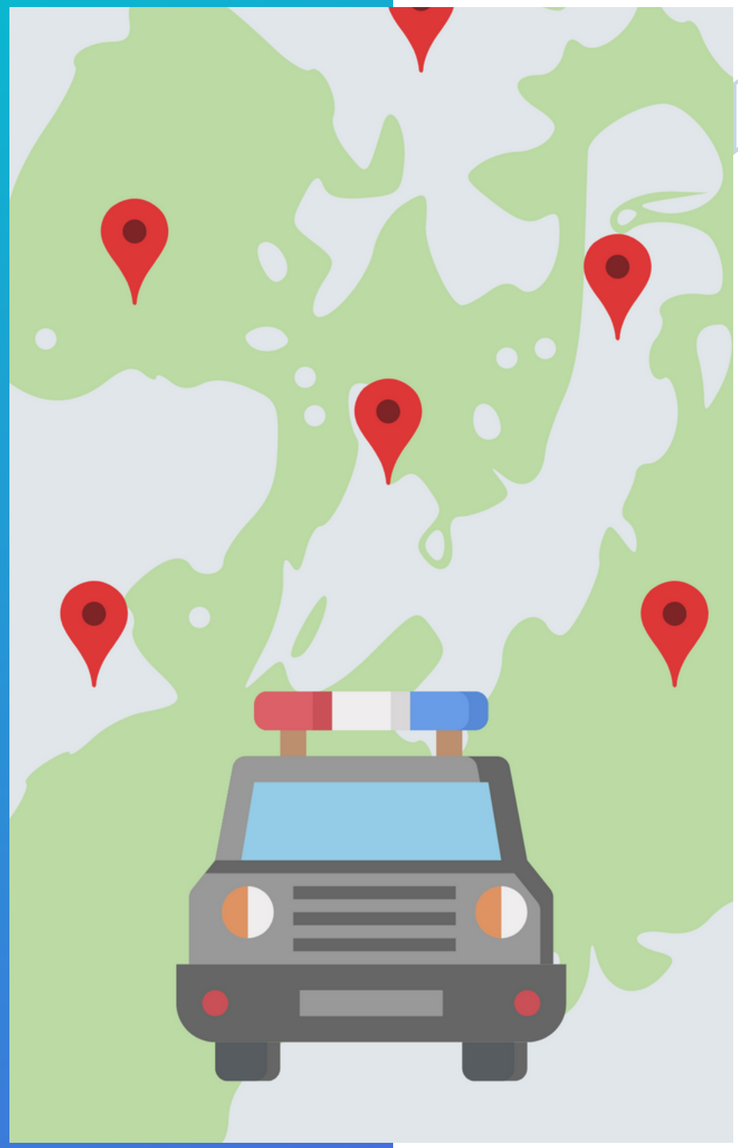


ROVE

BY IDENTIFIERS

PSID: INTL-DA-07

TEAM LEADER NAME: VARTUL BAHUGUNA

PROBLEM STATEMENT

Identification of crime prone area

. . .

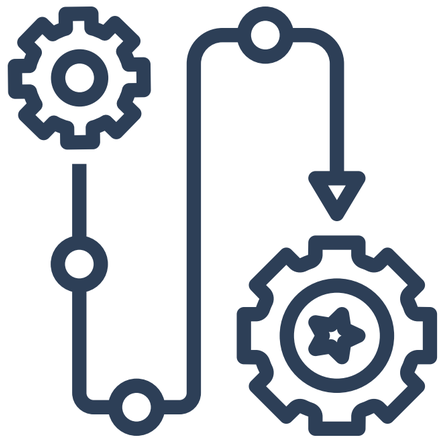
DATA COLLECTION AND AUGMENTATION

- Data set used is provided by Manthan's official site.
- Data is augmented for additional attributes; Gender and Age for more precise analysis.

DATA COLLECTION AND AUGMENTATION

	District	Event	Circle	Police Station	Caller Source	Event Type	Event Sub-Type	Create Date/Time	Latitude	Longitude	latitude1	longitude1	Age	gender	date2
0	LUCKNOW	P01042100004	C1	PS1	PHONE	Information Against Police	Misbehavior By Prv	01/04/2021 00:00:00	26.834	81.008	26.257089	81.639883	22	F	2021-01-04 00:00:00
1	LUCKNOW	P01042104316	C1	PS1	PHONE	Threat In Person	Attack	01/04/2021 12:09:00	26.828	81.014	26.350572	81.256241	49	M	2021-01-04 12:09:00
2	LUCKNOW	P01042104847	C1	PS1	PHONE	Dispute	Dispute In Hospital	01/04/2021 12:51:00	26.840	81.009	26.414184	81.048264	42	F	2021-01-04 12:51:00
3	LUCKNOW	P01042105074	C1	PS1	PHONE	Gambling	Play Cards	01/04/2021 13:10:00	26.828	81.002	26.741813	81.411443	6	M	2021-01-04 13:10:00
4	LUCKNOW	P01042105152	C1	PS1	PHONE	Threat In Person	Attack	01/04/2021 13:18:00	26.834	81.033	26.350578	81.020204	49	F	2021-01-04 13:18:00

PRE-PROCESSING



- TOKENIZATION: The tokens were extracted from the raw data provided.
- STOP WORDS: Non-essential English words used in a sentence are removed.
- LEMMATIZATION & STEMMING: were used to extract context of a word.

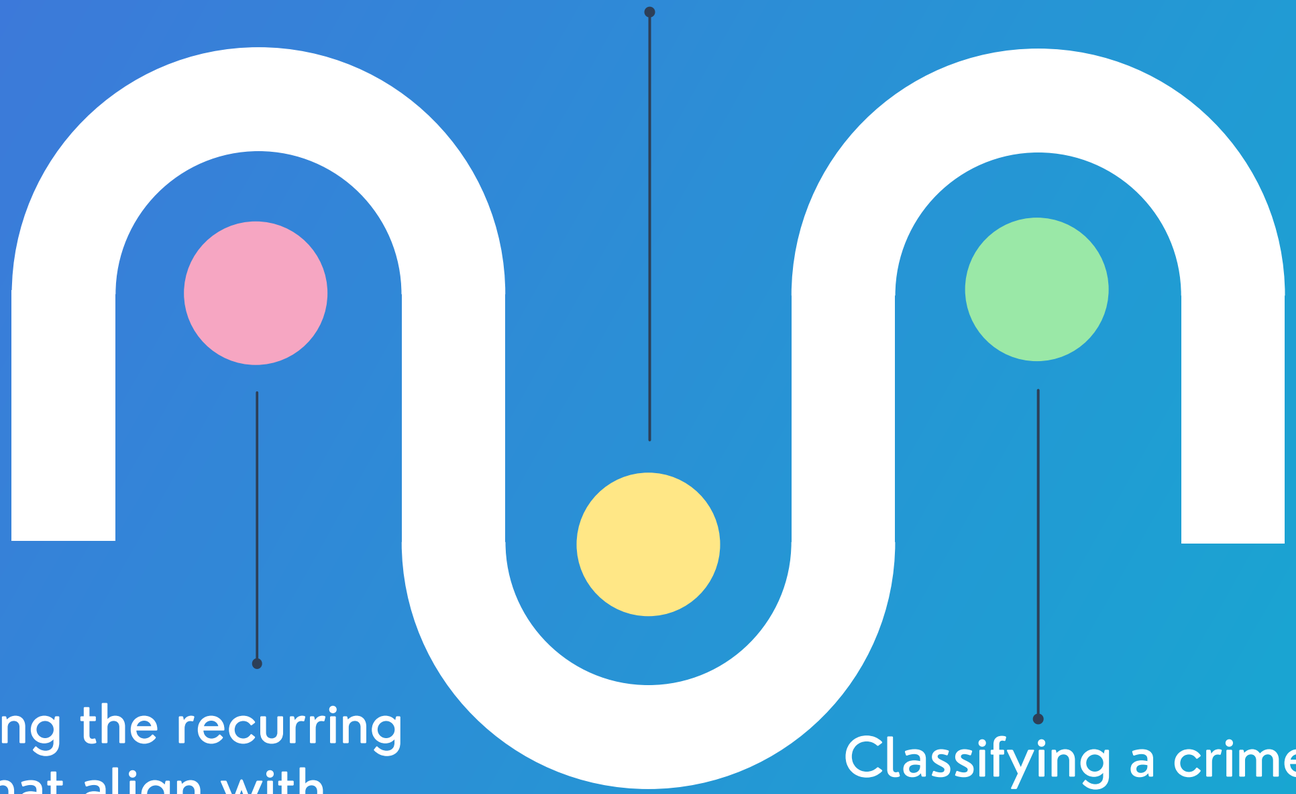
STOPWORDS REMOVAL AND TOKENIZATION

In [9]:
```
df = df.dropna()                                    #drop null values
df['Event Type'] = df['Event Type'].apply(str)      #converting floats and ints to string
ls =df['Event Type'].to_list()
n = len(ls)
ps = ls.copy()
for i in range(0,n):
    s = ls[i]
    stop_words = set(stopwords.words('english'))    #stop words removal
    word_tokens = word_tokenize(s)                  #tokenizing sentences
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    ls[i].join(filtered_sentence)
    ls[i] = ls[i].lower()
ls
```

Out[9]:
```
['information against police',
 'threat in person',
 'dispute',
 'gambling',
 'threat in person',
 'missing',
 'information against police',
 'theft',
 'dispute',
 'dispute',
 'dispute',
 'domestic violence',
 'threat in person',
 'threat in person',
 'threat in person',
 'property disputes',
 'information against police',
 'property disputes',
 'domestic violence',
```

FORMULATION OF INITIAL MODEL

Classify obtained words in different levels on basis of severity.



Identifying the recurring words that align with crime.

Classifying a crime in levels; if two different level of crimes are identified level assigned is one with higher order.

MODEL BUILDING

1. Identification of pattern for training model.
2. Generating rules using spacy pipeline
3. build training data set using these rules
4. build & train model using the data set using blank NER model

PATTERN GENERATION

In [9]:
```
def create_patterns(t, ls):
    patterns = []
    for item in ls:
        pattern = {
            "label":t,
            "pattern":item
        }

        patterns.append(pattern)
    return patterns
```

In [ ]:
```
```

In [10]:
```
pattern1 = create_patterns('L1', l1)
pattern2 = create_patterns('L2', l2)
pattern3 = create_patterns('L3', l3)
patterns = pattern1+pattern2+pattern3
patterns
```

Out[10]:
```
[{'label': 'L1', 'pattern': 'trafficking'},
 {'label': 'L1', 'pattern': 'election Offences'},
 {'label': 'L1', 'pattern': 'explosive'},
 {'label': 'L1', 'pattern': 'murder'},
 {'label': 'L1', 'pattern': 'assault'}
```

TRAINING THE DATASET

In [11]:
```
def gen_rules(patterns):
    nlp = English()
    ruler = nlp.add_pipe("entity_ruler")
    ruler.add_patterns(patterns)
    nlp.to_disk("crime_ner")
```

In [12]:
```
gen_rules(patterns)
```

In [13]:
```
nlp = spacy.load('crime_ner')
```

In [20]:
```
def test_model(model, text):
    doc = nlp(text)
    results = []
    entities = []
    for ent in doc.ents:
        entities.append((ent.start_char, ent.end_char, ent.label_))
    if(len(entities) > 0):
        results = (text, {'entities': entities})
        return results
TRAIN_DATA
```

Out[20]:
```
[('threat in person', {'entities': [(0, 6, 'L3')]}),
 ('dispute', {'entities': [(0, 7, 'L3')]}),
 ('gambling', {'entities': [(0, 8, 'L3')]}),
 ('threat in person', {'entities': [(0, 6, 'L3')]}),
 ('missing', {'entities': [(0, 7, 'L3')]}),
 ('theft', {'entities': [(0, 5, 'L3')]}),
 ('dispute', {'entities': [(0, 7, 'L3')]}),
```

TOKENIZATION & LEMMITIZATION

In [31]:
```python
def preprocess_text(doc):
    # Remove all special characters
    doc = re.sub(r'\W', ' ', str(doc))

    # Remove all single characters
    doc = re.sub(r'\s+[a-zA-Z]\s+', ' ', doc)

    # Remove single characters from the start
    doc = re.sub(r'\^[a-zA-Z]\s+', ' ', doc)

    # Substituting multiple spaces with single space
    doc = re.sub(r'\s+', ' ', doc, flags=re.I)

    #Removing prefixed 'b'
    doc = re.sub(r'^b\s+', "", doc)

    #Converting to Lowercase
    doc = doc.lower()

    #Lemmatization
    tok = doc.split()
    tok = [stemmer.lemmatize(word) for word in tok]
    tok = [word for word in tok if word not in en_stop]
    tok = [word for word in tok if len(word)>3]

    preprocessed_text = ' '.join(tok)

    return preprocessed_text
```

In [35]:
```python
sent = preprocess_text('threat in person')
print(sent)
```

```
threat person
```

Out[35]:
```
str
```

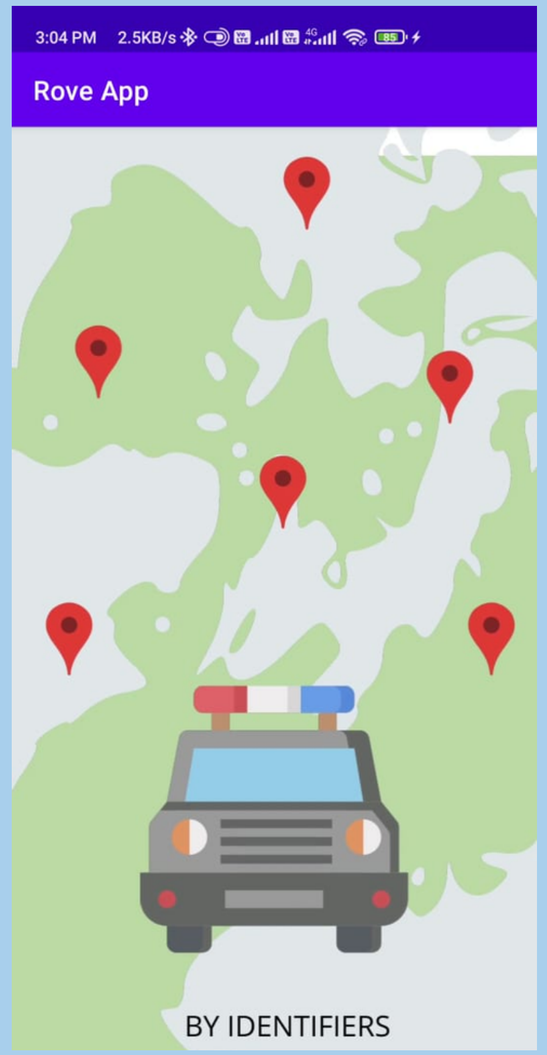
SPELL CHECKER

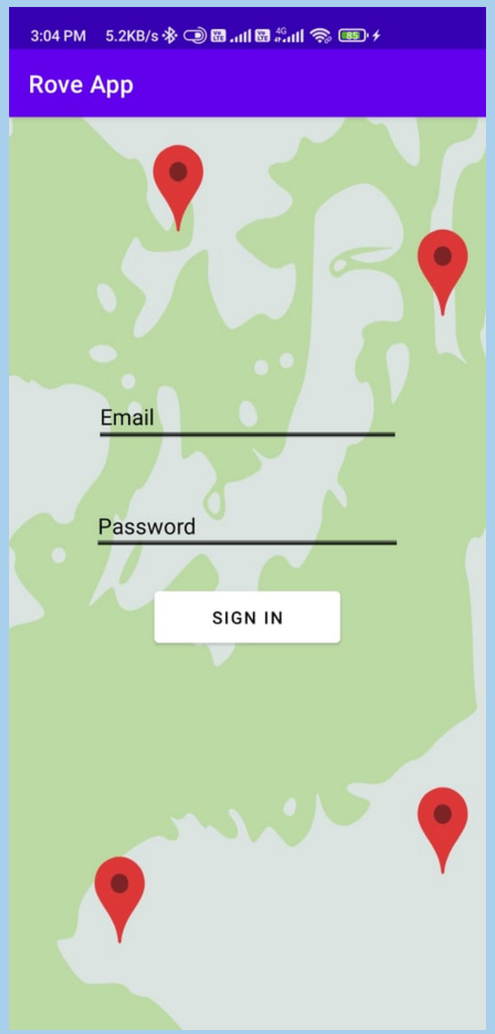
In [ ]:
```
%%time
ft_model = FastText(
    word_tokenized_corpus,
    size = embedding_size,
    window = window_size,
    min_count = min_word,
    sample = down_sampling,
    sg = 1,
    iter = 100
                    )
```

In [36]:
```
from spellchecker import SpellChecker

spell = SpellChecker()

# find those words that may be misspelled
misspelled = spell.unknown(['something', 'is', 'hapenning', 'here'])

for word in misspelled:
    # Get the one `most likely` answer
    print(spell.correction(word))

    # Get a list of `likely` options
    print(spell.candidates(word))
```

```
happenning
{'hapening', 'happenning'}
```

APPLICATION IMPLEMENTED





THANK YOU