

OTDK-dolgozat

2025



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
Gazdaság- és Társadalomtudományi Kar
Pénzügyek tanszék

**Idősoros mély neurális háló és Mixture Density Network alapú
részvényárfolyam előrejelzés ensemble modellek alkalmazásával**

**Stock Price Prediction using Time-series Deep Neural Networks and
Mixture Density Network-Based Ensemble Models**

Készítette: Krebs Áron Antal
Mérnökinformatika szakos hallgató

Konzulens: Békési Gergő Bendegúz
Doktorandusz

Kézirat lezárása: 2024. december 30.

Tartalomjegyzék

ABSZTRAKT	5
ABSTRACT	6
1. BEVEZETÉS.....	7
1.1. ÁRFOLYAM ELŐREJELZÉS A SZAKIRODALOMBAN.....	7
1.2. KIHÍVÁSOK.....	11
1.3. AZ ELEMZETT RÉSZVÉNYRŐL	11
2. NEURÁLIS HÁLÓK	12
2.1. NEURÁLIS HÁLÓK ÁLTALÁNOSÁGBAN	12
2.2. AKTIVÁCIÓS FÜGGVÉNYEK.....	13
2.3. DENSE RÉTEG	15
2.4. DROPOUT RÉTEG	15
2.5. RNN RÉTEG.....	16
2.6. LSTM RÉTEG	16
2.6.1. Kapuk típusai.....	16
2.7. GRU RÉTEG.....	17
2.8. MDN RÉTEG.....	18
2.9. SELF-ATTENTION RÉTEG.....	21
2.10. ENSEMBLE MODELLEK	22
3. A NEURÁLIS HÁLÓ ADATAINAK KEZELÉSE	23
3.1. TECHNIKAI INDIKÁTOROK	23
3.2. ADATOK ELŐFELDOLGOZÁSA.....	26
3.3. TARGET DATA KIVÁLASZTÁSA	27
3.4. TRAIN, TESZT, VALIDATIONAL ADATHALMAZOK FELOSZTÁSA.....	28
3.5. ADATOK STANDARDIZÁLÁSA	28
3.6. ADAT SZEKVENCIAK	29
4. MDN RÉTEG OUTPUTJÁNAK FELDOLGOZÁSA.....	30
4.1. KUMULÁLT CDF FÜGGVÉNY BECSLÉS	30
4.2. SÚLYOZOTT VÁRHATÓ ÉRTÉK BECSLÉS.....	31
4.3. SÚLYOZOTT VARIANCIA BECSLÉS	31
4.4. 1-SIGMA INTERVALLUM	32
4.5. 2-SIGMA INTERVALLUM	33
5. BACKTESTING	34
5.1. TELJESÍTMÉNY MUTATÓK.....	35

6.	KERESKEDÉSI STRATÉGIÁK.....	36
6.1.	SÚLYOZOTT VÁRHATÓ ÉRTÉK, SÚLYOZOTT VARIANCIA HASZNÁLATA	36
6.2.	KUMULÁLT CDF HASZNÁLATA	36
6.3.	SÚLYOZOTT ÁTLAG ÉS 1-SIGMA INTERVALLUM HASZNÁLATA.....	37
6.4.	STOP LOSS, TAKE PROFIT MEGHATÁROZÁSA	37
6.4.1.	<i>Delta stop loss és take profit.....</i>	<i>37</i>
6.4.2.	<i>1-Sigma intervallum használata</i>	<i>37</i>
6.5.	POZÍCIÓK NYITVATARTÁSA	38
7.	HIPERPARAMÉTER OPTIMALIZÁCIÓ	39
8.	TESZT EREDMÉNYEK.....	40
	REFERENCIA.....	45

ABSZTRAKT

Az utóbbi években egyre nagyobb figyelmet kapott a mesterséges intelligencia alapú eszközök használata a pénzügyi szektorban. A hagyományos piaci elemzéseknél a mesterséges intelligencia lényegesen komplexebb összefüggéseket is figyelembe tud venni, jobban tudja integrálni működésébe az egyes nemlineáris viselkedéseket.

Ezen tanulmány a neurális hálóra épülő részvény előrejelző modellek széleskörű vizsgálatát mutatja be, olyan fejlett architektúrák megvalósítására helyezve a hangsúlyt, mint a Long Short-Term Memory, Gated Recurrent Units, Recurrent Neural Network, self-attention, és Mixture Density Network megoldások. Az első két említett architektúrán alapú modellek kiválóan megtalálják az időbeli függőségeket az adatokban, jobban, mint egy hagyományos Recurrent Neural Network, ami azt jelenti, hogy a múltbéli függőségeket a jelenben is tudják alkalmazni részvény árfolyam előrejelzésre.

A Mixture Density Network használatával a modell képes a pontbecslés helyett (például következő nap konkrét záróára) a lehetséges jövőbeli árak eloszlását előrejelezni. Az eloszlásnak köszönhetően a bizonytalanságról is kapunk képet, ami lehetővé teszi a piac árnyaltabb megfigyelését, és segíti a jobb kockázatkezelés alkalmazását.

Tanulmányom újszerűsége, hogy az említett modelleket, nemcsak önmagukban használtam fel, hanem a legkülönbözőbb kombinációkban sorosan és párhuzamosan is. Ezáltal meg tudtam vizsgálni, hogy a modellek egymásra hatása, hogyan segíti az előrejelzést. Szintén egyedi, hogy az Mixture Density Network réteget a logaritmikus hozam jóslásával alkalmaztam. Az említett réteg kimeneteit a kapcsolódó szakirodalom jelen álláshoz képest újszerű módon dolgoztam fel. Ennek köszönhetően több megalkotott modellem sharpe rátája is meghaladja a 0,4 értéket, ezzel megelőzve az S&P500 indexet, versenyképes befektetési alternatívaként megjelenve.

ABSTRACT

In recent years, the use of artificial intelligence-based tools has gained increasing attention in the financial sector. Compared to traditional market analysis, artificial intelligence can account for significantly more complex interrelationships and better integrate nonlinear behaviors into its operations.

This study presents an extensive examination of stock price forecasting models based on neural networks, focusing on the implementation of advanced architectures, such as Long Short-Term Memory, Gated Recurrent Units, Recurrent Neural Networks, self-attention, and Mixture Density Network solutions. Models based on the first two mentioned architectures excel at identifying temporal dependencies in the data, surpassing traditional Recurrent Neural Networks, meaning that they can apply past dependencies to predict current stock prices.

Using the mixture density network, the model is capable of predicting the distribution of possible future prices rather than a point estimate (e.g., the exact closing price for the next day). The distribution provides insight into uncertainty, enabling more nuanced observations of the market and assisting in the application of better risk management practices.

The novelty of my study lies in the fact that these models were not only used independently but also in various combinations, both sequentially and in parallel. This allowed me to examine how the interaction between the models aids forecasting. Another unique aspect is that I applied the Mixture Density Network layer to predict logarithmic returns. The outputs of this layer were processed in novel ways compared to the current state of the literature. As a result, several of the models I developed achieved a Sharpe ratio exceeding 0.4, outperforming the S&P 500 index and presenting themselves as competitive investment alternatives.

1. BEVEZETÉS

A részvények a világpiac meghatározó alapelemei. Naponta több száz milliárd dollár értékben kereskednek vele a világ részvénytőzsdéin. A NASDAQ tőzsdén 2024. október 1-én a tíz napra vett átlagos napi volumen 92 milliárd dollár volt. A részvények változása számos tényezőtől függhet, a cég teljesítő képességétől, világgazdasági hírektől, eseményektől és nem utolsósorban a piaci hangulattól.

1.1. Árfolyam előrejelzés a szakirodalomban

A részvényárfolyamok előrejelzése hosszú ideje az egyik legszélesebb körben tanulmányozott téma, amely a történelmi árak, ármozgások vagy trendek elemzésére összpontosít a jövőbeli árak előrejelzése céljából. Számos modell és előrejelzési módszer került kidolgozásra a részvényárak előrejelzésére.

A hagyományos statisztikai időbeli előrejelzési módszerek, mint például az Autoregressive Integrated Moving Average (ARIMA) és annak változatai, még mindig gyakran használatosak hatékonyságuk miatt. Az ARIMA megközelítést széles körben alkalmazzák időbeli adatok elemzésére és előrejelzésére. Bár az ARIMA hasznosnak bizonyult a rövid- és középtávú ártrendek megragadásában, nehezen kezeli a részvénytőzsdéken gyakran megfigyelhető összetett dinamikát és nemlineáris mintázatokat. Ennek következtében a hagyományos árfolyambeclsési rendszerek korlátainak leküzdése érdekében bevezették a gépi tanulási (Machine Learning, ML) és mély tanulási (Deep Learning, DL) technikákat [1].

Az ML megközelítések igen ígéretesek voltak a pénzügyi piacok összetettségének megértésében. Ezt az összetettséget jellemzi a részvényárakat befolyásoló különböző elemek dinamikus kölcsönhatása. A hagyományos ML algoritmusok, mint például az artificial neural network (ANN) [2], a k-nearest neighbors (KNN) [3], a support vector machine (SVM) [4], és az ensemble modellek [5] sikeresen és széles körben alkalmazásra kerültek a részvényár-előrejelző rendszerekben.

Azonban a pénzügyi piacok nemlineáris és időben változó természetének kezeléséhez a mély tanulási megközelítések még hatékonyabbnak bizonyultak. A DL modellek képesek megragadni az összetett időbeli függőségeket és nemlineáris mintázatokat, amelyek gyakoriak a részvényár-mozgásokban. Modern modellek, mint a Convolutional Neural Network (CNN) [6], a Long-Short Term Memory (LSTM) és a bidirectional LSTM (BiLSTM) hálózatok a folytonos függvények közelítését használják, és kevesebb feltételezéssel alkalmazkodnak az

adatokhoz, ezáltal nagyobb pontosságot és hatékonyságot érnek el a nemlineáris problémák megoldásában [1].

Az LSTM hálózatok különösen hatékonyak az időbeli adatok feldolgozásában, mivel képesek hosszabb távú függőségek tanulására és megőrzésére. Például Ghosh és munkatársai [7] bemutatták az LSTM hálózatok és a Random Forest (RF) alkalmazásának hatékonyságát az S&P 500 index részvényárainak irányváltozásainak előrejelzésében az intraday kereskedéshez.

A kutatók számos tanulmányban vizsgálták a neurális hálózati modellek teljesítményét a részvénypiaci előrejelzésekben, különböző architektúrákat és módszereket alkalmazva az előrejelzési pontosság növelése érdekében [8]. Maiti [9] például az LSTM és a Generative Adversarial Networks (GAN) alkalmazását vizsgálta az indiai Nemzeti Tőzsdén a részvényárak előrejelzésére. Tanulmánya kiemeli a DL modellek potenciálját a valós kereskedők gondolkodási folyamatának emulálásában a gördülő szegmentáció technikája révén, amely a tanítási és tesztelési adatkészletet partícionálja, hogy megvizsgálja a különböző intervallum-partíciók hatását az előrejelzési teljesítményre.

De Pauli, Kleina és Bonat [10] öt neurális hálózati architektúra teljesítményét hasonlították össze Brazília B3 tőzsdéjén a legkeresettebb részvények záró árának előrejelzésében. Eredményeik szerint a radiális bázisfüggvény hálózat kivételével minden architektúra megfelelő illeszkedést és ésszerű előrejelzéseket nyújtott, a többrétegű perceptron architektúra pedig kiemelkedett előrejelzési teljesítményével. Ez az összehasonlító elemzés rámutat a megfelelő neurális hálózati architektúrák kiválasztásának fontosságára az előrejelzendő pénzügyi idősorok specifikus jellemzői alapján.

Singh [11] a felügyelt gépi tanulási modellek így többek között az ANN-ek, hatékonyságát vizsgálta az indiai Nifty 50 index előrejelzésében. A tanulmány megállapította, hogy bár a lineáris regresszió és az ANN hasonló előrejelzési eredményeket mutatott, az ANN-ek több időt igényeltek a tanításhoz és validáláshoz. Ez a megfigyelés rámutat a neurális hálózati modellek alkalmazásának kompromisszumára az előrejelzési pontosság és a számítási hatékonyság között a részvénypiaci előrejelzésben.

Ezen túlmenően, hibrid modelleket is fejlesztettek, amelyek különböző előrejelzési modellezési technikákat integrálnak a részvényár-előrejelzések pontosságának és megbízhatóságának javítása érdekében. Ezek a hibrid modellek gyakran kombinálják a DL technikákat más módszerekkel, például a CNN-LSTM, LSTM vagy BiLSTM attention mechanizmusokkal, transformer modellekkel és gráfkonvolúciós neurális hálózatokkal (Graph Convolutional Neural Networks, GraphCNN).

Például Rather és munkatársai [12] kombinálták az RNN-eket az Exponential Smoothing (ES) technikával és az Autoregressive Moving Average (ARMA) modellekkel, hogy kifejlesszenek egy új és robusztus hibrid előrejelzési modellt. Ez a modell kiválóan teljesít az adatok hirtelen változásainak és kiugrásainak (nemlineáris mintázatok) előrejelzésében, hatékonyan javítva az előrejelzési teljesítményt és jelentősen csökkentve az előrejelzési hibákat.

A LSTM struktúra különböző architektúrákban kiváló időbeli modellezési képességeket mutatott. Bár az LSTM egyszerű alkalmazása már eleve felülmúlta a hagyományos gépi tanulási és artificial neural network módszereket, az LSTM más stratégiákkal vagy architektúrákkal való integrálása gyakran mégjobb előrejelzési eredményekhez vezet. Például Abdul Quadir Md és munkatársai [13] egy több rétegű LSTM modellt vezettek be, amely 95,9%-os előrejelzési pontosságot ért el a tanítási adatokon és 98,1%-ot a teszt adatokon, ami kiemeli a több rétegű LSTM struktúra előnyeit [14].

A hibrid modellek közé tartoznak azok is, amelyek az LSTM-eket más módszerekkel kombinálják, mint például attention mechanizmusokkal vagy transformerekkel. Yang és munkatársai [15] egy Hierarchical Transformer alapú Multi-Task Learning módszert terveztek, amely a negyedéves eredményhívások szöveges és hanganyagait használja az S&P 500 részvénytőke jövőbeli rövid és hosszú távú ármegmozgásainak előrejelzésére. A transformerek a hosszú távú függőségek megoldására használják az self-attention mechanizmusokat, ezáltal növelve a hatékonyságot és pontosságot a szekvenciális adatok feldolgozásában, és jelentős áttöréseket érnek el a természetes nyelvfeldolgozás és kapcsolódó területeken. Azonban a nagyon hosszú szekvenciák esetén a transformerek szembe találhatják magukat a nagy számítási és memóriaigényekkel, ami korlátozza gyakorlati alkalmazhatóságukat nagy léptékű alkalmazásokban.

A gráf neurális hálózatok is jelentős figyelmet kaptak a részvénytőke előrejelzésekben, mivel kiválóan teljesítenek a részvénytőke hálózatokban lévő összetett csomóponti kapcsolatok kezelésében és elemzésében, hatékonyan megragadva az információáramlást és az interakciókat. Például Chen és munkatársai [16] egy GCNN-alapú közös előrejelzési modellt fejlesztettek ki, amely kombinálja a célvállalatok és a hozzájuk kapcsolódó vállalatok információit a részvényárak előrejelzéséhez.

Az utóbbi időben a Large Language Modellek, (LLM), mint például a BERT és GPT is alkalmazásra kerültek a részvénytőke előrejelzésben. Ezek a modellek képesek megérteni és generálni természetes nyelvet, így hatékony eszközöket kínálnak a pénzügyi hírek és jelentések szöveges elemzéséhez és előrejelzéséhez. Például Yang és munkatársai [15] egy előtanított

WWM-BERT-et használtak a szöveges információk feldolgozására és tokenbeágyazások generálására.

A fentiek alapján látható, hogy az LSTM, GRU és RNN alapú modellek különösen előnyösek a részvénytípiaci előrejelzésekben, mivel képesek megragadni az időbeli függőségeket és a nemlineáris mintázatokat a pénzügyi időbeli adatokban. Ezen modellek alkalmazása számos tanulmányban bizonyítottan javította az előrejelzési pontosságot [8][14].

Azonban a részvénytípiaci adatok magas volatilitása és zajossága miatt fontos a valószínűségi eloszlások becslése is. Itt jön képbe a Mixture Density Network (MDN), amely képes a valószínűségi eloszlások modellezésére és így a bizonytalanság kezelésére az előrejelzésekben. Az MDN rétegek integrálása a neurális hálózatokba lehetővé teszi az eloszlások előrejelzését a célváltozókra a bemeneti értékek alapján, ami különösen hasznos a pénzügyi előrejelzésekben, ahol a bizonytalanság és a kockázat kezelése kritikus fontosságú.

Ezen munka keretein belül megvizsgálom, hogy hogyan lehet a részvények jövőbeni értékét megjósolni, ismertetem a felmerült kihívásokat, azok lehetséges megoldásait, illetve az alkalmazott technológiákat, módszereket. A részvény árak előrejelzésére neurális hálókat használok. A neurális hálóba LSTM, GRU, RNN, self-attention és MDN rétegeket építettem be különféle kombinációkban. Az MDN réteggel eloszlásokat jósolok a választott target adatra a bemeneti értékek segítségével. Ilyen target adat például az aktuális tőzsde zárási ár és másnapi tőzsde zárási ár hányadosának logaritmus. A bemeneti értékek a részvénynek az aktuális nyitó és az azt megelőző, legmagasabb, legalacsonyabb, valamint a záró árai (open, high, low, close price), és az azokból számított technikai indikátorok. A technikai indikátorok részvénytípiac összetettsége és stochasztikussága végett jelentek meg, ezáltal a részvények jobb analizálhatóságát biztosítják.

Az LSTM és MDN kombinációja lehetővé teszi, hogy az időbeli mintázatok megragadása mellett a bizonytalanságot is figyelembe vegyünk az előrejelzésekben, ami a fent említett szakirodalmak fényében egyedülálló és hatékony megközelítés.

1.2. Kihívások

A neurális hálókbán előjöhethet az eltűnő gradiens probléma, mely megnehezíti a modellek tanítását, ugyanis, ha túl kicsi a gradiens, akkor a modell nem tudja jól visszaterjeszteni a hibát, és ezáltal nem tudja a súlyait a megfelelő mértékben frissíteni. Ez a probléma több forrásból fakadhat, például a túl komplex a modell, vagy a későbbi 2.2 fejezet szerinti ReLU aktivációs függvények használata eredményezhet ilyen jelenséget. Egy másik kihívás az előzőnek a fordítottja, amikor a gradiens olyan mértékben megnő, hogy az már kezelhetetlen, ezt angolul exploding gradients problem-nek hívják. Ilyenkor a modell súlyai túlzottan meg lesznek változtatva.

1.3. Az elemzett részvényről

A tanulmányom során a Sony Corporation japán vállalat által kibocsátott SONY részvényt vizsgáltam. A Sony Corporation, Japán egyik vezető technológiai és szórakoztatóipari vállalata, globálisan is jelentős szereplő. Bár az elemzések gyakran az amerikai piacra fókuszálnak, a japán piac elemzése különleges lehetőséget kínálhat, különösen a Sony-hoz hasonló cégek esetében.

A 2024 első negyedében a Sony összesített árbevétele 3.011 milliárd yenre nőtt, ami 1,6%-os emelkedést jelent az előző év azonos időszakához képest. Az eredmény 10,3%-os növekedése (279,106 millió yen) hatékony költségkezelésre és a bevételek eredményes kihasználására utal.

Az egyes üzletágak közül kiemelkedik a Játék és Hálózati Szolgáltatások szegmens, amely 864,911 millió jen bevételével 93,031 millió jennel haladta meg a tavalyi adatokat. Szintén növekedett a Szórakoztató Elektronika és Szolgáltatások bevétele (600,915 millió jen, 29,132 millió jen bővülés). Ugyanakkor a Pénzügyi Szolgáltatások szegmens 232,816 millió jennel kevesebb bevételt termelt.

Összességében a Sony számos területen stabil növekedést mutat, és a diverzifikált portfóliója, valamint japán piacra helyezett fókusza erős alapokat biztosít számára. [17]

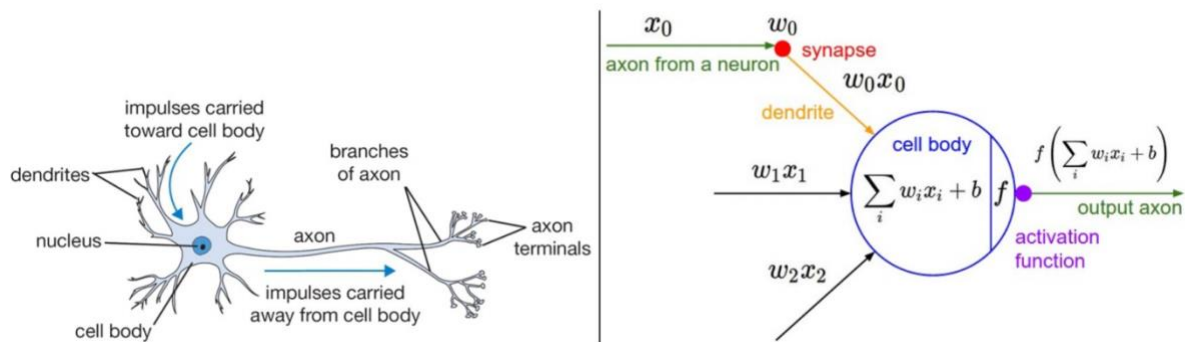
2. NEURÁLIS HÁLÓK

Ebben a fejezetben a neurális hálókat fogom bemutatni. Az egyes használt rétegek, hibafüggvények és aktivációs függvények kerülnek bemutatásra.

2.1. Neurális hálók általánosságban

A neurális hálók legegyszerűbb fajtája az emberi agy által inspirálódott gépi tanulási modell. Az emberi agy idegsejt (neurális) hálózatát reprezentálják különböző matematikai funkciók összekapcsolásával.

A neurális hálózat alapvető építőköve a neuron, ami az agy egy idegsejtjének felel meg. Minden neuron bemeneti adatokat vesz fel, azokat feldolgozza, majd egy kimenetet állít elő. A neuron a bemeneteit súlyozottan összeadja, ami egy aktivációs függvénybe kerül, aminek a kimenete a neuron kimenete lesz. A következő ábrán egy emberi idegsejt és a gépi neurális hálók neuronjának, amit csomópontnak is neveznek, a megfeleltetése látható:



1. ábra - emberi idegsejt és a neurális háló egy neuronja

A neurális hálózat felépítése rétegekre oszlik. A neurális hálózatban háromféle réteg létezik:

- **Bemeneti réteg:** Ezen a rétegen keresztül kap bemenetet a neurális háló a külvilágból. Ebben a rétegben minden egyes csomópont a bemeneti adatok egy-egy jellemzőjének felel meg.
- **Rejtett réteg(ek):** Ezek a bemeneti és kimeneti rétegek közötti rétegek. Az ezekben a rétegekben lévő csomópontok számításokat végeznek, és információt továbbítanak a bemeneti csomópontokból a kimeneti csomópontokba. A neurális hálózatnak tetszőleges számú rejtett rétege lehet, és minden rétegnek tetszőleges számú csomópontja lehet.

- **Kimeneti réteg:** Ez az utolsó réteg a neurális hálózatban. A rejtett rétegeken keresztül megtanult információkat kapja a bemenetére, majd feldolgozás után a kimenetére bocsátja.

2.2. Aktivációs függvények

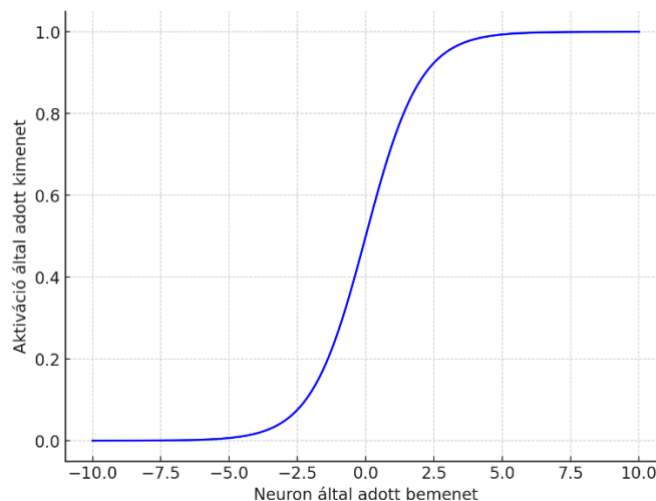
Ahogy az agy eldönti, hogy mely bemeneti jelekre, illetve jelerősségre aktiválja (vagyis tüzelje) a neuront. Az aktivációs függvények is ezt a jelenséget hivatottak szimulálni.

Az aktivációs függvények fontos szerepet játszanak a neurális hálózatokban, mert azok határozzák meg a neuronok kimenetét a bemeneti adatok alapján. Ezek a függvények lehetővé teszik a neurális háló számára, hogy nemlineáris összefüggéseket keressen.

Az aktivációs függvények célja, hogy aktiválási értéket adjanak a neuronoknak, amelyek aztán hozzájárulnak a következő réteg kimenetéhez. Az aktiválási érték megadja, hogy a neuron mely értékekre aktiválódjon, illetve milyen értéket adjon a kimenetén.

A leggyakrabban használt aktivációs függvények:

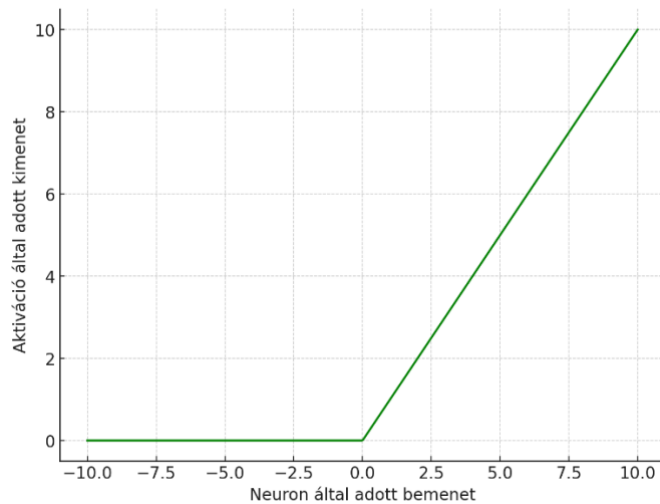
- **Sigmoid:** Ezen függvények S-alakjuk miatt kapták a sigmoid nevet. A sigmoid függvény értékei 0 és 1 közötti valószínűségeket reprezentálnak. Minél nagyobb a bemeneti érték, annál közelebb lesz a kimeneti érték az 1-hez, illetve minél kisebb a bemeneti érték, annál közelebb lesz a 0-hoz a kimeneti érték. Ezek miatt általában olyan modelleknél használják, ahol valószínűséget kell jósolni a kimeneten (osztályozási feladatnál, csak 2 osztályra ad érvényes valószínűségeket).



2. ábra - Sigmoid függvény

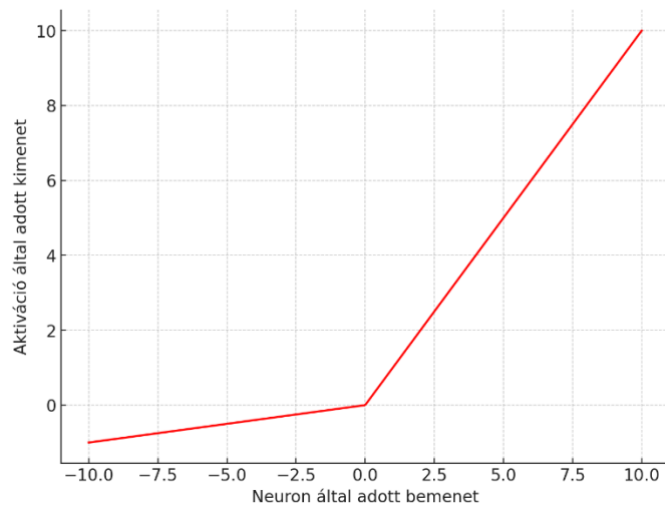
- **ReLU:** (Rectified Linear Unit) Ez a függvény a bemeneti értéket adja vissza, ha az pozitív, és 0-t, ha az negatív. A ReLU függvény gyakran alkalmazott aktivációs

függvény a mély neurális hálózatokban. A problémája, hogy a negatív oldalon a függvény gradiense 0 lesz, emiatt hibavisszaterjesztésnél (negatív bemenetre) néhány súly nem lesz frissítve (ez a dying ReLU problem).



3. ábra - ReLU függvény

- Leaky ReLU: enyhén pozitív értéket ad negatív bemenet esetén, ezért megoldja a dying ReLU problémát.



4. ábra - Leaky ReLU függvény

- Softmax: A softmax függvény, ellentétben a sigmoiddal, használható több mint 2 osztályra osztályozási problémáknál, ugyanis a kimeneti értékek összege 1. Ezt az (1) képlet írja le.

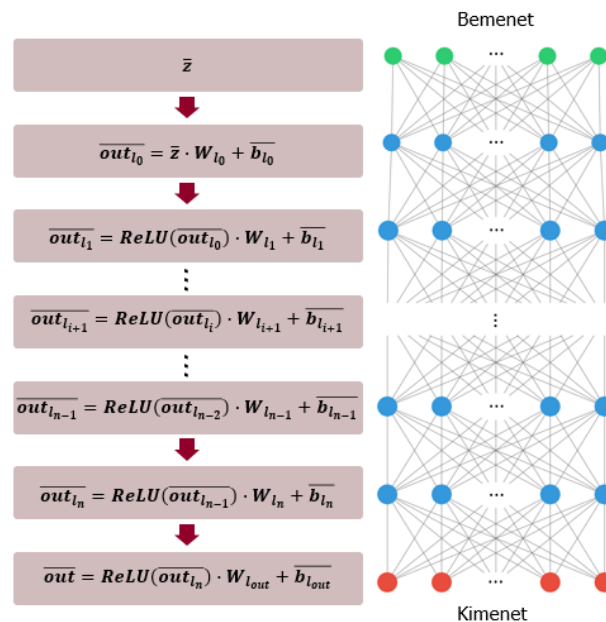
$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad (1)$$

ahol z_i az i -edik elem bemeneti értéke, e^{z_i} az i -edik elem exponenciális értéke, amely biztosítja, hogy a kimeneti értékek pozitívak legyenek. $\sum_j e^{z_j}$ normalizálja az egyes z_i -re vonatkozó kimenetet úgy, hogy azok összege 1 legyen.

Az aktivációs függvények kiválasztása a megoldandó probléma és a modellezendő adatok típusától függ. Fontos megfontolni a megfelelő függvények kiválasztását, mivel ezek befolyásolják a neurális hálózat tanulási folyamatát és a kimenetek skáláját. [18]

2.3. Dense réteg

Ez a legalapvetőbb réteg a neurális hálókban. A dense sűrűt jelent, ez a jelentés összhangban van a réteg működésével, mivel a dense rétegek egymással sűrűn vannak



5. ábra - Dense réteg felépítése [19]

kapcsolatban. A réteg a minden bemenete összeköttetésben van egy súllyal az előző réteg minden kimenetével, tehát minden bemeneti neuron kapcsolatban van minden kimeneti neuronnal.

2.4. Dropout réteg

A dropout réteg eltérő módon viselkedik a tanulási és tesztelési fázisokban. Tanulás során a dropout véletlenszerűen "kikapcsolja" a neurális hálózat egyes neuronait, így segít megelőzni az overfittinget [20]. Ezzel szemben a tesztelési fázisban a dropout réteg nem kapcsol ki neuronokat, és a tanulás során alkalmazott „dropout rate” alapján a kimeneti aktivációk skálázódnak, hogy kompenzálják a korábbi kikapcsolást, és a teljes hálózat működése stabil maradjon.

2.5. RNN réteg

Az RNN-t szekvenciális adatok modellezésére használják. Ez a réteg a memóriával rendelkező rétegek közül (RNN-nek nevezik általánosan) a legalapvetőbb, egy rejtett állapottal [21] rendelkezik, melyet mindig a bemenet és az előző rejtett állapotból számol ki. Majd azt felhasználva megalkotja a kimenetet. A réteg csak rövidtávú függőségeket képes megtalálni, illetve hajlamos a vanishing gradient és az exploding gradient problémára [22], ezért érdemesebb az RNN-ek más változatait használni.

2.6. LSTM réteg

Az LSTM réteg az RNN rétegek egy típusa, ez a réteg képes a szekvenciális adatok hosszabb távú függőségeit megtartani. Az LSTM-ek segítségével a szekvenciális adatokat, például idősoros adatokat, szöveget és beszédet lehet feldolgozni és elemezni. Ezáltal kiválóan alkalmas árfolyambecslésre. Azonban szem előtt kell tartani, hogy ez a réteg is korlátos mennyiségű szekvenciális adat függőségeit képes megjegyezni, tehát nem érdemes egyszerre túl sok adatot becsülni. A réteg egy memóriacellát és 3 kaput tartalmaz. Ezeket az adatokat az áramlásának vezérlésére használja, ezáltal lehetővé téve, hogy szükség szerint szelektíven tartsa meg vagy dobja el az információt. Ennek segítségével a réteg el tudja kerülni a hagyományos RNN-eket sújtó vanishing gradient problémát.

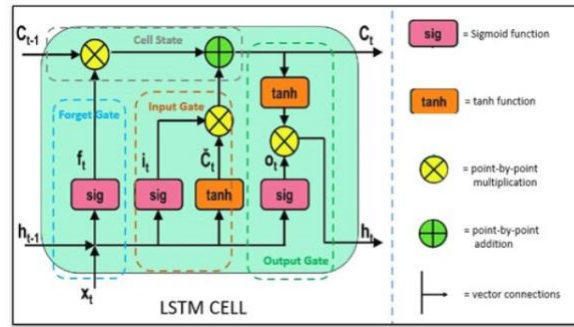
Az LSTM-eket széles körben használják különböző alkalmazásokban, például a nyelvi feldolgozásban, a beszédfelismerésben és az idősor-előrejelzésben.

2.6.1. Kapuk típusai

Az LSTM-ben háromféle kapu létezik: input gate, forget gate, output gate. Az input gate szabályozza az információáramlást a memóriacellába. A forget gate szabályozza a memóriacellából kifelé történő információáramlást. Az output gate eldönti, hogy mely információt küldje a kimenetre.

A kapuk mindegyike szigmoid függvényekkel van megvalósítva. A függvény egy 0 és 1 közötti számot ad vissza, és az érték szerint viselkedik a kapu.

Az 5. ábra vizuálisan reprezentálja az LSTM felépítését.



6. ábra - Az LSTM struktúrája

A **forget kapu**: kiszámolja, hogy az előző cella rejtett állapota (h_{t-1} az 5. ábrán) és az új bemeneti adatok (x_t az 5. ábrán) alapján mi az, amit el kell felejtene a memóriacellának.

Az **input kapu**: eldönti, hogy mely információ menjen be a memóriacellába. Az input kapu bemenetként megkapja az előző cella rejtett státuszát és az új bemeneti adatot, ezek közül eldönti mennyit tartson meg a szigmoid függvény értékének segítségével. A szigmoid függvény 0 értékére semmit se tart meg, 1 értékre mindent megtart, a kettő között pedig csak egy adott részét tartja meg. Tehát az input kapu frissíti a memóriacella állapotát.

Az **output kapu**: megkapja az előző cella rejtett státuszát és a bemenetet, majd egy szigmoid függvénnyel eldönti, hogy mi a releváns információ. Ezt a megmaradt információt és a már frissített memóriacella egy tanh függvényen áteresztett értékeit összeszorozza és ezzel kész a cella outputja. [23]

2.7. GRU réteg

A GRU a Gated Recurrent Unit rövidítése. Ez a réteg a recurrent neural network (RNN) egy típusa, ahogy az LSTM is, amihez hasonlóan a szekvenciális adatok modellezésére alkalmas. A GRU egy egyszerűbb architektúrával rendelkezik, így jóval gyorsabb tanítani.

Ahogy az 3.5. pontban említettem az LSTM egy memóriacellával rendelkezik, ezzel szemben a GRU réteg egy candidate action vektorral rendelkezik, amit egy reset kapu és egy update kapu kezel.

A réteg minden lépésben kiszámolja a candidate action vektort az előző rejtett állapot és az aktuális input alapján. Majd ennek a vektornak a segítségével frissíti a rejtett állapotot.

A reset kapu kiszámítása a bemenet és az előző rejtett állapot segítségével történik:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \quad (2)$$

ahol W_r és U_r mátrixok, melyek rendre a bemenetet és az előző rejtett állapotot súlyozzák. Ezen mátrixokat modell a training során tanulja meg. Az előző rejtett állapotot h_{t-1} és a bemenetet x_t jelöli.

Az update kaput a reset kapuhoz hasonlóan számolja ki a réteg más mátrixok használatával:

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \quad (3)$$

A candidate action vektort a bemenet és az előző rejtett állapot reset kapuval vett szorzata alapján számolja ki a következőképpen:

$$\tilde{h}_t^j = \tanh(W x_t + U(r_t \odot h_{t-1}))^j, \quad (4)$$

ahol W és U mátrixok, melyeket a training során tanul meg a modell.

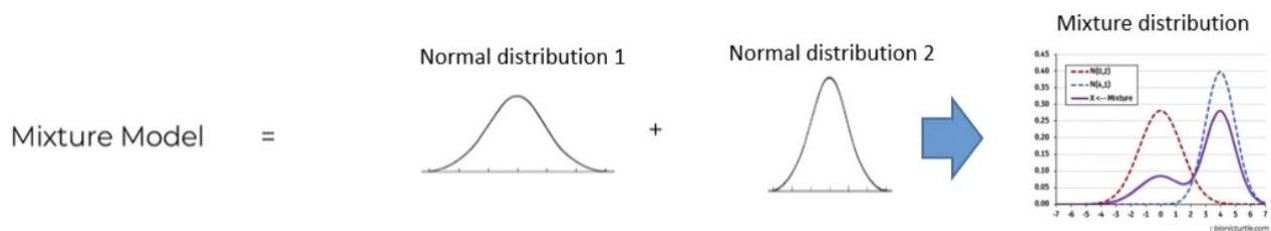
A réteg aktivációját az előző rejtett állapot és a candidate activation vektor interpolációjával kapjuk meg:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j, \quad (5)$$

Az LSTM-nél egyszerűbb architektúrának köszönhetően, a GRU kevesebb paraméterrel rendelkezik, ezáltal a modellnek kevesebb paramétert kell frissítenie, ami a gyorsabb tanulást eredményez. Ez a réteg is képes a szekvenciális adatok hosszútávú függőségeit felismerni és megjegyezni, habár pont az egyszerűbb struktúra végett rosszabbul teljesít hosszabb szekvenciák és nagyobb adatmennyiség esetén az LSTM-nél [24]. Az LSTM hajlamosabb overfittingre kisebb adathalmazokon [25].

2.8. MDN réteg

Az MDN, vagyis Mixture Density Network egyesíti a neurális hálók és a mixture models tulajdonságait. A segítségével a regressziós problémák megoldására lehet modellt létrehozni. Az MDN egy feltételes valószínűségi eloszlást prediktál, ami a Mixture Modell eloszlása. A Mixture Modell akár több Gauss eloszlásból is állhat. Az MDN ezen Gauss eloszlások paramétereit jósolja meg.



7. ábra - Mixture eloszlás

Minden Gauss eloszláshoz tartozik egy átlag (μ), egy szórásnégyzet/variancia (σ^2) és végül

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

egy mixing coefficient (π), ezen lesznek a modell kimenetei. A mixing coefficient az egyes eloszlásokhoz egy súlyt/valószínűséget ad, ami reprezentálja, hogy az egyes Gauss eloszlások milyen arányban vesznek részt a teljes eloszlásban, tehát a teljes eloszlás az egyes Gauss eloszlások súlyozott összege. Mivel a π -k valószínűségeket reprezentálnak, ezért kisebbek kell legyenek 1-nél és az összegük 1-et kell kiadjon. Ezen tulajdonságokat egy másik megközelítés szerint is be lehet látni, miszerint a Gauss görbe alatti területnek 1-nek kell lennie:

$$\int_{-\infty}^{\infty} f(x)dx = 1 \quad (6)$$

Az MDN valószínűségi eloszlására is igaz a fenti feltétel, ebből következik, hogy ha az egyes Gauss görbékhez nem veszünk súlyokat, akkor a területük összege nagyobb lesz 1-nél, ami értelemszerűen hibás lenne.

Ahogy azt említettem az MDN feltételes valószínűségi eloszlások modellezését teszi lehetővé:

$$p(t|x) = \sum_{k=1}^K \pi_k(x) \mathcal{N}(t|\mu_k(x), \sigma_k^2) \quad (7)$$

ahol a t eloszlás paraméterei meghatározhatók a neurális hálózat kimeneteiből, feltéve, hogy a hálózat bemenete x . A K az eloszlások számát jelenti. Összesen K darab π lesz a kimeneten, minden eloszláshoz egy. Az átlagból (μ) és varianciából (σ^2) viszont $K \times L$ darab lesz, ahol az L a dimenzió nagysága.

A keverési együtthatókat softmax kimenetek halmazaként számítjuk ki, ahol a_k^π határozza meg a hálózat által kibocsátott keverési együtthatókat. Az a a hálózat egy neuronjának a kimenete:

$$\pi_k(x) = \frac{\exp(a_k^\pi)}{\sum_{k=1}^K \exp(a_k^\pi)} \quad (8)$$

A variancia a neurális hálózat neuronja általi kimenet exponenciálisa:

$$\sigma_k(x) = \exp(a_k^\sigma) \quad (9)$$

A $K \times L$ darab átlag kiszámításához a neurális hálózat kimenetét használjuk bármilyen módosítás nélkül:

$$\mu_k(x) = a_{kj}^\sigma, \quad (10)$$

ahol a_{kj}^σ : a hálózat kimeneti súlya, amely az j -edik neuronhoz tartozik, és a k -edik komponenshez kapcsolódik.

A súlyokat és bias-okat a likelihood negatív logaritmusának minimalizálásával tesszük meg, amit negatív log-likelihood-nak neveznek, ez lesz a modell hibafüggvénye:

$$E(w) = - \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k(x_n, w) \mathcal{N}(t_n | \mu_k(x_n, w), \sigma_k^2(x_n, w)) \right\} \quad (11)$$

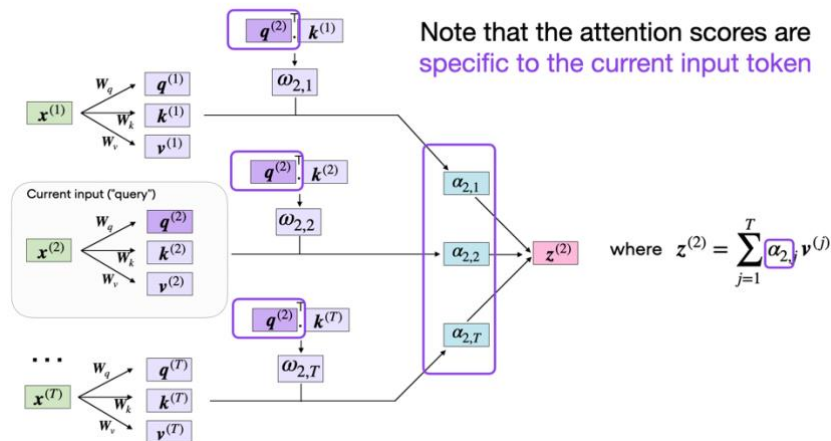
ahol N az adatok száma (például a tanulási adatokban szereplő minták száma), $\mathcal{N}(t_n | \mu_k(x_n, w), \sigma_k^2(x_n, w))$ a k -edik eloszláshoz számított súlyfüggvény az elvárt kimenetet (t_n) figyelembe véve, $\pi_k(x_n, w)$ a mixing coefficient, amely a k -edik eloszlás súlya.

2.9. Self-attention réteg

A self-attention rétegnek több típusa van, én a legáltalánosabb típusát használtam. A réteg lényege, hogy a beadott szekvenciából az adatokat súlyozottan használja fel, aszerint, hogy mely bemenetet ítéli meg a réteg relevánsnak és kevésbé relevánsnak.

A réteg 3 vektort alkalmaz, ezek a query, key és value vektorok. A szekvencia minden adata tartalmazza a 3 vektort. A self-attention először veszi a szekvencia első adatának query vektorát és a szekvencia minden egyes adatának key vektorával veszi a skaláris szorzatát. Ezzel a réteg kiszámolta, hogy a szekvencia adatai mennyire fontosak az első adathoz képest. A kiszámolt skaláris szorzatokat normalizálja a key vektorral, majd alkalmazza a softmax függvényt, aminek segítségével elkerülhető a vanishing gradient problem és az exploding gradient problem is. A key vektorral azért kell normalizálni, mert ha nem tennénk meg, akkor a softmax függvény használata után az egyes bemenetekhez túl kicsi súly tartozna. Ezt az eredmény szokás attention score-nak nevezni. Következő lépésként a réteg egy súlyozott összeget számol, ahol az előző lépésben kapott attention-score-ok lesznek a súlyok, a komponensek pedig az eddig még nem használt value vektorok. Ezzel előállt az első bemenethez a hidden state.

Ezt követően ugyanezen lépéseket végrehajtja a réteg iteratíván a szekvencia következő adataira, tehát a szekvencia minden bemenetéhez kapunk egy hidden state-et.



8. ábra - a self-attention mechanizmusa [26]

A Multi-head attention a self-attention egy típusa, aminek a neve is sugallja, hogy több head-et használunk. A head a query, key és value vektorok kiszámolásához tartozó súly mátrixok (mindegyikhez egy tartozik), a súly mátrixokat a modell a training során tanulja. Ennek a rétegnek az alkalmazásával több head-et tudunk inicializálni, praktikusán ez azt jelenti, hogy mindegyik head-hez kiszámolja a fent leírt folyamatot. Ennek az előnye, hogy mindegyik

head a bemeneti szekvencia más-más részeire tud fókuszálni, tehát több függőséget tud találni a szekvencián belül.

Egy hátránya a self-attention rétegeknek, hogy nem veszi figyelembe a szekvencia bemeneteinek a pozícióit, tehát ha például a szekvenciánk 3 gyertya, ahol sorban a gyertyák záró árai rendre 110, 111, 112, akkor a réteg a következő szekvenciával egyenértékűnek fogja tekinteni: 112, 110, 111, ami természetesen nem kedvező, hiszen a hasonló szekvenciák között különbséget kell tenni. Ezt a problémát ki lehet küszöbölni egy positional encoding réteg alkalmazásával, ami a hullámfüggvények alkalmazásával, gondoskodik arról, hogy minden pozíciót egyedivé tegyen, így a modell meg tudja tanulni, hogy mely pozícióhoz tartozik a szekvencia értéke. A réteg kétféleképpen is implementálható, az első változatban a hullámfüggvények kimeneteit a szekvencia értékeihez adjuk. A második változatban a hullámfüggvény kimeneteit összefűzzük a szekvencia értékeivel. Mindkét megoldást implementáltam. Tapasztalataim szerint mindkettő megfelelően teljesít, a 2. verziónak a hátránya, hogy megnöveli a dimenziót, így további számítási terheltséget vezet be, így az első verziót alkalmaztam.

2.10. Ensemble Modellek

A neurális hálók legegyszerűbb megalkotása a soros rétegek alkalmazásával lehetséges, ami a rétegek egymásutáni elhelyezkedését jelenti. Emellett lehetőség van a rétegek párhuzamos elhelyezésére. Ennek segítségével többfajta réteget tudunk elhelyezni párhuzamosan, így mindegyik ág a bemeneti szekvencia más és más függőségeit tudja rögzíteni. Egy remek példa erre, CNN és multi-head attention rétegek alkalmazása a két párhuzamos ágon. A CNN ág képes lokális (rövidtávú) függőségek felismerésére, ráadásul kicsi bias értékkel rendelkeznek, ezért kevésbé hajlamos overfitting-re és könnyen tanítható. Ezzel szemben a multi-head attention képes a hosszabbtávú függőségeket lokalizálni, így a két ág használatával a rövid és hosszú távú mintákat képes a modell megtalálni. Már csak a két ágat kell egyesítenünk, amit megtehetünk többféleképpen is. Használhatunk konkatenációt, vehetjük a két ágból a minimumot, maximumot vagy az átlagot, de vannak egyéb lehetőségek is. Amikor különböző architektúrájú rétegeket használunk párhuzamos kapcsolatban, akkor ensemble modellekről beszélünk.

3. A NEURÁLIS HÁLÓ ADATAINAK KEZELÉSE

Ezen fejezetben a neurális hálózat bemenetének adatairól, és azok feldolgozásáról fogok írni, illetve a háló kimenetéről.

Alapvetően árfolyam adatok letöltésével a kiválasztott intervallumokra kapjuk meg a gyertyákhoz tartozó adatokat. Ezeket az adatokat röviden OHLCV-nek nevezzük.

Az OHLC (open, high, low, close) adatok egy részvény adott időintervallumára vonatkozó nyitó-, legmagasabb, legalacsonyabb és záróárakat mutatják. Ezek az adatok alapvető fontosságúak a részvény ármozgásának elemzése szempontjából, hiszen egy teljes képet adnak az adott időszak árfolyamváltozásáról. A legtöbb elemzés napi, órás vagy 5 perces időintervallumok alapján készül. A tanulmányomban én is napos adatokat használtam. [27]

Egy másik gyakori bemenet a **volume** (volumen). A volumen a kereskedett részvények vagy ügyletek mennyiségét méri. A volumen és az OHLC adatok változásának különböző jelentéseket tulajdonítanak, ezeket az adatokat röviden OHLCV-nek nevezzük. A volumen jelezheti a piac erejét, mivel a növekvő volumenű, erősödő piacokat általában erősnek tekintik. Az OHLCV adatok kombinált értelmezése, ahogy az ábrán is látható, további betekintést nyújt a piaci hangulatba és erejébe.

Identify Market Interpretation Using Price Volume & Open Interest						
Price		Volume		Open Interest		Interpretation
	Rising		Rising		Rising	A. Strong B. Bullish C. Price Continue to Rise
	Rising		Falling		Falling	A. Weakening B. Moderately Bearish C. Current Uptrend is Weakening
	Falling		Rising		Rising	A. Weak B. Bearish C. Downtrend is Continues
	Falling		Falling		Falling	A. Strengthening B. Moderately Bullish C. Current Downtrend Weakening
www.shubhlaxmicommodity.com						

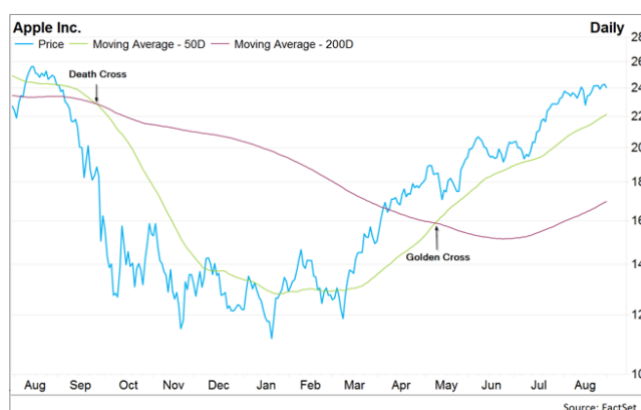
9. ábra - Volumen, OHLCV események értelmezése [28]

3.1. Technikai indikátorok

Az OHLCV adatokból származó technikai indikátorok segítenek mélyebb összefüggések felismerésében a piaci viselkedésben. Az árfolyam elemzők ezeket az indikátorokat különböző mintákhoz és jelekhez kötik, amelyek alapján következtetnek a jövőbeli ármozgásokra. Az elemzések során az indikátorok közötti kapcsolatokat is értelmezik, hogy jobban megértsék a piac dinamikáját. Annak érdekében, hogy a modell jobb előrejelzéseket készítsen, az OHLCV

adatok alapján számolt technikai indikátorokat is felhasználtam, bízva abban, hogy a modell azonosítja azokat a mintákat, amelyek megbízható előrejelzést nyújtanak.

Az **SMA** (Simple Moving Average), ahogy a neve is sugallja, egy egyszerű átlagot jelent, kiszámolja az átlagot a választott intervallum alatt. Általában a záró árnak szokták az átlagát venni. A leggyakoribb periódusok az átlagok számítására a 10, 20, 50 és a 200. Az egyes periódusú SMA-k közötti változásra több minta (pattern) is létezik, amelyek a részvény előrejelzésére szolgálnak. Ilyen pattern a „death cross”, ez akkor jelenik meg a grafikonon, ha a részvény rövid távú mozgóátlaga a részvény hosszú távú mozgóátlaga alá esik. „Golden cross”-nak azt nevezzük, ha a rövid távú mozgóátlag a hosszútávú mozgóátlag fölé kerül, és a lefelé irányuló trend kimerülését jelezheti előre. A rövid távú mozgóátlagnak szokás venni a 10 napos periódust, hosszúnak pedig a 20 naposat, én is ezt a párosítást választottam.



10. ábra - Golden cross, Death cross

Egy másik típusa az átlag alapú indikátoroknak az exponenciális mozgó átlag, vagyis az **EMA** (Exponential Moving Average), amely egy olyan típusú mozgóátlag (MA), amely nagyobb súlyt és jelentőséget tulajdonít a legfrissebb adatoknak. Az EMA nagyobb mértékben reagál a közelmúlt árváltozásaira, mint az SMA, mivel az SMA minden adathoz azonos súlyt ($1/N$ -et, ahol N a periódus) használ. Többféleképpen is felhasználható. Alkalmazható csak magával a részvennyel együtt a következőképpen: ha az EMA görbéje növekszik, és a részvény a görbe alatt marad, akkor ezt egy bullish jelnek szokták értelmezni, és fordítva, ha az EMA csökken és a részvény a görbe fölé tör, akkor az egy bearish jelet jelent. Egy másik felhasználása hasonlóan történik, mint ahogy azt az SMA-nál láttuk. Egy crossover mintát alkalmaz: ha a 10 napos EMA a 20 napos fölé kerül, akkor az a részvény erősödését jelentheti, ha viszont a 10 napos EMA a 20 napos alá esik, az a részvény gyengülését jelentheti.

A Moving Average Convergence/Divergence (**MACD**) egy technikai indikátor, amely segít az árfolyamok trendjeinek azonosításában. Ezen indikátor a momentum indikátorok közé tartozik, mivel az árfolyam erősségét adja meg. Az MACD egy részvény árfolyamának két exponenciális mozgóátlaga (EMA) közötti kapcsolatot mutatja. A MACD-vonalat úgy számítják ki, hogy a 26 periódusú EMA-t kivonják a 12 periódusú EMA-ból, amely létrehozza a MACD vonalat. Ezenkívül van egy kilenc napos EMA, amit a MACD-vonalból számítanak, ezt „signal line”-nak nevezik, mert az generálja a jelet, a részvény vételére vagy eladására. A 9 napos EMA-t az MACD vonallal együtt rajzoljuk ki.



11. ábra - MACD indikátor

A relative strength index (**RSI**) az MACD-hez hasonlóan egy momentum indikátor. Az RSI egy értékpapír közelmúltbeli árváltozásának sebességét és nagyságát méri, azzal, hogy megállapítja az árfolyamról, hogy az túlvett (overbought) vagy túladdott-e (oversold). Az RSI értékei egy nullától 100-ig terjedő skálán jelennek meg. Jelezheti, hogy mikor érdemes vásárolni és eladni. Hagyományosan a 70-es vagy annál magasabb RSI érték túlértékelt állapotot jelez, tehát a részvényt „túlvették”. A 30-as vagy az alatti érték alulértékelt állapotot jelez.

A Commodity Channel Index (**CCI**) az MACD és az RSI-hez hasonlóan egy momentum indikátor. Jelzi, ha egy részvény overbought vagy oversold. A Commodity Channel Index az aktuális ár és a történelmi átlagár közötti különbséget méri. Ha a CCI pozitív, az azt jelzi, hogy az ár a történelmi átlag felett van. Ezzel szemben, ha a CCI negatív, az ár a történelmi átlag alatt van. A CCI korlátlan, ami azt jelenti, hogy a végtelenségig emelkedhet vagy csökkenhet. Emiatt a túlvett és túladdott szinteket jellemzően úgy határozzák meg az egyes eszközökre vonatkozóan, hogy megnézik a történelmi szélsőértékeit CCI-nek, ahonnan a részvény trendje épp irányt váltott.

A logaritmikus hozam, vagy röviden **log return**, a részvény relatív változását mutatja meg, ellenben a „normális” return-től (két close price különbsége), ami egy abszolút változást mér. A log return-t a záró árfolyamra szokták kiszámolni a következőképpen:

$$z_t = \log\left(\frac{p_t}{p_{t-1}}\right), \quad (12)$$

ahol p_t az aktuális záró árfolyam, és p_{t-1} az azt megelőző záró árfolyam. A log return használatának másik előnye, hogy normális eloszlású, ami segíti a neurális hálók taníthatóságát. Ennek oka, hogy egy szám természetes logaritmus a folytonosan differenciálható függvény, tehát a log return eloszlása is folytonosan differenciálható. [29]

A **Bollinger Bands** (Bollinger „szalagok/sávok”) segít felmérni a részvények és más értékpapírok volatilitását aszerint, hogy azok túl- vagy alulértékeltek-e. Az eszközt John Bollinger pénzügyi elemző fejlesztette ki, amely 3 görbét tartalmaz, melyek a részvénnel együtt mozognak. A középső görbe a részvényárfolyam 20 napos mozgóátlaga (SMA). A felső és az alsó sávok a középső vonaltól annak varianciájával számított görbék. Amikor egy részvény volatilisabbá válik, akkor a sávok kiszélesednek, amikor kevésbé az, akkor összehúzódnak. Sok kereskedő úgy gondolja, hogy ha a részvény a felső sávot kezdi megközelíteni, akkor a részvény túlértékelté válik, ha pedig az alsó sávot, akkor alulértékelté válik, és erre építenek kereskedési stratégiát.

A Weighted Moving Average (**WMA**) hasonló az SMA-hoz azzal a kivétellel, hogy a WMA nagyobb hangsúlyt fektet az aktuális gyertyához időben közelebbi adatokra, mint a távolabbiakra. Ezt úgy teszi meg, hogy a legfrissebb adathoz veszi a legnagyobb súlyt, majd egyenletesen egyre kisebbeket vesz a távolabbi adatokhoz. A súlyok alkalmazásával számolja ki a mozgó átlagot.

3.2. Adatok előfeldolgozása

A bementi adatok értéke túlnyomó részt nem korlátos, tehát elméletben bármekkora értéket fel tudnak venni. Ez azért baj, mivel, ha a modellt ilyen értékekkel tanítjuk fel, akkor az az adatok értékeivel csak egy bizonyos intervallumban találkozhat, és ha tesztelésére kerül sor, előfordulhat, hogy olyan bemeneti adatok kerülnek a modellbe, amik az intervallumon kívül esnek, tehát az előrejelzés pontatlan lehet. Ennek kiküszöbölésére az adatokat nem „nyersen” adtam be, hanem azok értékeinek változását alkalmaztam. A képlet általánosan:

$$a_{diff} = a_i - a_{i-1} \quad (13)$$

A bemenetre kizárólag a a_{diff} -et adtam be, a nyers adatot nem. Elméletben ez a változás sem lesz korlátos, hiszen az adat bármekkora lehet, de a gyakorlatban a modell 2 egymást követő

napi részvény közötti különbség értékével nagy valószínűséggel találkozott a múltban, tehát teszteléskor nem lesznek nem várt adatok, vagy csak elhanyagolható mértékben.

Az átlag alapú indikátorokra érdemes különbséget számolni, hiszen az átlagokat egy bizonyos intervallumon számoljuk, és például, ha egy részvény erős növekedési tendenciát mutat, akkor mindig újabb és újabb értékekkel találkozik majd a modell. Ilyen indikátorok a következők: SMA, EMA, WMA, MACD, Bollinger Bands.

Az RSI indikátor egy 0 és 100 közötti értékkel rendelkezik, tehát erre felesleges lenne különbséget számolni, mert az csak rontana az indikátor értelmezhetőségen.

Noha a CCI nem korlátos, de egy adott részvényre egy idő után „beáll”, tehát általánosságban nem fog a múltbéli értékek intervallumán kívül esni, ezért erre az indikátorra sem számoltam különbséget.

A log return-re sem szükséges különbséget számolni, hiszen az eleve egy relatív változást ad meg. Hasonlóan további indikátorok alkalmazása esetén is végig lehetne gondolni, hogy érdemes-e a különbségeket venni a nyers adat helyett.

Továbbá megjegyzendő még, hogy indikátorok kiszámolásakor a bemeneti adatok első néhány sorában NaN értéket kapunk, tehát a modellnek egy nem releváns értéket, ezen sorokat ki kell törölni az adatok közül. Ezáltal adatokat veszünk, ilyen értékvesztés lehet az átlag kiszámolásából adódóan. Például, ha 20 napos SMA-t számolunk, akkor az első 19 sorra nincs értéke, csak a 20. sortól lesz. Egy 150 napos SMA-t már jobban meg kell gondolni, hogy megéri-e alkalmazni, mert ott már jelentősebb az adatvesztés.

3.3. Target data kiválasztása

A target data, az az adat, amit meg szeretnénk jósolni. Az MDN csak regressziós problémákra alkalmazható, tehát ennek megfelelően kell megválasztani a kimeneti adatot. Ez lehet az aktuális napot követő nap záró ára és az aktuális nap záróárának különbsége vagy hányadosa. A logaritmikus hozam (log return) használata előnyös, mivel normális eloszlást követ, ami jól illeszkedik az MDN által használt Gauss-eloszláshoz.

Hosszabb távú előrejelzések esetén a log return későbbi időpontokra is számolható, amivel a stratégia is hosszabb távú pozíciókra optimalizálható. Természetesen ebben az esetben a kereskedési stratégiát is változtatni kell, mivel a távolabbi árfolyamok előrejelzésekor a nyitott pozíciót is tovább kell nyitva tartani. A konkrét kereskedési stratégiákat részletesen a 6. fejezetben ismertetem.

3.4. Train, teszt, validation adathalmazok felosztása

Az adatokat érdemes három részre osztani: train, validation és teszt részekre. Én 70:20:10 arányban osztottam fel az említett sorrendben. A validation-nal adathalmaz azért fontos, hogy tudjunk early stopping-ot használni: a train adathalmazon tanítjuk a modellünket közben a validation adatokon teszteljük minden epoch után, ha a modellünk egyre rosszabb eredményt ér el a teszteken, akkor leállítjuk a tanítást, hogy elkerüljük az overfitting-et.

A teszt adathalmazon teszteljük a már feltanított modellünket. Ezen a teszteket sokféleképpen el lehet végezni, lehet hagyományosan a modellel loss-ra, accuracy-ra lefuttatni, de a részvényeknél hatásosabb, ha szimuláljuk a modell teljesítményét részvénykereskedésen demo környezetben, amivel a kereskedési stratégiánk hatékonyságát is teszteljük.

Ezen adathalmazok létrehozása fontos, hiszen ha nem tennénk meg, és modellünket tanítását és tesztelését ugyanazon adatokon végeznénk, akkor az a jövőbe látna, és hamis képet kapnánk a modell teljesítményéről.

Szokás még más arányokban felosztani az adatokat. Alapvetően kutatási környezetben a training halmazt szokás nagyobbnak venni, ami 70:20:10 vagy 70:15:15 arány általában. Az ipari/szolgáltatás területen, a teszt halmazok aránya teszi ki a nagyobb részt, mivel ott elengedhetetlen a modelleket teljes körűen tesztelése, mielőtt azokat a gyakorlatban felhasználják. [30]

3.5. Adatok standardizálása

Az bemeneti adatok standardizálása fontos szerepet játszik annak biztosításában, hogy a modell hatékonyan tudjon tanulni, és pontos előrejelzést tudjon végezni. Az adatok standardizálása úgy alakítja át a bemeneti jellemzőket, hogy azok átlaga 0 és szórása 1 legyen. A részvénypiaci adatokon, ahol a jellemző tartományok jelentősen eltérhetnek (pl. a részvényárak több száz vagy ezeres nagyságúak lehetnek), a nyers, nem szabványosított adatokon történő modeltanítás számos problémát okozhat. Ahogy említettem a 3.2 szekcióban, nem használtam nyers adatokat a modell tanítására, ennek ellenére érdemes standardizálni, mivel a standardizálás számos előnnyel jár.

A standardizálás segít elkerülni a vanishing és exploding gradient problémáját azáltal, hogy a bemeneti értékeket hasonló tartományon belül tartja, ami megkönnyíti a hálózat konvergálását a tanulás során.

A megfelelően standardizált adatok lehetővé teszik az optimalizáló számára, hogy hatékonyabban mozogjon a paraméterterben, ezáltal a modell hatékonyabban tud konvergálni az optimális megoldás felé.

A standardizálás folyamatának első lépése, hogy az adott feature-ből kivonjuk a feature-re számított átlagot, így kapott adatok 0 középké lesznek. Következő lépésben elosztjuk az eredményt a jellemző szórásával, ezáltal az 1 szórású lesz. Ezt a folyamatot minden jellemzőre megteszük.

Az adatok standardizálása során kiemelten fontos, hogy a jellemzők átlagát és szórását kizárólag a training adathalmazra számoljuk ki. Amennyiben ugyanezeket a statisztikai mutatókat a teszt vagy validációs adathalmazra is kiszámítanánk, majd az így nyert értékekkel standardizálnánk, az a modell számára olyan előnyt jelentene, mintha a jövőbeli adatokba „belelátna”. Ez a modell teljesítményét nagymértékben megnövelné a teszteken, és torzított, félrevezető képet adna az eredményekről. Ennek elkerülése érdekében a teszt és validációs adatok standardizálása során a training adathalmazon kiszámított átlagot és szórást kell használni, hogy a modell valós prediktív képességét értékelhessük.

3.6. Adat szekvenciák

Annak érdekében, hogy ne csak az aktuális nap adatait kapja meg a modell, szekvenciákat kell létrehozni a már feldolgozott, adathalmazokra felosztott és standardizált adatokból. Az így kapott szekvenciák kerülnek a modell bemenetére. A szekvenciák nagyságát körültekintően érdemes megválasztani, ha túl kicsi, akkor nem biztosítunk lehetőséget a hosszú távú függőségek megtalálására. Amennyiben túl nagy előfordulhat, hogy a modell nem találja meg a rövid távú függőségeket, és a tanulási folyamat is bonyolultabb, ezáltal lassabb lesz. A használt réteget is figyelembe kell venni, például az LSTM alkalmazása esetén, ahogy azt említettem a 2.6-os szekcióban, nem képes korlátlanul nagy szekvenciák kezelésére, ezért nem érdemes kiugróan hosszú szekvenciákat alkalmazni. A szekvencia hosszának összhangban kell lennie az adatok típusával, például a részvénytőke adatok esetében fontos, hogy a szekvencia lefedje az árfolyamok változásának lényeges szakaszait, például egy piaci trendet vagy ciklust.

A szekvenciák hosszát dinamikusan választottam meg a tanítás során a modellek prediktív képességének figyelembevételével, a részletesebb tanítási eljárást a 7. fejezetben ismertetem.

4. MDN RÉTEG OUTPUTJÁNAK FELDOLGOZÁSA

Az MDN (Mixture Density Network) működését és alkalmazását korábban a 2.8-as szekcióban részletesen bemutatam. A kutatásomban minden általam használt modell ezzel a kimeneti réteggel rendelkezett, amelynek a feladata, hogy az adott valószínűségi eloszlások paramétereit előállítsa. Azonban a modell által szolgáltatott nyers kimeneteknek (átlagok, szórások és mixing coefficients) még további feldolgozása szükséges ahhoz, hogy használható kereskedési stratégia születhessen. A következő szekciókban bemutatom az általam alkalmazott különféle feldolgozási módszereket.

4.1. Kumulált CDF függvény becslés

Ebben a megoldásban a réteg által megalkotott eloszlások Cumulative Distribution Function (CDF) függvényeit vesszük súlyozottan. A használt képlet:

$$\begin{aligned}\tilde{p}_t &:= \mathbb{P}[Y_t > 0] = \int_0^{\infty} p_t(y; \lambda, \mu, \sigma) = 1 - \mathbb{P}[Y_t \leq 0] \\ &= 1 - \int_{-\infty}^0 p_t(y; \lambda, \mu, \sigma),\end{aligned}\tag{14}$$

ahol Y_t a log return random változója t időpillanatban, a teljes eloszlásra vett sűrűség függvényt pedig $p_t(y; \lambda, \mu, \sigma)$ adja meg. Az egyenlet egyszerűsíthető a jobb oldalon látható formára.

A $\int_{-\infty}^0 p_t(y; \lambda, \mu, \sigma)$ részt kibonthatjuk az egyes eloszlásokra.

$$1 - \mathbb{P}[Y_t \leq 0] = 1 - \sum_{i=1}^K \lambda_i \phi_i(0),\tag{15}$$

ahol K a Gauss eloszlások száma, $\phi_i(0)$ az i -edik eloszláshoz tartozó CDF függvény, λ_i pedig az egyes eloszlásokhoz tartozó mixing coefficient.

A képlet szerinti megoldást kumulált CDF-nek nevezem, mivel a $\mathbb{P}[Y_t \leq 0]$ valószínűséget CDF-nek hívják, és a CDF-et kumuláltan súlyozva vettem az egyes eloszlásokra. Mellesleg ez a név lehetővé teszi erre a feldolgozási módszerre való egyszerű hivatkozást.

A képlet eredménye megadja, hogy mekkora valószínűséggel lesz a logaritmikus hozam pozitív. Tehát, ha a log return-t az aktuális nap és az azt követő nap közötti záró árra vesszük, akkor azt adja meg, hogy mekkora valószínűséggel lesz nagyobb a későbbi záró ár, mint a korábbi. Az árfolyamra becsült CDF a 12-es ábrán látható. [31]



12. ábra - CDF becslés árfolyammal együtt megjelenítve

4.2. Súlyozott várható érték becslés

A Gauss eloszlások paramétereinek felhasználására más lehetőség is van. Egyszerűen vehetjük a becsült várható értékek súlyozott összegét a következő módon:

$$\mu_{weighted} = \sum_{i=1}^K \lambda_i \mu_i \quad (16)$$

A képlet megadja, hogy minden eloszlást figyelembe véve mi lesz a target data várható értéke. Fontos kiemelni, hogy ez a megoldás nem használja fel a varianciát, ezáltal az előrejelzés nem tartalmaz kockázat becslést. Emiatt érdemes egy olyan megoldással együtt használni, amely méri a modell bizonytalanságát.

4.3. Súlyozott variancia becslés

A súlyozott varianciát a következő képlet segítségével számítottam ki:

$$\sigma_{weighted}^2 = \sum_{i=1}^K \lambda_i (\sigma_i^2 + (\mu_i - \mu_{weighted})^2) \quad (17)$$

A fenti képlet alapján súlyozottan vesszük figyelembe az egyes eloszlások szórásnégyzetét (σ_i^2), illetve hozzáadjuk az egyes eloszlások súlyozott várható értéktől ($\mu_{weighted}$) való eltéréseinek négyzetét. A λ_i az egyes komponensekhez tartozó mixing coefficient, amely meghatározza, hogy az adott komponens mekkora súllyal szerepel a teljes eloszlásban.

Végül, hogy varianciát kapjunk a $\sigma_{weighted}^2$ gyökét vesszük:

$$\sigma_{weighted} = \sqrt{\sigma_{weighted}^2} \quad (18)$$

A súlyozott variancia segítségével mérhetjük, hogy az egyes becslések milyen mértékben biztosak, és mekkora kockázatot hordoznak magukban. Ez a megközelítés különösen hasznos, mivel a modell által generált bizonytalansági becsléseket is figyelembe tudjuk venni a kereskedési döntések során. Így lehetőség van arra, hogy a kereskedési stratégia ne csupán a várható értékeken alapuljon, hanem a várható volatilitásra is építsen. A súlyozott várható érték és súlyozott variancia a 13. ábrán látható az árfolyammal együtt.



13. ábra - Súlyozott várható érték és súlyozott variancia árfolyammal együtt

4.4. 1-Sigma intervallum

A súlyozott variancia és súlyozott várható érték együttes felhasználására más lehetőség is van. Segítségükkel behatárolható a log return-ben szereplő későbbi nap záróára két érték közé. Ehhez először a súlyozott várható érték felső és alsó határát határozom meg:

$$\mu_{1sigma\ upper} = \mu_{weighted} + \sigma_{weighted} , \quad (19)$$

$$\mu_{1sigma\ lower} = \mu_{weighted} - \sigma_{weighted} \quad (20)$$

Ezen értékek segítségével, valamint a logaritmikus hozamban szereplő korábbi nap árfolyamadatai alapján becsülhetjük meg az ár felső és alsó határait:

$$P_{1sigma\ upper} = P_{actual\ price} e^{\mu_{1sigma\ upper}} , \quad (21)$$

$$P_{1sigma\ lower} = P_{actual\ price} e^{\mu_{1sigma\ lower}} , \quad (22)$$

ahol $P_{actual\ price}$ a log return célváltozóban szereplő korábbi naphoz tartozó ár (esetemben a záró), természetesen feltéve, hogy a korábbi nap már ismert. Target data-nak a 3.3. szekcióban ismertetett log return változatokat használtam.

A felső és alsó határok meghatározásával jobban megérthetjük, hogy a kívánt nap záróára milyen tartományon belül mozoghat. Az így kapott ár alapú intervallumok világosabb képet adnak a modell bizonytalanságáról, mint a súlyozott variancia. Amennyiben az aktuálisan meghatározott intervallum szűkebb, mint az előző intervallumok, akkor ez arra utal, hogy a modell nagyobb magabiztossággal rendelkezik az adott előrejelzés kapcsán.

4.5. 2-Sigma intervallum

A 2-Sigma intervallum kiszámítása hasonló a 1-Sigma intervalluméhoz, azonban ebben az esetben a súlyozott varianciát kétszeres mértékben vesszük figyelembe. Ez szélesebb intervallumot eredményez. Csak abban az esetben érdemes pozíciót nyitni, ha az intervallum viszonylag szűk. Mivel a 2-Sigma intervallum minden esetben szélesebb, mint a 1-Sigma, ez csak akkor fordulhat elő, ha a modell nagy fokú bizonyossággal rendelkezik az előrejelzéseit illetően.

5. BACKTESTING

A backtesting nélkülözhetetlen eszköz a pénzügyi modellek fejlesztésében és ellenőrzésében. Ez a folyamat lehetővé teszi, hogy a modell teljesítményét történelmi adatokon teszteljük anélkül, hogy közvetlenül éles piaci környezetben használnánk. A módszer azt mutatja meg, hogy egy adott stratégia milyen eredményeket hozott volna a múltban, és segít feltárni a modell esetleges gyengeségeit vagy erősségeit, még mielőtt tényleges kereskedésre alkalmaznánk.

Adatok előkészítése: Az első lépés a múltbeli piaci adatok összegyűjtése és feldolgozása. Ebben a szakaszban kiemelt fontosságú az adatok minősége, hiszen a pontatlan vagy hiányos adatok hibás eredményekhez vezethetnek. A backtesting-et a nyers piaci adatokra végezzük, ahogy az a gyakorlatban is történne, de fontos, hogy a backtesting során a modell bemeneti adatai ugyanúgy legyenek előkészítve, ahogy a modell tanításakor is: például standardizálni kell az adatokat, szekvenciákat kell létrehozni, és a használt technikai indikátorokat kiszámolni, hogy a szimuláció eredményei hitelesek legyenek. Természetesen, ha már feldolgoztuk és felosztottuk egyszer az adatainkat, akkor használhatjuk a teszt vagy validációs halmazt a tesztelés során.

Stratégia végrehajtása: A teszt során a korábban begyűjtött adatokra alkalmazzuk a modellt és kereskedési stratégiát. A modell előrejelzéseit használjuk fel arra, hogy kereskedési döntéseket hozzunk, például vételi vagy eladási pozíciókat nyissunk. A modell által generált jeleket, mint például a várható logaritmikus hozamot, illetve a volatilitást, különböző szabályokkal kombinálhatjuk, ezzel kockázat kezelést létrehozva.

A teszt eredménye: A backtesting során hozott kereskedési döntések alapján különféle mutatók számíthatók, amelyek eltérő szempontok szerint értékelik a stratégia teljesítményét. Ilyen mutatók például a hozam, a volatilitás, a Sharpe-mutató, valamint a maximum drawdown. Ezek arra szolgálnak, hogy megmutassák, mennyire hatékony a stratégia, illetve milyen mértékben képes kezelni a kockázatokat különböző piaci feltételek között. Továbbá segítenek azonosítani, hogy a modell mennyire biztosít stabil eredményeket hosszú távon is, így lényeges szerepet játszanak a stratégia validálásában.

5.1. Teljesítmény mutatók

A backtesting fontosságának ismertetése után ebben a szakaszban részletesen definiálom az általam alkalmazott mérőszámokat. Lényeges megjegyezni, hogy a backtesting során nem csupán a modellt, hanem a kereskedési stratégiát is együttesen értékeljük. Ez azt jelenti, hogy ha bármelyikük gyenge vagy hibás, az összesített eredmény is kedvezőtlen lesz, függetlenül attól, hogy a másik mennyire jól teljesít. A legfontosabb mutatók közé tartoznak:

- **Hozam:** A hozam a befektetés értékének százalékban kifejezett változása egy adott időszakra. Ez az egyik legegyszerűbb és legfontosabb mérőszám.
- **Sharpe-ráta:** A hozam és a kockázat arányát méri, és egy gyakran használt mutató a stratégiák kockázatának értékelésére. A magas Sharpe-mutató azt jelzi, hogy a stratégia viszonylag alacsony kockázat mellett éri el a hozamot. A számítására használt képlet:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p} \quad (23)$$

, ahol R_p a kereskedésünk hozama, R_f a kockázatmentes kamatláb és σ_p az adott időintervallumon mért hozamok szórását adja meg.

- **Maximum drawdown (Maximális visszaesés):** Ez a mutató egy befektetés értékének legnagyobb csökkenését méri a csúcspontjától az azt követő mélypontig, mielőtt egy új csúcspontot érne el. Ez segít felmérni a stratégia kockázatkezelési képességét.
- **Expectancy (Várakozás):** Ez a mutató az átlagos nyereséget vagy veszteséget mutatja egy-egy ügyletre vetítve. A várható hozam az egyes ügyletek átlagos eredményét méri, és hosszú távú sikeresség előrejelzésében segíthet.
- **Win Rate (Nyeresési ráta):** A nyereséges ügyletek aránya a teljes ügyletszámhoz képest. Ez a mutató megmutatja, hogy a stratégia hányszor eredményezett nyereséget az összes ügylethez képest.
- **Évesített hozam:** Az évesített hozam az adott időszak hozama éves szintre levetítve.
- **Évesített volatilitás:** Az évesített volatilitást az egyes hozamok szórásaként határozzuk meg éves szintre skálázva.

6. KERESKEDÉSI STRATÉGIÁK

A kereskedési stratégiák kifejlesztése és alkalmazása elengedhetetlen a pénzügyi modellek valós piaci környezetben való sikeres használatához. Különböző stratégiákat hoztam létre attól függően, hogy a modell kimeneti értékeit hogyan értelmeztem (az eltérő értelmezéseket az 5. fejezetben ismertettem).

A kereskedési döntéseket különböző feltételekhez kötöttem. Az alábbiakban ismertetem az általam használt stratégiákat.

6.1. Súlyozott várható érték, súlyozott variancia használata

Ennek a stratégiának a használata esetén modell az előrejelzett logaritmikus hozam súlyozott átlagát (várható érték) adja meg (4.2. fejezet), amely közvetlenül összefüggésbe hozható a jövőbeli ár alakulásával. Az egyes pozíciók nyitásánál figyelembe vesszük a súlyozott varianciát is (4.3. fejezet), amely meghatározza a modell bizonytalanságát. Ha az előrejelzett értékek alapján a jövőbeli ár várhatóan emelkedni fog, vagyis a súlyozott átlag pozitív és a bizonytalanság mértéke kicsi, akkor egy long pozíciót nyitunk. Amennyiben az előrejelzett log hozam negatív, és a modell szintén alacsony szórást mutat, short pozícióba lépünk. Az alacsony szórást 1,5 alatti értéként definiáltam.

A kereskedési döntések így a modell magabiztosságán és a kedvező előrejelzett logaritmikus hozamon alapulnak, melyek kombinációja segíti a hatékony kereskedést.

6.2. Kumulált CDF használata

A modell kimeneteinek kumulált CDF formájában történő felhasználása, amint azt az 4.1. szekcióban részletesen kifejtettem, lehetőséget nyújt a logaritmikus hozam pozitív értékének bekövetkezését valószínűségi alapú előrejelzésére. A CDF számítása során nem csupán az átlag, hanem a szórás is szerepet kap, ezáltal a modell bizonytalanságát is mérjük, amely statisztikus formában jelenik meg.

A stratégia úgy épül fel, hogy amikor ez a valószínűség egy adott küszöbérték fölé emelkedik, long pozíciót nyitunk, ha pedig egy meghatározott érték alá csökken, akkor short pozíciót választunk. A backtesting során a long pozíció küszöbértékét 0,6-os szinten állapítottam meg, míg a short pozíció esetén ezt az értéket 0,4-ben határoztam meg, amely a long küszöb komplementere.

6.3. Súlyozott átlag és 1-Sigma intervallum használata

Amint azt az 4.4.-es fejezetben említettem, az 1-sigma számítása során a modell szórásokat használ arra, hogy intervallumokat állítson fel, amelyek valószínűsíthetően tartalmazzák logaritmikus hozam, illetve a jövőbeli ár alakulását. A stratégia alapja, hogy kereskedési döntést akkor hozunk, ha az előrejelzett intervallum mérete kisebb, mint az átlagos intervallum nagysága, mivel ilyenkor a modell nagyobb biztonsággal képes előrejelezni. Ezzel az óvatos megközelítéssel csak akkor nyitunk pozíciót, ha az intervallum nagysága biztonságosnak tekinthető.

A stratégia lényege, hogy long pozíciót akkor nyitunk, ha a súlyozott várható érték ($\mu_{weighted}$) pozitív és az aktuális 1-Sigma intervallum kisebb, mint az előző 10 napra számított átlagos intervallum. Short pozíciót pedig abban az esetben nyitunk, ha $\mu_{weighted}$ negatív és az 1-Sigma intervallum szintén kisebb az átlagosnál.

6.4. Stop loss, take profit meghatározása

A kereskedési pozíciók megnyitásakor alapvető fontosságú a stop loss (SL) és take profit (TP) értékek alapvető meghatározása, amelyek kulcsfontosságúak a kockázat kezelésében. A stop loss (SL) egy olyan előre meghatározott árfolyamérték, amelynél a pozíció automatikusan lezárul annak érdekében, hogy korlátozzuk a veszteségeket. Ezzel szemben a take profit az az ár melyet a befektetésünkkel el kívánunk érni, hogy nyereséget könyvelhessünk el.

6.4.1. Delta stop loss és take profit

A legáltalánosabb és legegyszerűbb módszer az SL és TP meghatározására az, hogy százalékos mértéket alkalmazunk a nyitási árhoz viszonyítva. Például a stop loss esetén egy olyan árfolyamot határozunk meg, amelynél legfeljebb 2%-os veszteséget vállalunk, míg a take profit egy olyan célárat jelent, amely 4%-os nyereség eléréséhez szükséges.

6.4.2. 1-Sigma intervallum használata

Ahogy az 5.4.-es fejezetben részleteztem, a 1-Sigma intervallummal az előre jelzett ár két határ közé szorítható, melyek ideális SL és TP szintekként szolgálhatnak. A long pozíció esetén take profitként a felső határt, stop lossként az alsó határt használom, short kereskedés esetén fordítva. Ezzel a módszerrel a modell előrejelzésein alapuló kockázatkezelést végzünk.

6.5. Pozíciók nyitvatartása

A pozíciók nyitvatartásának kezeléséhez egy dinamikus stop loss frissítési szabályt alkalmazok, amely a piaci mozgásokhoz igazítja a kockázatkezelést. Ha a pozíció három napon belül nem éri el a kitűzött take profit (TP) vagy stop loss (SL) szinteket, az SL értékét újraszámítom. Long pozíció esetén az új SL a háromnapos időszak legalacsonyabb ára és a korábban meghatározott SL közül a magasabb érték lesz. Ezzel a módszerrel a stop loss szint követi a részvény növekvő vagy konszolidált mozgását, így ha a részvény árfolyama fordul, és csökkenő tendenciába lép, az SL aktiválódásakor kisebb nyereség vagy kisebb veszteség realizálható. Short pozíció esetén hasonló módon történik az SL frissítése: az új SL a háromnapos időszak legmagasabb ára és a régi SL közötti alacsonyabb érték lesz. Ez a megközelítés lehetővé teszi a stop loss szintek adaptálását a piac mozgásaihoz, csökkentve a veszteségek kockázatát, miközben lehetőséget ad a nyereség maximalizálására.

7. HIPERPARAMÉTER OPTIMALIZÁCIÓ

A hiperparaméter optimalizáció alapvető szerepet játszik a modellek teljesítményének maximalizálásában. Míg a modellek tanulható paraméterei a training során kerülnek meghatározásra, addig a hiperparaméterek azok a konfigurációs beállítások, amelyek kívülről érkeznek, de szintúgy jelentősen befolyásolhatják a modell hatékonyságát, általánosíthatóságát.

A hiperparaméterek közé tartozhat a különböző rétegek neuronjainak száma, a tanulási ráta, az epoch-ok száma, a batch mérete, valamint bármilyen egyéb beállítás, amely befolyással lehet a modell teljesítményére. Mivel ezen paraméterek értékei eltérően hathatnak a modell teljesítményére, optimalizálásuk elengedhetetlen ahhoz, hogy a modell stabil és jól általánosítható legyen különböző adatokon is, mivel fontos, hogy a modell bármely új adaton konzisztensen jó teljesítményt érjen el.

A hiperparaméter optimalizáció során célunk az, hogy megtaláljuk azokat a beállításokat, amelyek a lehető legjobb eredményeket adják. Ehhez különböző algoritmusokat alkalmazhatunk. Például ilyen a random search, mely randomizáltan választ ki értékeket a megadott hiperparaméter paramétertartományból, a grid search, mely az összes általunk választott kombinációt végigpróbálja vagy fejlettebb megközelítések, mint a bayesi optimalizáció, amely valószínűségi megközelítést használ, így lehetővé téve a hiperparaméter tér hatékony bejárását.

Az Optuna egy rendkívül hatékony hiperparaméter optimalizáló könyvtár, amely rengeteg konfigurációs lehetőséget biztosít [32]. Meghatározhatjuk a hiperparaméterek lehetséges tartományait, valamint kiválaszthatjuk az optimalizáláshoz alkalmazott algoritmust. Az optuna alapbeállításként a TPE (Tree-structured Parzen Estimator) algoritmust használja, mely a bayesi algoritmust implementálja.

Fontos megfontolni, hogy mely értékre érdemes optimalizálni. Én a validációs halmazon mért backtesting eredményeire optimalizáltam a paramétereket. Többféle eredményre is kipróbáltam az optimalizációt, hogy megtaláljam a legideálisabb mutatót, ilyenek voltak például az elért hozam, valamint a Sharpe ráta, melyeket az 5.1. szekcióban ismertettem.

Amennyiben a teszt halmazt választottam volna a validációs halmaz helyett, akkor bár a modell nem finomítja a tanulható paramétereit a teszt adatokon, a hiperparaméter optimalizációs algoritmus mégis a külső paramétereket úgy állítaná be, hogy az adatokon a legjobb eredményt érje el. Ez pedig azt a kockázatot hordozza magában, hogy a modell a teszt adatokon jól teljesít, de más, ismeretlen adatokon már nem, ezáltal nem lenne képes megfelelően általánosítani.

8. TESZT EREDMÉNYEK

A teszteredményeket az Optuna hiperparaméter-optimalizáló segítségével értem el. Az optimalizáció során a modelleket betanítottam, majd a validációs adathalmazon végeztem backtesting-et. Ha a hozam egy meghatározott küszöbértéket meghaladott (például 10%-ot), akkor a modellt a teszt adatokon is kiértékeltem. Az Optunában a validációs halmazon mért backtesting során elért Sharpe-mutató volt az optimalizálandó célérték.

A táblázatban szereplő eredmények kiválasztása a teszt adathalmazon mért értékek alapján történt. A legmagasabb Sharpe-mutatóval rendelkező modellt választottam ki a teszt adathalmazon végzett backtesting alapján, majd a kiválasztott modell validációs halmazon mért backtesting eredménye automatikusan bekerült a következő táblázatba, amely a validációs halmaz eredményeit tartalmazza.

Az 1. és 2. táblázat azoknak a modelleknek az eredményeit foglalja össze, amelyeket a 6.2-es fejezetben leírt CDF-alapú módszerrel értékeltem. Az 1. táblázat a teszt adathalmazon, míg a 2. táblázat a validációs halmazon elért eredményeket mutatja be.

1. táblázat - CDF előrejelzés teszt adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	80.6449	58.0681	1.3879	-48.562922	66.6667	2.0928
<i>Multi-Head attention, MDN</i>	45.0519	72.4773	0.6209	-48.079483	48.1481	0.5368
<i>LSTM MDN</i>	69.8336	64.2089	1.0868	-39.381582	52.9412	1.0035
<i>GRU-MDN</i>	226.9289	277.1143	0.8187	-79.177846	55.1724	0.7178

2. táblázat - CDF előrejelzés validációs adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	34.8137	82.166	0.4231	-48.5629	52,5	0.611541
<i>Multi-Head attention, MDN</i>	156.5687	203.0162	0.771	-48.0794	50.0	1.788527
<i>LSTM MDN</i>	40.5852	64.7395	0.6261	-39.3815	46.666	0.74044
<i>GRU MDN</i>	32.4579	144.1874	0.2248	-79.1778	45.385	0.275899

A 3. és 4. táblázat az eredményeket mutatja be, amelyek a 6.1. fejezetben ismertetett súlyozott várható érték és súlyozott variancia stratégiának alkalmazásával készültek. A 3. táblázat a tesztadatokon mért eredményeket tartalmazza, míg a 4. táblázat a validációs adathalmazon elért eredményeket mutatja be.

3. táblázat - súlyozott várható érték és variancia stratégia alkalmazása teszt adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	35.4418	91.842	0.3854	-54.0983	52.0000	0.2749
<i>Multi-Head attention, MDN</i>	91.9136	135.5059	0.6779	-36.6445	55.8824	0.3848
<i>LSTM MDN</i>	360.5383	455.6278	0.7912	-52.3970	55.1724	0.6157

<i>GRU MDN</i>	35.8396	61.2328	0.5845	-42.8502	42.1053	0.6363
----------------	---------	----------------	--------	----------	---------	---------------

4. táblázat - súlyozott várható érték és variancia stratégia alkalmazása validációs adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	48.4045	169.9595	0.2845	-70.154676	44.897959	0.271378
<i>Multi-Head attention, MDN</i>	108.0202	271.911	0.3971	-62.467688	55.714286	0.417593
<i>LSTM MDN</i>	16.093	164.8868	0.0973	-88.291489	46.568627	0.195356
<i>GRU MDN</i>	57.1084	80.9015	0.7053	-29.535641	48.648649	0.775749

Az 5. és 6. táblázat azokat az eredményeket mutatja be, amelyeket a 6.3. fejezetben ismertetett 1-Sigma stratégia és a 6.4.2. fejezetben részletezett SL és TP 1-Sigma szerinti beállításai alapján értem el. Az 5. táblázat a tesztadatok alapján kapott eredményeket tartalmazza, míg a 6. táblázat a validációs adathalmazra vonatkozó eredményeket foglalja össze.

5. táblázat - 1-Sigma stratégia alkalmazása teszt adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	72.6741	153.6126	0.4728	-48.8434	48.4848	0.7606
<i>Multi-Head attention, MDN</i>	13.3103	133.3702	0.0994	-63.6680	49.4949	0.1689
<i>LSTM MDN</i>	55.7369	56.079	0.993	-16,446	62,5	1,9322
<i>GRU MDN</i>	213.2395	216.5527	0.9845	-35.6160	60.0000	1.5026

6. táblázat - 1-Sigma stratégia alkalmazása validációs adatokon

Modell	Évesített hozam [%]	Évesített volatilitás [%]	Sharpe-ráta [-]	Maximális visszaesés [%]	Nyerési ráta [%]	Expectancy (várakozás) [-]
<i>LSTM-GRU Multi-Head attention MDN</i>	44.8568	140.4849	0.3189	-58.696218	63.157895	0.866673
<i>Multi-Head attention, MDN</i>	21.9154	208.5192	0.1049	-80.249771	51.886792	0.238025
<i>LSTM MDN</i>	72.1758	143.6823	0.502	-66.138691	58,139535	0.983288
<i>GRU MDN</i>	12.5128	94.2828	0.1322	-63.128903	48.837209	0.422251

A teszteredmények összegzéséből látható, hogy a vizsgált három stratégia közül a CDF-alapú érte el a legmagasabb Sharpe-mutatót (1,3879) a teszt adathalmazon. Emellett ebben a stratégiában volt a legjobb a nyereségi ráta és a várakozás (expectancy) is, ami arra utal, hogy ez a megközelítés hozta a leginkább kiszámítható és kedvező eredményeket. A második legjobb Sharpe-mutatót (0,9845) a 1-Sigma stratégia adta.

A CDF stratégia esetében a legjobb eredményeket a *LSTM-GRU Multi-Head Attention MDN* modell érte el a Sharpe-mutató tekintetében, azonban a validációs halmazon mért értékek jelentősen alacsonyabbak voltak, ami némi instabilitásra utal. Ezzel szemben a *Multi-Head Attention MDN* modell stabilabb teljesítményt mutatott, mivel mind a teszt, mind a validációs halmazon hasonló Sharpe-mutatókat produkált (0,6209 és 0,771), ami arra utal, hogy ez a modell kevésbé érzékeny az adathalmazok közötti különbségekre.

A súlyozott várható érték és variancia stratégiát vizsgálva az *LSTM MDN* modell érte el a legjobb Sharpe-mutatót a teszt halmazon, azonban a validációs eredmények itt is alacsonyabbak voltak, ami az eredmények nagyobb volatilitását tükrözi. A *GRU MDN* modell ugyanakkor a validációs halmazon jobb teljesítményt mutatott, és a teszt halmazon is jól szerepelt, ami arra utal, hogy ebben a stratégiában ez a modell stabilabban viselkedett a különböző adatokon.

Az 1-Sigma stratégiát tekintve a *GRU MDN* modell bizonyult a legeredményesebbnek a Sharpe-mutató tekintetében a teszt halmazon, viszont a validációs eredményekben némileg visszaesett. Az *LSTM MDN* modell ehhez képest kiegyensúlyozottabb teljesítményt nyújtott: teszt adatokon közel azonos eredményeket ért el, és a validációs adatokon is stabilabban teljesített.

Összegzőképpen elmondható, hogy a CDF-alapú stratégia átlagosan jobb eredményeket hozott a többi stratégiához képest, különösen a Sharpe-mutató, a nyereségi ráta és a várakozás szempontjából.

REFERENCIA

- [1] Vuong, P. H., Phu, L. H., Van Nguyen, T. H., Duy, L. N., Bao, P. T., & Trinh, T. D. (2024). A bibliometric literature review of stock price forecasting: from statistical model to deep learning approach. *Science Progress*, 107(1), 00368504241236557.
- [2] Selvamuthu, D., Kumar, V., & Mishra, A. (2019). Indian stock market prediction using artificial neural networks on tick data. *Financial Innovation*, 5(1), 1-12.
- [3] Lin, G., Lin, A., & Cao, J. (2021). Multidimensional KNN algorithm based on EEMD and complexity measures in financial time series forecasting. *Expert Systems with Applications*, 168, 114443.
- [4] Xiao, J., Zhu, X., Huang, C., Yang, X., Wen, F., & Zhong, M. (2019). A new approach for stock price analysis and prediction based on SSA and SVM. *International Journal of Information Technology & Decision Making*, 18(01), 287-310.
- [5] Syukur, A., & Istiawan, D. (2021). Prediction of LQ45 Index in Indonesia Stock Exchange: A Comparative Study of Machine Learning Techniques. *International Journal of Intelligent Engineering & Systems*, 14(1).
- [6] Gunduz, H., Yaslan, Y., & Cataltepe, Z. (2017). Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, 137, 138-148.
- [7] Ghosh, P., Neufeld, A., & Sahoo, J. K. (2022). Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. *Finance Research Letters*, 46, 102280.
- [8] Oyewole, A. T., Adeoye, O. B., Addy, W. A., Okoye, C. C., Ofodile, O. C., & Ugochukwu, C. E. (2024). Predicting stock market movements using neural networks: a review and application study. *Computer Science & IT Research Journal*, 5(3), 651-670.
- [9] Maiti, A. (2020, November). Indian stock market prediction using deep learning. In *2020 IEEE REGION 10 CONFERENCE (TENCON)* (pp. 1215-1220). IEEE.
- [10] Teixeira Zavadzki de Pauli, S., Kleina, M., & Bonat, W. H. (2020). Comparing artificial neural network architectures for Brazilian stock market prediction. *Annals of Data Science*, 7(4), 613-628.
- [11] Singh, G. (2022). Machine learning models in stock market prediction. *arXiv preprint arXiv:2202.09359*.

- [12] Rather, A. M., Agarwal, A., & Sastry, V. N. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6), 3234-3241.
- [13] Md, A. Q., Kapoor, S., AV, C. J., Sivaraman, A. K., Tee, K. F., Sabireen, H., & Janakiraman, N. (2023). Novel optimization approach for stock price forecasting using multi-layered sequential LSTM. *Applied Soft Computing*, 134, 109830.
- [14] Bao, W., Cao, Y., Yang, Y., Che, H., Huang, J., & Wen, S. (2024). Data-driven stock forecasting models based on neural networks: A review. *Information Fusion*, 102616.
- [15] Yang, L., Ng, T. L. J., Smyth, B., & Dong, R. (2020, April). Html: Hierarchical transformer-based multi-task learning for volatility prediction. In *Proceedings of The Web Conference 2020* (pp. 441-451).
- [16] Chen, Y., Wei, Z., & Huang, X. (2018, October). Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 1655-1658).
- [17] Sony. (2024). *Sony global*. Retrieved October 30, 2024, from <https://www.sony.net/>
- [18] Baheti, P. (2021). Activation functions in neural networks [12 types & use cases]. V7 Labs. Retrieved October 29, 2024, from <https://www.v7labs.com/blog/neural-networks-activation-functions>
- [19] Békési, G., Barancsik, L., & Hartmann, B. (2024). Deep neural network based distribution system state estimation using hyperparameter optimization. *Results in Engineering*, 102908.
- [20] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.
- [21] Ming, Y., Cao, S., Zhang, R., Li, Z., Chen, Y., Song, Y., & Qu, H. (2017, October). Understanding hidden memories of recurrent neural networks. In *2017 IEEE conference on visual analytics science and technology (VAST)* (pp. 13-24). IEEE.
- [22] Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients?. *Advances in neural information processing systems*, 31.
- [23] Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8), 5929-5955.
- [24] Yang, S., Yu, X., & Zhou, Y. (2020, June). Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International*

workshop on electronic communication and artificial intelligence (IWECAI) (pp. 98-101). IEEE.

[25] AI Tech Trend. (2023). *LSTM vs GRU: Which one is better for recurrent neural networks?* AI Tech Trend. Retrieved October 29, 2024, from <https://aitechtrend.com/lstm-vs-gru-which-one-is-better-for-recurrent-neural-networks/>

[26] Raschka, S. (2023). *Understanding and coding the self-attention mechanism of large language models from scratch*. Sebastian Raschka. Retrieved October 29, 2024, from <https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html>

[27] Yahoo Finance. (2024). Yahoo Finance. Yahoo. Retrieved October 29, 2024, from <https://finance.yahoo.com/>

[28] Shubhlaxmi Commodity. (2024). Price, volume, and open interest. Shubhlaxmi Commodity. Retrieved October 29, 2024, from <https://www.shubhlaxmicommodity.com/p/price-volume-open-interest.html>

[29] Saturn Cloud. (2023, June 19). What Are Logarithmic Returns and How to Calculate Them in Pandas Dataframe. Retrieved from <https://saturncloud.io/blog/what-are-logarithmic-returns-and-how-to-calculate-them-in-pandas-dataframe/#why-are-logarithmic-returns-important-in-finance>

[30] Jan Niehues. (2024, October 5). *Evaluation*. Lecture presented in Advanced Artificial Intelligence, Department of Computer Science, Karlsruhe Institute of Technology.

[31] Fong, S. C. (2021). *Distributional prediction of foreign exchange rates with mixture density network* (Master's thesis). Imperial College London.

[32] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623-2631).