**Analiza sentymentu na podstawie wpisów z Twittera**

**Jakub Soborski**

**Opis danych wejściowych**

W moim projekcie wykorzystałem zbiór danych: <https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis>. Dataset jest całkiem spory ponieważ składa się z prawie 70000 rzędów, ma jednak tylko 3 kolumny. Dwie z nich to predykatory: text – zawiera wpisy użytkowników, oraz game – mówi nam z threada jakiej gry pochodzi wiadomość. Mamy również kolumnę sentiment, która mówi nam czy dana wiadomość jest nacechowana negatywnie, pozytywnie lub neutralnie. Zbiór danych był raczej dobrze zbalansowany tj. nie było dużych dysproporcji pomiędzy różnymi labelami ale jakieś były co musiało być uwzględnione w doborze metryki treningowej. Celem modelu będzie nauczenie się wyłapywania pozytywnych i negatywnych emocji z tekstu.

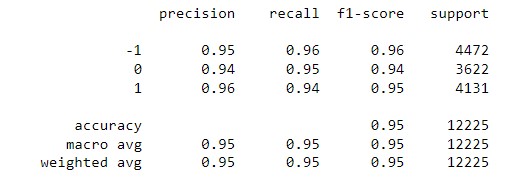
**Strategia preprocessingu i podziału danych**

Na początku rozpocząłem od kilku prostych wizualizacji danych. To pozwoliło mi stwierdzić ze gra z której wiadomość pochodzi może mieć wpływ na potencjalny wynik modelu tj. istnieją gry generujące więcej pozytywnych/negatywnych emocji niż inne. Następnie połączyłem dwa predykaty w jeden uwzględniając thread na końcu wiadomośći w kolumnie text. Następnie musiałem wykonać sporo operacji na kolumnie text ponieważ aby model mógł nauczyć się rozumieć tekst to trzeba przedstawić mu go w postaci liczb. Na tym etapie musimy zastosować którąś z metod Bag of Words, ja zdecydowałem się na tfidf, która dodatkowo bierze pod uwagę częstośc występowania słów w dokumencie tj. przypisuje większe wagi słowom które występują często w danym dokumencie ale rzadziej w innych dokumentach w datasecie. To pozwala nam na wykrycie słów kluczowych w danym kontekście. Dodatkowo aby zmniejszyć wymiarowość danych w tfidf biorę pod uwagę tylko te słowa, które występują więcej niż 5 razy w tekście to pozwala mi na redukcje liczby słów w alfabecie z prawie 30000 do 8000 i znacząco usprawnia działanie większości modeli statystycznych, które słabo sobie raczą gdy n = p. Dodatkowo wcześniej usuwam z wiadomości niepotrzebne informacje takie jak url, stopwordsy, nicki użytkowników itd. I finalnie wykonuje lemmatyzacje, tak aby traktować tak samo słowa które są odmienione np. walk i walked, ale także korzystając z nltk.pos\_tag kojarząc słowa patrzę na to jaką część mowy reprezentują aby podczas lemmatyzacji nie utracić za dużo informacji. Podział danych dokonuję zwykłym train\_test\_splitem tylko na 2 zbiory danych w stosunku 80/20. Zbiór testowy będzie zbiorem walidacyjnym na którym będą określane finalne metryki modelu (nie bierze on udziału w treningu gdyż tam korzystam z cv). Dane rozdzielam tak aby zachować rzeczywiste proporcje sentymentów do tego używam argumentu stratify = y.

**Strategia doboru modelu**

W przypadku tego problemu dobór modelu był prosty z faktu tego że preprocessing był efektywny oraz niestety z ograniczeń wynikających z faktu że trenowanie modelu było czasochłonne (40000 rekordów po 8000 kolumn) nie udało mi się przeprowadzić kompleksowego doboru hiperparametrów modelów. Jednak prosty model SVC z jądrem wielomianowym stopnia 3 prezentował się wystarczająco dobrze i zdecydowałem się na użycie tego modelu (95% accuracy na danych treningowych). Rozważałem jeszcze Random Forest i być może któreś z metod boostingowych, jednak swoich bazowych konfiguracjach nie radziły one sobie tak dobrze jak SVC a ich czas szkolenia był również długi co uniemożliwiało wyszukanie odpowiednich hiperparametrów tak aby pokonać model SVC. Metryką z jakiej korzystałem w trakcie treningu było balanced\_accuracy, które zwraca średnią z accuracy każdej klasy wybrałem tą metrykę ponieważ problem nie faworyzuje ani precyzji ani czułości a także jest lekko niezbalansowany.

**Wyniki testowe**

Na danych treningowych mój prosty model radził sobie bardzo dobrze.

Wyniki są bardzo zbliżone do zestawu treningowego co oznacza że uniknęliśmy overfittingu. Więcej wyników na zbiorze testowym znajduje się w notebooku z projektem.

**Jak używać modelu**

Model radzi sobie zarówno z danymi które mają thread z którego pochodzą jak i nie. W notebooku zostało wykazane że obecność kolumny game nie ma aż takiego wpływu na performance modelu (różnica w accuracy na danych testowych jest na poziomie 1%). Do modelu można także przekazywać pojedyncze wiadomości (jako stringi) dla nich też jest napisana obsługa.

**Dalsze kroki**

W dalszych krokach na pewno przydałoby się poświęcić więcej czasu na znalezienie odpowiednich hiperparamterów dla modelu lub może znalezienia modelu, który radzi sobie jeszcze lepiej. Przydałoby się również pozyskać więcej danych gdyż dla datasetu z tak dużą liczbą kolumn nasza aktualna liczba danych dla wielu bardziej skomplikowanych modeli może być nie wystarczająca aby wyjaśnić całą wariancję.