Raport Projektu:

„Przewidywanie indeksu EURO STOXX 50 na

na podstawie danych numerycznych oraz reprezentacji tekstu”

Jako projekt chciałem przynajmniej w części powtórzyć pracę „*Explainable Deep Learning for Thai Stock Market*”, w której porównywane były wyniki modeli korzystających tylko z danych tekstowych, tylko z danych numerycznych oraz model korzystający zarówno z danych numerycznych jak i tekstowych.

Zbiór danych:

Dane z historyczne z giełdy pobrałem bezpośrednio z serwisu MarketWatch. Do danych numerycznych dodaje następnie wskaźniki z analizy technicznej giełdy, za pomocą biblioteki ta w pythonie. Problemy pojawiły się przy zdobyciu danych tekstowych, dla których nie ma łatwo dostępnego zbioru danych. W tym przypadku przy pomocy web-scrappingu (skrypt którego użyłem jest w pliku scrap.py na githubie) uzyskałem nagłówki artykułów o rynku europejskim z serwisu Reuters.com z lat 2009-2021 (docelowo pracowałem na zbiorze danych z 2010-2021)

Modele:

Udało mi się stworzyć 3 modele, model tylko na danych tekstowych (w pliku utils.py zapisany jako klasa TextOnlyGRU), tylko dane numeryczne (LSTMNumeric) oraz zarówno tekstowe jak i numeryczne (TextAndNumeric).

1. **TextOnlyGRU:**

   Model przed wpuszczeniem danych tekstowych najpierw przekształca je na tensory za pośrednictwem clasy Lang która pojawiła się w notebookach z ćwiczeń, oraz funkcji TensorFromSentence również z tego samego notebooka. Następnie dane przepuszczane są przez warstwę Embedding, do 2-warstwowego GRU, na końcu przechodzą przez sieć fully connected.

2. **LSTMNumeric:**

   Dane które wchodzą jako dane numeryczne, to przeskalowane dane z 5 dni razem ze wskaźnikami z analizy technicznej. Przechodzą przez 3-warstwowy LSTM następnie są „spłaszczane” i wpuszczane do sieci fully connected z dropoutem.

3. **TextAndNumeric:**

   Podobnie jak poprzednio, dane tekstowe są przerabiane na tensory jak w modelu TextOnlyGRU, dane numeryczne to również przeskalowane dane (razem ze wskaźnikami z analizy technicznej) z 5 dni. Dane tekstowe przechodzą przez warstwę Embedding, następnie do 2-warstwowego GRU. Dane Numeryczne wchodzą do 2-warstwowego bidirectional LSTM, następnie przechodzą przez Linear który przerzuca dane z całego LSTM (10*hidden ponieważ LSTM jest bidirectional i mamy dane z 5 dni) na rozmiar 1xhidden. W tym momencie konkatenujemy wektor z GRU oraz z LSTM + Linear do wektora o rozmiarze 2xhidden który przechodzi przez sieć fully connected z dropoutem.

W każdym modelu korzystam z optymalizatora Adam, oraz z Exponential Learning Rate Scheduler, a jako funkcji kosztu używam Mean Square Error. Dane rozłożyłem na zbiór treningowy (od 2010-2017), walidacyjny (2017-2019) oraz testowy (2019-2021). Każdy model ma przewidzieć zwrot z następnego dnia, zadany wzorem:

$$y_t = (c_{t+1} - c_t)/c_t$$

Gdzie yt to zwrot, ct+1 to cena zamknięcia z następnego dnia, a ct to cena zamknięcia z dnia t. Przy przewidywaniach giełdy z tego co zauważyłem najczęściej korzysta się z takiej funkcji return, ponieważ wartości z niej (w większości przypadków) układają się w Gaussa.

Modele oceniam za pomocą funkcji kosztu, funkcji accuracy zadanej wzorem:

$$Accuracy\ \% = \frac{\sum_{t=1}^{t=n}(h_t)}{n} \times 100\ \% \quad [14] \qquad (2)$$

$$h_t = \begin{cases} 1 & \text{if } \hat{y}_t \text{ and } y_t \text{ are both positive or negative,} \\ 0 & \text{other wise,} \end{cases}$$

Gdzie y z daszkiem to predykcja, a yt to faktyczny zwrot. Korzystam również przy testowaniu z funkcji Hit Profit zadanej wzorem:

$$Hit\ profit = \sum_{t=1}^{t=n}(2h_t - 1)\, y_t$$

Gdyż taka funkcja ocenia zyski modelu niezależnie od przyjętej strategii tradingowej. Wszystkie te 3 funkcje wziąłem bezpośrednio z pracy na której się opierałem.

Wyniki:

Model	**Loss**				**Accuracy (%)**				**Hit profit(%)**			
	2019	2020	2021	Average	2019	2020	2021	Average	2019	2020	2021	Average
Text Only	0,438	**2,437**	0,586	1,153667	45,6	49,41	47,91	47,64	-4,375	**24,061**	5,428	8,371333
Numeric Only	**0,401**	2,442	**0,539**	**1,127333**	**56,48**	52,53	**57,91**	**55,64**	**5,689**	-12,244	7,666	0,370333
Text + Numeric	0,489	2,502	0,568	1,186333	49,37	**56,03**	52,5	52,63333	-0,861	22,796	**32,509**	**18,148**

Podsumowanie:

Zarówno w tym projekcie, jak i w pracy naukowej jeżeli chodzi o hit profit najlepszy wynik osiągnął model na danych tekstowych oraz numerycznych. Uczenie modeli z elementami tekstowymi było dla mnie bardzo problematyczne, ponieważ nie potrafiłem ułożyć danych w taki sposób by użyć batch_size większy niż 1 (przez różne długości tekstu), przez co wydaje mi się że te wyniki dałoby się polepszyć, szczególnie że zbiór danych był dużo mniejszy (zbiór nagłówków był dość mały w porównaniu do danych z pracy naukowej [około 14 tys do około 600 tys]. Cały kod znajduje się oczywiście w repozytorium w folderze projekt.