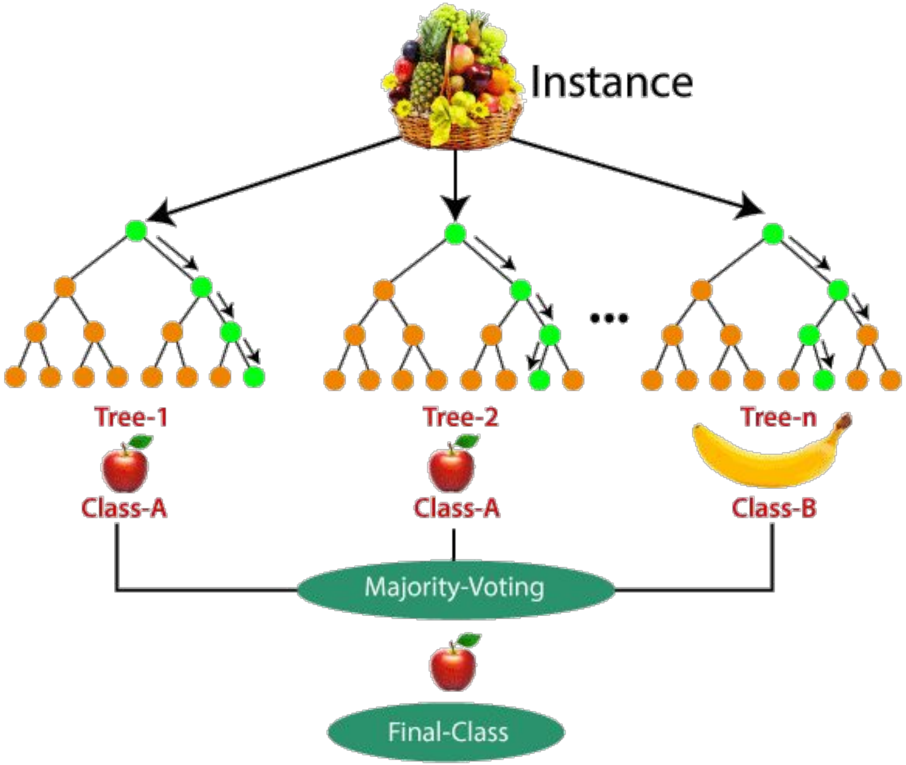
X_1	X_2	X_3	X_4	X_5	Y
0.1	5.2	2.1	0.2	2.9	A
0.2	4.9	1.9	0.1	3.1	A
5.1	4.8	1.9	1.2	6.1	B

codes

Modelling Experiment

Random Forest

- Is an ensemble tool that takes a subset of observation and subset of variables to build a decision tree
- It builds multiple such decision tree and amalgamates them together to get more accurate and stable prediction



Model Evaluation

Hyperparameter Tuning

Conclusion and Future Work

- We can use machine learning algorithm for cybersecurity task
- The best algorithm used for tokenization is TF-IDF, while for classification is random forest
- Not always the hyperparameter tuning process always get better performance
- For future data scientist doing similar project: get more data, increase the number of classes, increase the complexity of the model and regularize

Don’t stop learning by doing!!