**Obsah obrázku Grafika, logo, grafický design, Písmo

Popis byl vytvořen automaticky**

**Mendelova univerzita v Brně** Mendel University in Brno

**Provozně ekonomická fakulta** Faculty of Business and Economics

**Ústav informatiky**Institute of Informatics

Predikce vývoje akciích na finančním trhu  
Processing of player and betting data

**Seminární PRÁCE** SEMINAR THESIS

**AUTOR PRÁCE** **Bc. David Michalica, Bc. Adam Pech**AUTHOR

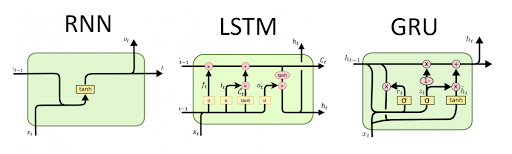
**VEDOUCÍ PRÁCE** **doc. Ing. František Dařena, Ph.D**.   
SUPERVISOR

**BRNO 2024**

# Úvod do problematiky

Náš projekt se zaměřuje na analýzu a predikci dat o vývoji cen amerických akcií. Cílem je vytvořit model, který dokáže na základě historických dat predikovat budoucí pohyby cen. Tato úloha je klíčová pro obchodníky, investory a další subjekty, kteří se snaží optimalizovat své strategie na finančních trzích. Vzhledem k povaze časových řad, které mají tendenci vykazovat sezónní vzory, cyklické trendy a náhlé změny, je nezbytné použít metody strojového učení schopné zpracovávat sekvenční data.

Jedním z přístupů, které jsou pro tuto úlohu vhodné, jsou rekurentní neuronové sítě (RNN), konkrétně varianty LSTM (Long Short-Term Memory) a GRU (Gated Recurrent Unit). Tyto modely umožňují efektivní učení z historických vzorů a dokážou zachovat informace o dlouhodobých závislostech v datech.



## Architektura modelu

Navržený model hlubokého učení využívá vrstvy LSTM nebo GRU, které jsou doplněny o pokročilé techniky, jako je:

* **Dávková normalizace (Batch Normalization):** Zlepšuje stabilitu a rychlost učení.
* **Regularizace výpadku (Dropout):** Snižuje riziko přeučení tím, že náhodně deaktivuje určité neurony během tréninku.
* **Husté výstupní vrstvy (Dense Layers):** Zajišťují transformaci reprezentací na finální predikci.

Model umožňuje konfiguraci pro různé délky vstupních sekvencí, horizonty predikce a kombinace vstupních funkcí, což umožňuje flexibilitu a přizpůsobení specifickým požadavkům.

# Popis vstupních dat

Yfinance je populární knihovna v Pythonu, která umožňuje snadné stahování finančních dat z Yahoo Finance. Tato data zahrnují historické informace o vývoji cen akcií, indexů, komodit a dalších finančních nástrojů. Pro náš projekt se zaměřujeme na klíčové parametry historických dat o akciích, které obsahují následující atributy:

* Datum (Date) - Každý záznam je spojen s konkrétním datem.
* Otevírací cena (Open) - Cena akcie při otevření obchodního dne.
* Zavírací cena (Close) - Cena akcie při ukončení obchodního dne.
* Nejnižší cena (Low) - Nejnižší hodnota, na kterou akcie dosáhla během obchodního dne.
* Nejvyšší cena (High) - Nejvyšší hodnota, na kterou akcie dosáhla během obchodního dne.
* Objem obchodování (Volume) - Celkový počet akcií, které byly obchodovány během dne.
* Adjusted Close (Upravená zavírací cena) - Zavírací cena upravená o vliv dividend, rozdělení akcií nebo dalších firemních akcí.
* Splits (Rozdělení akcií): Informace o změnách v počtu akcií společnosti.
* Metadata: Obsahují informace o firmě (sektor, tržní kapitalizace a další).

# Struktura projektu

Projekt je strukturovaný do několika složek a souborů, které budou popsány následovně. Projekt je takto strukturován pro lepší orientaci a udržitelnost kódu. Sktruktura je znázorněna následujícím seznamem:

* Složka dokumentace
* Složka data
  + data\_collector.py
  + data\_preprocesor.py
* Složka helper
  + data\_io.py
  + date\_validator.py
  + direktory\_setup.py
  + logger.py
  + market\_data\_calculator.py
  + result\_printer.py
* Složka models
  + model\_evaluator.py
  + prediction\_model.py
* Složka tools
  + corelation\_analyzer.py
  + data\_analyzer.py
  + elbow\_analyzer.py
  + kmenas\_analyzer.py
  + pca\_analyzer.py
  + TickerAnalysis.py
  + Training\_pipeline.py
* main.py
* config.py

# Soubor main.py

## Globální konstanty

Definuje parametry projektu, např.:

* **Časový rozsah dat:** START\_DATE, END\_DATE.
* **Cesty k souborům:** RAW\_DATA\_PATH, LOG\_FILE\_PATH.
* **Parametry dat:** SEQUENCE\_LENGTH, PREDICTION\_HORIZON, TRAIN\_RATIO, VAL\_RATIO.
* **Ticker akcie:** STOCK\_TICKER (např. "KO" pro Coca-Cola).

## Funkce

# Soubor data\_cleaner.py

Zpracovává chybějící hodnoty, odstraňuje sloupce s nedostatečnými údaji, čistí základní sloupce tržních dat a přizpůsobuje čtvrtletní finanční údaje denní frekvenci. Poskytuje podrobnou zprávu o čištění dat.

Třída StockDataCleaner slouží k:

* Odstranění sloupců s příliš velkým množstvím chybějících hodnot.
* Vyplňování chybějících hodnot (forward fill a backward fill).
* Čištění technických indikátorů, finančních metrik a dalších dat.
* Generování statistik a reportů o procesu čištění.

## Konstruktor \_\_init\_\_

### Argumenty:

* **min\_non\_null\_ratio**: Určuje minimální poměr nenulových hodnot ve sloupci, aby byl ponechán.

### Proměnné:

* **self.min\_non\_null\_ratio**: Nastavení tohoto parametru (default je 0.7).
* **self.cleaning\_stats**: Slovník, kde jsou ukládány statistiky o procesu čištění pro každý ticker.

## Metody

### clean\_stock\_data(stock\_data)

* **Vstup**: Slovník, kde klíče jsou tickery (např. "KO") a hodnoty jsou DataFrame s daty.
* **Výstup**: Slovník s vyčištěnými DataFrame pro každý ticker.
* **Průběh**:
  1. Pro každý ticker inicializuje statistiky o původních sloupcích a řádcích.
  2. Zavolá metodu \_clean\_single\_stock pro zpracování jednotlivého DataFrame.
  3. Odstraňuje tickery, jejichž data jsou po čištění prázdná.

### \_clean\_single\_stock(df, ticker)

* **Vstup**: DataFrame s daty o jednom tickeru.
* **Výstup**: Vyčištěný DataFrame.
* **Průběh**:
  + **Odstranění sloupců**: Sloupce s chybějících hodnot jsou odstraněny.
  + **Vyplňování důležitých sloupců**:
    - Důležité tržní sloupce: Open, High, Low, Close, Volume.
    - Technické indikátory: Daily\_Return, Volatility, MA50, MA200.
    - Finanční metriky: Sloupce začínající Financial\_.
* **Vyplňování zbývajících null hodnot**: Nejprve forward fill, poté backward fill.
* **Odstranění nekonečných hodnot**: Nekonečna se nahradí null hodnotami a následně vyplní.

## get\_cleaning\_report()

* Výstup: DataFrame s přehledem čištění:
  + Počet původních a konečných sloupců a řádků.
  + Počet odstraněných sloupců.
  + Seznam odstraněných sloupců.

## plot\_null\_percentages(ticker)

* **Vstup**: Ticker akcie.
* **Funkce**: Vizualizuje procento chybějících hodnot ve sloupcích.
* **Poznámka**: Používá matplotlib k vytvoření grafu. Pokud není dostupná, vrací chybu.

## Jak funguje proces čištění

### Odstranění sloupců:

* Pokud je více než 30 % hodnot ve sloupci chybějících (min\_non\_null\_ratio = 0.7), sloupec je odstraněn.
* Statistiky o chybějících hodnotách jsou ukládány.

### Vyplňování hodnot:

* Důležité sloupce (Open, Close, apod.) a technické indikátory (Daily\_Return, Volatility) jsou vyplněny metodou forward fill a backward fill.
* Ostatní sloupce jsou nejprve vyplněny dopředu (forward fill) a pak zpětně (backward fill).

### Odstranění nekonečných hodnot:

* Nekonečné hodnoty jsou nahrazeny NaN a poté znovu vyplněny.

### Konečný výstup:

* Vyčištěný DataFrame, který obsahuje pouze sloupce a řádky s dostatečnými daty.

## Statistiky čištění

* Každý ticker má statistiky o:
* Počtu původních a konečných sloupců/řádků.
* Odstraněných sloupcích a jejich důvodech.

# Soubor data\_collector.py

Sběr dat: Získává historická denní cenová data a čtvrtletní finanční výkazy pro seznam burzovních indexů pomocí rozhraní Yahoo Finance API. Vypočítá další technické ukazatele a finanční poměry.

Třída StockDataCollector slouží k:

* Sbírání denních dat o akciích z Yahoo Finance API pomocí knihovny yfinance.
* Výpočtu technických indikátorů (např. klouzavých průměrů, volatility).
* Převodu čtvrtletních finančních dat na denní frekvenci.
* Ukládání a načítání dat na/z disku.

## Konstruktor \_\_init\_\_

## Metody

# Soubor data\_preprocesor.py

## Konstruktor \_\_init\_\_

## Metody

# Soubor model\_evaluator.py

Vypočítá komplexní metriky hodnocení na testovací sadě včetně MSE, RMSE, MAE, R-squared, MAPE a směrové přesnosti. Vytváří vizualizace skutečných a předpovídaných cen a rozdělení chyb předpovědi. Generuje hodnotící zprávu.

## Konstruktor \_\_init\_\_

**Vstupy**:

* **model**: Trénovaný model pro predikci akcií.
* **preprocessor**: Instance předzpracovatele dat (např. škálování).
* **prediction\_horizon (int)**: Počet dní, na které model předpovídá dopředu.

## Metody

### Metoda evaluate\_predictions

**Vstupy:**

* **X\_test *(np.ndarray)***: Testovací vstupy (sekvence dat pro predikci).
* **y\_test *(np.ndarray)***: Odpovídající skutečné hodnoty pro testovací vstupy.
* **ticker *(str)***: Symbol akcie (pro výběr správného škálovače).

**Výstupy:**

* Slovník metrik hodnocení:
  + mse *(float)*: Střední kvadratická chyba.
  + rmse *(float)*: Kvadratická chyba v původní škále.
  + mae *(float)*: Střední absolutní chyba.
  + r2 *(float)*: Koeficient determinace.
  + mape *(float)*: Střední absolutní procentní chyba.
  + directional\_accuracy *(float)*: Přesnost směru pohybu ceny.

**Průběh:**

* Model předpoví hodnoty pomocí self.model.predict(X\_test).
* Predikce i skutečné hodnoty se škálují zpět na původní měřítko pomocí metody self.preprocessor.inverse\_transform\_predictions.
* Vypočítají se regresní metriky, např.:
  + **MAPE**: Vynechá se dělení nulami.
  + **Directional Accuracy**: Pomocí np.sign() porovná směr změny ceny mezi skutečností a predikcí.
* Uloží metriky a predikce do atributů třídy.

# Soubor prediction\_model.py

Predikce: Provádí předpovědi budoucích cen pomocí natrénovaného modelu na nových datech. Provádí inverzní škálování pro převod předpovědí zpět na původní cenovou stupnici.

## Konstruktor \_\_init\_\_

* sequence\_length (int): Počet časových kroků ve vstupní sekvenci.
* n\_features (int): Počet vlastností na časový krok.
* prediction\_horizon (int): Počet časových kroků, na které model předpovídá.
* use\_gru (bool): Určuje, zda použít vrstvu GRU namísto LSTM (výchozí: True).

## Metody

### Metoda \_build\_model

**Vstupy:** Žádné explicitní vstupy (využívá atributy třídy).

**Výstup:** Objekt modelu TensorFlow/Keras.

**Průběh:**

* Sestaví model pomocí Sequential API:
  + **Vstupní vrstva**: Očekává data tvaru (sequence\_length, n\_features).
  + **První rekurentní vrstva**:
    - Používá GRU nebo LSTM s 256 jednotkami.
    - Aplikuje regulární L2 penalizaci (0.0001).
  + **BatchNormalization** a **Dropout**: Stabilizují trénink a redukují přeučení.
  + **Druhá a třetí rekurentní vrstva**: Menší počet jednotek (128, 64) pro hierarchické extrahování rysů.
  + **Husté vrstvy**: Dvě husté vrstvy s aktivací relu (128, 64 jednotek).
  + **Výstupní vrstva**: Lineární aktivace pro regresní výstup o velikosti prediction\_horizon.
* Kompiluje model s optimalizátorem Adam a nízkou rychlostí učení (learning\_rate=0.00005).

### Metoda \_create\_callbacks

**Vstupy:** checkpoint\_path *(str)*: Cesta k souboru, kam se ukládají nejlepší váhy modelu.

**Výstup:** Seznam callbacků pro trénink.

**Průběh:**

* EarlyStopping:
  + Monitoruje validační ztrátu.
  + Zastaví trénink, pokud se ztráta nezlepší po 15 epochách.
* ModelCheckpoint: Ukládá nejlepší váhy modelu na základě validační ztráty.
* ReduceLROnPlateau: Snižuje rychlost učení, pokud se ztráta nezlepší po 5 epochách.

# Soubor training\_pipeline.py

Tréninkový kanál: Trénuje model pomocí zpracovaných dat se zpětnými voláními pro včasné zastavení, kontrolní bodování modelu, snížení míry učení a protokolování TensorBoard. Ukládá nejlepší váhy modelu a protokoly trénování.

## Konstruktor \_\_init\_\_

* model: Objekt modelu (instance obsahující Keras model).
* log\_dir (str): Adresář pro ukládání logů (výchozí: "training\_logs").

## Metody

# Průběh programu

Obsah obrázku text, snímek obrazovky, číslo, Písmo

Popis byl vytvořen automaticky

## Popis diagramu:

1. **User** vytvoří instanci **StockTrainingPipeline**.
2. **StockTrainingPipeline** vytvoří instanci **StockPredictionModel**.
3. **StockTrainingPipeline** zavolá **setup\_logging()** pro nastavení logování.
4. **StockTrainingPipeline** zavolá **create\_callbacks()** pro vytvoření všech potřebných callbacků (EarlyStopping, ModelCheckpoint atd.).
5. **StockTrainingPipeline** zavolá **train()** na modelu pro zahájení trénování.
6. **StockPredictionModel** zavolá metodu **fit()** pro trénování modelu.
7. **Callbacks** monitorují validaci a vykonávají akce na základě podmínek (EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, TensorBoard).
8. Po každé epoše jsou volány zpětné volání, která upravují trénovací parametry (např. EarlyStopping nebo změnu tempa u ReduceLROnPlateau).
9. Po dokončení trénování **StockTrainingPipeline** zavolá metodu **\_log\_training\_results()**, aby zaznamenal výsledky trénování.

# Výsledky projektu