Classification of crimes using machine learning techniques for national crime data

Marija Trpchevska*, Aleksandra Dedinec M.Sc.*
*Faculty of Computer Science and Engineering, University Ss. Cyril & Methodius Skopje,
Republic of North Macedonia
marija.trpchevska@students.finki.ukim.mk, aleksandra.kanevche@finki.ukim.mk

***Abstract*—The paper aims to assist in a more accurate classification of crimes provided by North Macedonia's Ministry of Internal Affairs using machine learning techniques. By means of natural language processing, data is transformed into a format suitable for term frequency-inverse document frequency (TF-IDF) vectorization. Bayesian and Support Vector Machine models are utilized and evaluated. The results show non-negligible improvement toward a successful classification of crimes, confirming the beneficial nature of machine learning techniques towards such tasks.**

***Keywords*—Natural language processing, term frequency-inverse document frequency, TF-IDF, classification, multinomial Naive Bayes, Support Vector Machine**

I. INTRODUCTION

Classification as a concept is at the core of machine learning - data exists in such overflowing and daunting quantities in today's world that one needs to be able to filter through them by automated means should he wish to get to a more complex analysis. In past decades, machine learning models have diversified and have been tailored towards solving domain-specific classification tasks with a wide variety of inputs.

In all cases, however, the output is discrete; a classifier, as the name suggests, must choose between a finite number of categories with which to bind a given input. Outputs which have continuous values fall under the umbrella of regression problems. Both classification and regression are considered as so-called supervised learning approaches [1].The implied existence of an output itself means that someone (usually a human) has gone through the unenviable task of connecting certain inputs with corresponding outputs employing some sort of prior knowledge of the valid ways in which to make connections. A subset of these pairs is then fed to a classifier model whose task is to infer the rule-set used when those pairs were labelled. To determine whether the classifier has been trained well, those inputs previously unseen by it are given for classification (the output is therefore withheld) and its guesswork is compared. A good classifier mimics the decision rules made by a human, so most guesses should in principle coincide with the outputs.

The simplest case is one in which the choice is binary - between *'yes'* or *'no'*, *'malware'* or *'software'*, *'malignant tumor'* or *'benign tumor'*. Multi-class problems, like the one which concerns this paper, involve more than two classes. However, the same principles as before apply.

II. RELATED WORK

The whole endeavour rests upon previous work by [2] in which a concerted effort is made to turn informational bulletins sourced from the Ministry of Internal Affairs of the Republic of North Macedonia's website into a usable, interactive crime map showing all criminal activities made across the country in the past thirty days. This warrants text analysis of the bulletins detailing reported crimes in a given day in order to extract relevant facts i.e. the nature of the crime, broadly classified into seven categories: **weapons** (in the Macedonian implementation referred to as пиштол) (illegal possession and distribution of firearms or causing reckless endangerment while wielding a weapon), **violence** (насилство) (physical assaults), **theft** (кражба) (aggravated and armed robberies including burglaries), **documents** (документи) (paperwork forging), **drugs** (дрога) (illegal possession and distribution of narcotic substances), **traffic** (сообраќај) (traffic violations and reckless driving) and **other** (друго) (entries that do not fit into the preceding categories). Additionally, one must know the general location of the event (city); if the report contains a street address or village, they are combined to produce a more accurate location description. Values indicating the latitude and longitude needed for map construction are generated separately from the location. The dates of publication of the bulletin and when the crime has been committed are also recorded. The resulting crime map in question can be appreciated on *http://crimemap.finki.ukim.mk/home/en*.

Information extraction is done using basic keyword tagging (drawing conclusions based on the absence or presence of certain terms). Namely, the type of crime is determined by taking into consideration the terminology used in North Macedonia's Criminal Code and matching occurrences of it in the reports themselves. This proves of immense help when the official criminal offense is given in quotes, however many entries diverge from such conventions. In those cases, further processing based on a self-concocted dictionary of common words used to explain criminal deeds is made (for example, one would look for the word *'fight'* and relate the entry to **violence** or stumble upon the word *'shot'* and assume that it describes an event belonging to the **weapons** class).

As it pertains to the general field of forensics using machine learning, many approaches have been used to tackle crime statistics from a number of sources. For

example, [3] have employed similar methods in order to classify cybercrime offenses, proposing using Naive Bayes for classification and K-means for clustering, having performed TF-IDF feature extraction beforehand, achieving 99% accuracy. Meanwhile, [4] make a comprehensive analysis of Twitter posts, trying to ascertain any criminal sentiments expressed in the tweets and classifying them as concerning or not using a text mining-based approach. With a Random forest classifier, an accuracy of 98.1% is gained. This duo of data mining followed by machine learning is preferred by [5] which make use of WEKA, an open-source data mining software in order to gauge the correlation between violent crime patterns from the Communities and Crime Unnormalized Dataset provided by the University of California-Irvine repository and actual crime statistical data for the state of Mississippi. A linear regression model provides the greatest correlation coefficient as well as the lowest error value.

III. DATA SET SOURCING AND PRE-PROCESSING

It is worth mentioning that the crime map remains an active project maintained by the authors, with entries added on a regular basis. All in all, some 40399 records have been collected dating from 21st June 2011 up until today and counting. Naturally, a narrower range has been selected for initial work. Entries belonging to the period between June 2020 and October 2021 are further isolated, having 2000 distinct data points which are then labelled (that is, a proper class for the type of crime has been given manually). This labelled data set will be taken as the ground truth upon which classifier models are evaluated for fitness.

In regard to natural language processing, a SpaCy [6] library instance is utilized due to its newly found support of the Macedonian language (with a trained *pipeline* - a step-by-step analysis of textual input on multiple linguistic levels- generously provided by the Macedonian software company Netcetera; their efforts are noted in *https://blog.netcetera.com/macedonian-spacy-f3c85484777f*). A SpaCy pipeline object boasts myriad capabilities and features. For the purposes of pre-processing, the following functionalities are used: SpaCy's tokenizer (divides sentences into *tokens* - each representing a word or collection of words that have a unique semantic value; it differs from simple white-space separation due to the injected language sensitivity when the pipeline is trained. Thus, when tokenizing the sentence *'The U.K. stands for The United Kingdom.'*, a good tokenizer would correctly identify *'U.K.'* as one token, but split off the punctuation at the end of the sentence), Parts-of-Speech tagger (identifies *'.'* as punctuation, and *'U.K.'* as a proper noun), Named Entity Recognizer (recognizes entities based on their format or real-world role, so *'U.K.'* would be recognized as a GPE or geopolitical entity, *'$1 billion'* as MONEY or monetary currency, etc.), Rule-based Matcher (looks for tokens which conform to some shape, so *'U.K.'* could be found by searching for the shape *X.X.* in the provided sentence, *X* standing for any capital letter) and lemmatizer (determines the base form of a word; *'be'* is a base form of *'was'* and *'assault'* a base form of *'assaulted'*).

Having said that, each data point goes through a text pre-processing module which renders its structure into a form recommended for more effective model training. First, each report is stripped of extra white-spaces as they are superfluous and given to the tokenizer. Of the individual tokens, those with a punctuation Parts-of-Speech tag are removed. The same applies to so-called stop words (the most frequently used words in the Macedonian language). No doubt all entries contain some sort of date, time of day, or numeric quantity, playing a similar role to the stop words. If The Named Entity Recognizer labels a token as DATE, TIME or CARDINAL accordingly, it is removed. The Rule-based Matcher steps in to remove references of people (both perpetrators and victims are referred to by initials only with their age provided in brackets, so a shape *X.X.(dd)*, *d* being a digit, is filtered for). Finally, a lower-case form of each token (this being done because for all intents and purposes, to a computer, *'Crime'* and *'crime'* are different words and will be treated as such even though it is evident that they refer to the same concept) is provided to the lemmatizer. All surviving lemmas are again joined together with white-spaces to form a "clean" report. Table I shows a sample report that has gone through said transformations.

TABLE I: EXAMPLE OF A DATA ENTRY BEFORE AND AFTER PRE-PROCESSING

Original report	**Transformed report**
СВР Охрид поднесе кривична пријава против Н.В.(18) од Скопје, поради постоење основи на сомнение за сторено кривично дело „загрозување со опасно орудие при тепачка или караница“. Тој, на 27.08.2020 во авто-кампот Љубаништа, физички го нападнал А.Ј.(20) од Охрид, при што користел и остар предмет.	свр охрид поднесе кривичен пријава скопје постоење основ сомнение сторено кривично дело загрозување опасен орудие тепач караница авто камп љубаништа физички нападнал охрид користел остар предмет

IV. TF-IDF VECTORIZATION AND MODEL SELECTION

Next, each report is vectorized using TF-IDF [7]. What intuitively comes to mind when trying to work out the main topic of some textual input is to count the number of occurrences of each word and associate the meanings carried by quantitatively prevalent words with it. When this is done to all reports, one might get a matrix with 2000 rows (one per report) and quite a large number of columns (one per unique word) representing the whole *vocabulary* of the data set. A *feature vector* is one matrix row (a report represented by word counts), *features* being

the words themselves (in subsequent text, ”feature vector”, ”data point”, ”sample” and ”report” are synonymous - the text from one report is a data point/ sample from the data set; it is later transformed into a feature vector). It is assumed that the matrix is largely sparse with many null entries. Memory efficiency aside, a more serious problem arises: longer reports will, on average, have higher counts per word and be more closely associated with a topic than shorter ones, even though both seem to describe similar events. Furthermore, if stop words are not removed beforehand, this would lead to disproportionate counts for words such as *'the'*, *'it'* or *'an'*, themselves being of no positive contribution towards determining the topic. Punctuation and commonly used named entities put an additional burden on the vocabulary with unnecessary columns.

To alleviate the issue of long reports, one switches to working with relative scaling, so the count of each word in a given report is divided by the total number of words in it i.e. their term frequency is calculated (this being the preferred method of Scikit-learn [8] - a library specifically designed with machine learning in mind and which will be used to import modules in order to implement what has been discussed). Similarly, one could calculate the logarithm of a word count (and add 1 to avoid having to take the logarithm of zero), essentially smoothing the influence of irrelevant, but frequent words.

Another refinement on top of term frequency is to down-scale weights for words that occur in many reports and are therefore less informative than those that occur only in a smaller portion and might indicate topic-specific terms. In Scikit-learn, this is done by taking the logarithm of the ratio between the total number of reports in the data set and the number of reports which contain some word w (adding a 1 to this value would make sure that words occurring in all reports are not entirely ignored).

Initially, the data set is split into a training set (80%) and a testing set (20%). The 1600 reports belonging to the training set are then fitted to a CountVectorizer followed by a TfidfTransformer (analogous to the previously described process). A vocabulary of 4912 unique words is generated. Meanwhile, their labels go through a LabelEncoder, because they are non-numerical as is and models handle numerical, categorical values better (this can be omitted, however).

Since the classification is made with discrete features, a natural choice for a model is multinomial Naive Bayes. The multinomial distribution normally requires integer feature counts, but in practice, fractional counts such as TF-IDF may also work. It is widely used in natural language processing contexts (Bayesian learning approaches have been used traditionally to filter ”spam” e-mails from ”ham” or categorize products in e-commerce settings based on description alone). The naivete of a Bayesian model stems from the fact that it presupposes conditional independence of features in a feature vector i.e. under the assumption that a given report belongs to class C, the intrinsic syntactic or semantic dependence between the

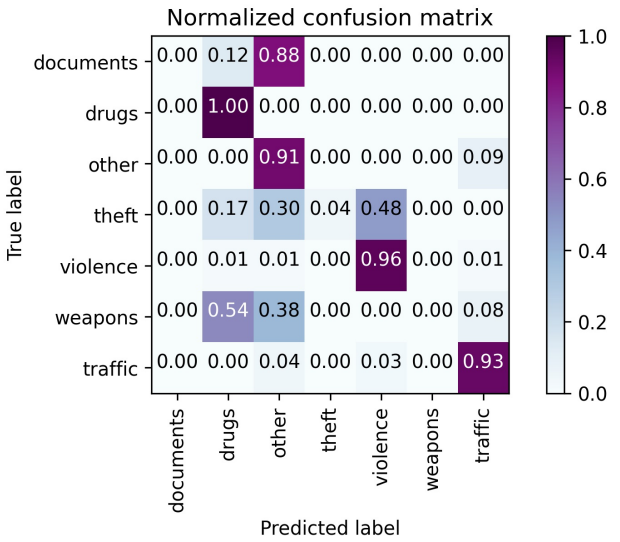

Figure 1: Confusion matrix generated by default-parameter multinomial Naive Bayes

words is ”explained away” by the class. That the word *'stabbed'* warrants occurrence of *'knife'* down the road is reduced to the probability that *'stabbed'* is present at all in a **violence** or **weapons** report (the same goes for *'knife'* and all the other words in that particular vector). These probabilities are multiplied, giving the probability that the report is part of class C. This is done for all possible values of C. Ultimately, the class with the highest probability will be chosen as the classifier's final label. While a MultinomialNB instance with default parameters is fitted to the training vectors and corresponding labels, the reports which are a part of the testing set go through CountVectorizer and are transformed only (not fitted) by TfidfTransformer. A confusion matrix in Fig. 1 shows the predicted labels and true labels accordingly. The values are normalized i.e. they closely correspond to the recall metric: if the predicted label of a data point equals that of the true label, then that point is considered a *true positive* with regard to the label examined. The others are mismatches and therefore constitute *false negatives*. The recall calculates the ratio between true positives and total predictions made (true positives + false negatives). In other words, in Fig. 1, 91% of points that actually belong to the class **other** are labelled by the classifier as such.

Unfortunately, initial results appear less than satisfactory. For classes **documents** and **weapons**, the classifier completely misses the mark and when unsure, defaults to the class **other** since the entries tend to vary most in its midst. **Theft** reports share a similar fate while all others give a surprisingly solid recall rate. The confusion between **drugs** and **weapons** can be explained by inter-sectional language (drug deals tend to be made between armed individuals); the same could be interpolated for false classifications of **theft** as **violence** (some sort of physical violence is needed when a planned theft is interrupted).

Report	True Label	Predicted Label
свр охрид поднесе кривичен пријава двајца малолетник село елшани малолетник охрид постоење основан сомнение сторени кривичен дела тешка кражба кривичен дело тешка кражба обид пријавените период градинка населба лескајца охридско употреба физичка сил насилен влегле одзема предмет	кражба	насилство
полициски службеник единица внатрешен работи свр велес подрачје велес лиши слобода велес била распишан национален потерница издржување казна затвор наредба основен кривичен суд скопје сторени кривичен дела недозволен изработување држење тргување оружје распрскувачки материја злосторничко здружување шлионажа измама прикривање уцена бил приведен полициска стан велес бил спроведен казнено поправен установа идризово издржување казната затвор	пиштол	друго
граничен премин деве баир пасош контролабило утврден извршен бришење штембил патната исправа р.албанија постоење основ сомнение сторено кривично дело фалсификување исправа бил лишен слобода предаден полициски службеник полициски стан крива паланка понатамошен постапка	документи	друго

Figure 2: Classification errors using default-parameter multinomial Naive Bayes

Fig. 2 provides examples where some discrepancy between true and predicted labels exists. A huge issue previously unmentioned is the unbalanced state of the data set. For it to be balanced, all classes need to share an equal representation. As it stands, the distribution of classes is as follows: 27.45% **other**, 18.80% **traffic**, 18.60% **drugs**, 21.25% **violence**, 7.25% **theft**, 2.05% **documents**, 4.60% **weapons**.

Taking these statistics into consideration, the confusion matrix seems to indicate a lack of data points for the three least represented classes in the data set. This will be remediated in further trials; there are no attempts to improve the classifier itself since Naive Bayes classifiers have a very limited parameter set and hyper-parameter tuning is unlikely to lead to a performance boost. TF-IDF already uses log probabilities, so there's no small-number problem. The test set itself has no zero-observation issue (all classes are present with some samples in the testing set, confirmed by looking at the support column of the classification support metric). It *does* confirm the imbalance seen in the data set, though: there are just 8 data points present for the class **documents** compared to the 118 for **other**. Pre-processing steps have already been taken in regard to the textual input itself. [9] propose some relief by smoothing out word counts based on the prominence of the class in the data set; when calculating the probability of a word occurring in a text of a given class *C*, a weighted average of the word count divided by the total number of occurrences of all words in reports of class *C* is used instead, with promising results.

Focus consequently shifts from Naive Bayes to Support Vector Machines (SVMs). SVMs have enjoyed much adoration for classification tasks because they provide robust decision boundaries between categories. They treat feature vectors as being sprawled across some vector space and then try to find a margin (linear or otherwise) which will not only separate classes but be as further from actual data points as possible, building a "buffer zone" with which to reduce possible erroneous classification on a testing set (it is better to have some physical distance between the boundary than thread the needle with one which is very close to a vector as minimal disturbances might result in the unwanted crossing of the boundary).

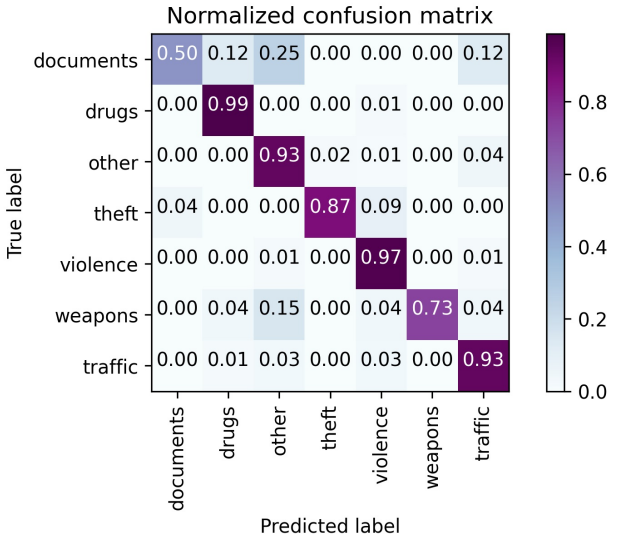

Figure 3: Confusion matrix generated by Support Vector Machine using Stochastic Gradient Descent

Those vectors closest to the margins "support" the decision of the model, hence the name. As per [10], SVMs prove to be suitable for text classification due to their handling of high-dimensional feature vectors well, especially when most features have some relevancy to the prediction (text pre-processing assures this is the case). As mentioned, even though each feature vector has a shape 1×4912, most entries are zeroes and SVMs are suited for such "dense concepts, sparse instances" formats. Finally, problems belonging to this type turn out to be linearly separable, so simpler models could account for a vast array of applications.

Returning to practical implementation, a linear SVM using Stochastic Gradient Descent is chosen (the model updates its parameter values after going through one sample at a time (or a small subset of samples - a *batch*) instead of taking the whole data set and *then* opting to tweak them. This results in a model that jumps quickly and easily through possible values for parameters while at the same time getting a correct rough estimate (order of magnitude, sign, etc.) with few samples. Therefore, an SGDClassifier is initialized with a *loss* parameter equal to *'hinge'* (it gives a linear SVM), *penalty* equal to *'l2'* (prefers simpler models when possible), *alpha* equal to *'0.001'* (how hard to penalize), a *random_state* equal to a random integer (e.g. *'42'*) so the order of the samples in the training set is reproducible across function calls because *max_iter* is set to *'5'* (how many times to call the fitting function, equivalent to the number of times to pass over the whole training data - no. of *epochs*). The resulting confusion matrix is shown in Fig. 3.

Indeed, a much better recall rate is provided for all classes compared to Naive Bayes, although **documents** and **weapons** could use more samples to differentiate them better from other types of crime. Consequently, this setup is chosen for further improvement by hyper-parameter tuning using Grid Search. This refers to the process of giving a range of values for parameters and

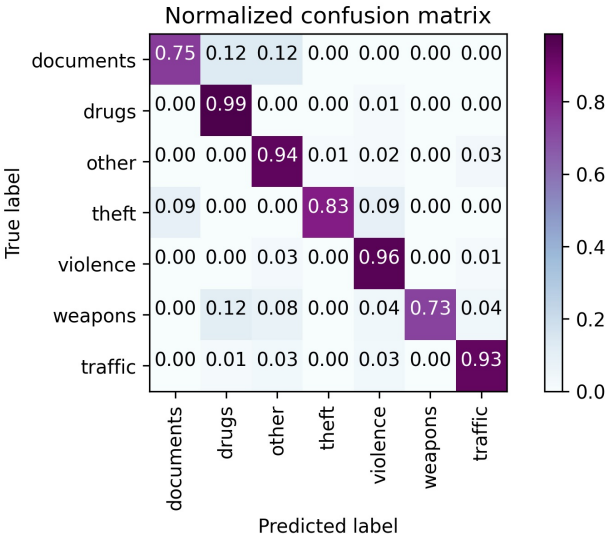

Figure 4: Confusion matrix generated by Support Vector Machine with Grid Search-tuned hyper-parameters

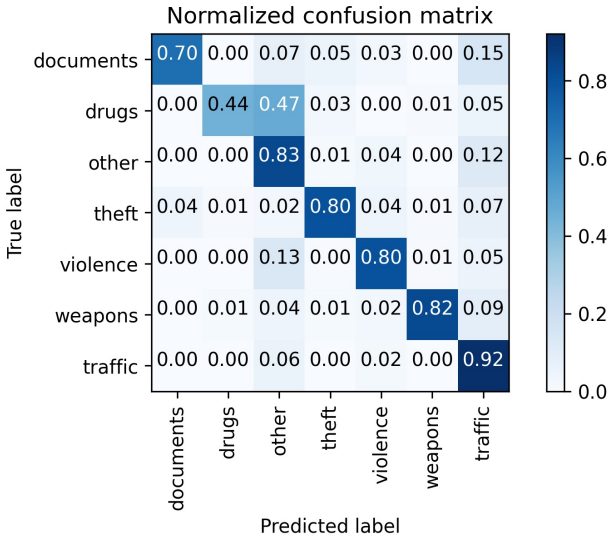

Figure 5: Confusion matrix from labels provided by crime map authors for the same data set

letting the model try out various combinations from the provided lists, eventually giving a configuration that has performed best (each configuration is tested on a subset of the provided data set using a five-fold cross-validation technique: the data set is randomly split five-ways, one of those splits (*folds*) is used as a testing set, the others are used for training; some accuracy score is achieved and the model is discarded. This is done five times so each fold gets to be a test set. The evaluation score of the model is averaged out from those five accuracy scores).

The CountVectorizer's *max_df* parameter is refined (this is a threshold value; words that have a document frequency strictly higher than the given threshold are ignored) as well as *ngram_range* (n-grams are collections of *n* consecutive words; unigrams are individual words, bigrams are formed by two words found next to each other in a report, etc. The parameter value indicates up to which *n* to build. A vocabulary is formed from all n-grams, substantially increasing its size, but hopefully, its predictive abilities to boot as words reveal more information when taken holistically and reflect their linguistic connections). The TfidfTransformer is tested upon its *use_idf* parameter (its role is self-evident) while the SGDClassifier is altered for better *loss*, *alpha* and *penalty* values. The resulting output showing the best parameters for the model is shown below:

```
{'sgdc__alpha': 0.0001, 'sgdc__loss':
'hinge', 'sgdc__penalty':
'l1', 'tfidf__use_idf':
True, 'vect__max_df': 0.75,
'vect__ngram_range': (1, 2)}.
```

New instances of the counter, vectorizer and classifier are made, initialized with the values provided by Grid Search and a new confusion matrix is constructed (Fig. 4).
As expected, the model has tuned its parameters in such a way as to provide minimal errors across classes, which has the effect of shifting some margins slightly and contributing to the major improvement of recall for **documents** (on the back of some minimal loss of recall in **theft** for example; this is a game of trade-offs after all).

V. COMPARISON OF MODEL AND CRIME MAP CONFUSION MATRICES

All that is left to do with regard to the current data set is to compare the matrix from Fig. 4 with the one made using labels for the samples made by [2] (Fig. 5) and determine whether there exists improvement in correct classification of crimes using the methods so far outlined (by comparing them with the ground truth as has been done with the classifiers).
In all classes except **weapons**, the model outperforms and gives better recall rates- substantially in the case of **drugs** (125% relative increase in recall) and marginally in others (below 20% relative increase). This is no cause for concern as the matrix itself appears quite satisfactory, so this should be considered an attempt at bettering an already solid outcome.

VI. CREATING A BALANCED DATA SET. SPLITTING CLASS 'OTHER' INTO MULTIPLE CLASSES

Since the matter of an unbalanced data set has been raised, it is of great interest to explore how dire of an issue it really is and test the Grid-Search SVM against one in which said issue has been mitigated. While at it, the class **other** is analysed and additional four types of crime extrapolated in its stead: **COVID-19** (breaking safety protocols by disobeying movement restrictions while positive for the SARS-CoV-2 virus; vaccination certificate forgeries fall under **documents**), **dog attacks** (there are overwhelming reports of individuals bitten by stray dogs), **disappearances** (cases of persons going missing or running away- not a crime per se, but such reports

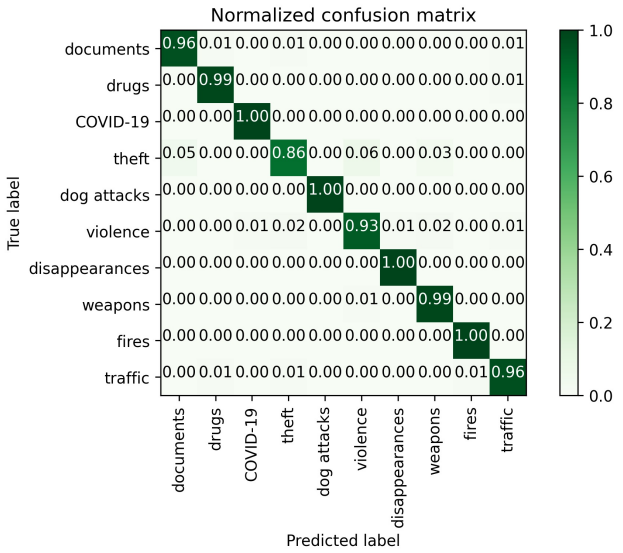

Figure 6: Confusion matrix from balanced data set by Support Vector Machine with Grid Search-tuned hyper-parameters

are common enough as to justify a separate category), **fires** (fires breaking out, intentionally or unintentionally e.g. as a result of a car crash).

Both old and new types of crimes will have around 370 different samples representing them in the new data set in order to secure proportional involvement. It might come as no surprise that a wider time period needs to be considered to achieve that. Therefore, entries spanning from June 2020 up until June 2022 are taken. If some class still lacks samples, even earlier records are added. Each new sample is manually labelled with the appropriate class and the whole data set is randomly permuted in order to break apart blocks of entries with the same label (to prevent losing the balance when splitting the set into training and testing sets, stratified splitting in which the proportion of samples from each class mirrors that of their distribution in the larger set is used).

In the end, a data set of 3694 unique data points (pre-processed into TF-IDF feature vectors) is given to the Grid-Search SVM classifier. A vocabulary composed of 32357 unigrams and bigrams is returned.

The results generated are near-perfect, as can be seen in Fig. 6; all but one of the recall rates hit the high nineties (the somewhat smaller percentage for **theft** could be due to some ambiguities regarding certain crimes. Cases where individuals have monetarily damaged institutions by deliberately withholding expenses and pocketing the difference themselves could very well be categorized as **theft** and **documents** (since the crime technically involves forging official documents). The same argument could be said for cases involving counterfeit money spending (in essence a type of forgery and apt for a **documents** label) and yet appropriate for **theft** as someone was deprived of the value of the monetary transaction. Nevertheless, trying to catch such slight variations in model predictions borders on overfitting the data- thus, no further corrections to the model itself are undertaken.

VII. CONCLUSION AND FUTURE WORK

By conforming to the best practices for handling textual information and careful analysis of model behaviour, a non-negligible improvement towards a successful classification of crimes is made, once again confirming the beneficial nature of machine learning techniques towards such tasks.

With much confidence, these same principles could be put to use in order to improve other aspects concerning the construction of a crime map. For example, more accurate location extraction could be made by semantic analysis of reports (natural language processing comes at hand immediately) so the place of birth of a suspect or victim is distinguished from the actual location of the crime in the eyes of the model. Such an undertaking might require training a Named Entity Recognizer from scratch, feeding into it a whole range of geographical locations (SpaCy's Macedonian pipeline is partially trained on Wikipedia texts written in Macedonian, so one would suppose that some geography-related articles are covered, however, it lacks much of the desired sensitivity to geolocations; this, in essence, means that further training and manual labelling of data is required, especially for street names, hospitals, banks, border crossings, etc.). Such a gargantuan undertaking nearing the informational scope of Google Maps is, of course, in need of more care and attention by a larger group of members and will be considered once those resources are secured.

REFERENCES

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[2] M. Jovanovik, D. Trajanov, I. Mishkovski, and D. Temelkovski, "Towards open data in macedonia: Crime map based on ministry of internal affairs' bulletins," *The 9th Conference for Informatics and Information Technology (CIIT)*, 2012.

[3] R. Ch, T. R. Gadekallu, M. H. Abidi, and A. Al-Ahmari, "Computational system to classify cyber crime offenses using machine learning," *Sustainability*, vol. 12, no. 10, p. 4087, 2020.

[4] S. Lal, L. Tiwari, R. Ranjan, A. Verma, N. Sardana, and R. Mourya, "Analysis and classification of crime tweets," *Procedia computer science*, vol. 167, pp. 1911–1919, 2020.

[5] L. McClendon and N. Meghanathan, "Using machine learning algorithms to analyze crime data," *Machine Learning and Applications: An International Journal (MLAIJ)*, vol. 2, no. 1, pp. 1–12, 2015.

[6] Y. Vasiliev, *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press, 2020.

[7] A. Rajaraman and J. D. Ullman, *Data Mining*. Cambridge University Press, 2011, p. 1–17.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[9] E. Frank and R. R. Bouckaert, "Naive bayes for text classification with unbalanced classes," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2006, pp. 503–510.

[10] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.