UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Katedra za računarstvo

Automatizacija istraživanja kripto valuta

- Web mining -

Student:

Andrija Petrović 1751

Mentor:

prof. dr Miloš Bogdanović

Sadržaj

1 Uvod

1.1 Definicija problema

Ovaj rad se bazira na automatskoj trgovini **kripto valuta**, oslanjajući se na **sentimentalnu analizu vesti** u koheziji sa **tehničkom analizom** tj. proračunima finansijskih indikatora. Proračuni trenutnog sentimenta i izabranih indikatora, omogućavaju da se sa izvesnom dozom sigurnosti predvidi budući pravac kretanja cene. Za potrebe rešavanja problema, korišćen je **Python** programski jezik, u sprezi sa **veštačkom inteligencijom** i tehnikom prikupljanja informacija sa internet-a, zvanom **Crawling**. Izvorni kod se nalazi na Github sajtu.

1.2 Trgovina hartijama od vrednosti

Berza ili tržište hartija od vrednosti, je fizički i poslovno organizovan prostor, na kome se po strogo utvrđenim pravilima trguje hartijama od vrednosti, novcem i stranim sredstvima plaćanja. Jedna od najpoznatijih berzi na svetu, jeste svakako Njujorška berza (NYSE). Berze kripto valuta, ili tzv. kripto menjačnice, funkcionišu na gotovo identičan način kao i tradicionalne berze, s tim da je pojavom kripto menjačnica, došla **velika inovacija** mogućnosti trgovanja bez prekida (24/7). Glavna razlika između klasične berze i berze kriptovaluta, sastoji se u tome, da se kupovinom akcija na tradicionalnoj berzi zapravo kupuje udeo u firmi, čime se postaje akcionar firme. Broj akcija u posedu određuje „položaj" u firmi. Kod kripto valuta, kupovina se uglavnom realizuje iz razloga buduće finansijske dobiti, pri čemu se ne ostvaruje „vlasništvo" nad tim projektom, za razliku od tradicionalne berze.

Cena akcija na berzi, pa tako i kripto valuta, predstavlja se grafikonom tzv. sveća. Grafikon može vizuelizovati oscilacije cene tokom raznih vremenskih intervala.

Slika 1. BTC/USD, 4h interval

2 Web mining

2.1 Crawler vs Scraper

Web crawling i **web scraping**, predstavljaju tehnike izdvajanja podataka sa web sajtova. Vrlo često se oba termina upotrebljavaju u istom kontekstu, iako se metodologije rada veoma razlikuju.

Crawling, podrazumeva, korišćenje specijalnih alata, za kopiranje i čuvanje podataka web sajta, u svrhu **arhiviranja** ili **indeksiranja**. Svetski poznati pretraživači web-a, kao što su Google, Yahoo i Bing, zapravo koriste metodologiju Crawler-a za indeksiranje web stranica. Na osnovu sadržaja web sajta, gradi se index upita, kako bi se krajnjem korisniku servirao sadržaj koji potražuje. Crawler najpre poseti predefinisane početne web adrese, a nakon izdvajanja podataka agregira novo pronađenje linkove i usmerava se ka svakom od njih, ponavljajući isti proces iznova i iznova.

Scraping sa druge strane, predstavlja proces izdvajanja velike količine ciljanih podataka iz nekog izvora na web-u. Nakon učitavanja zadatih web adresa, scraper učitava HTML kod iz koga izdvaja željene podatke.

2.2 Scrapy crawler

Poređenjem brzine **BeautifulSoup** scraping biblioteke sa **Scrapy** crawler-om, utvrđeno je da je pomenuti crawler tri puta brže obradio zadate web adrese. Sa porastom broja adresa, razlika u brzini doseže eksponencijalne nivoe. Iako je Scrapy crawler neuporedivo teže konfigurisati, njegova robustnost itekako dolazi do izražaja i olakšava iterativne konfiguracije pojedinačnih crawler-a. Stoga je baš Scrapy biblioteka implementirana u Python-u korišćena za web mining, u daljem izlaganju.

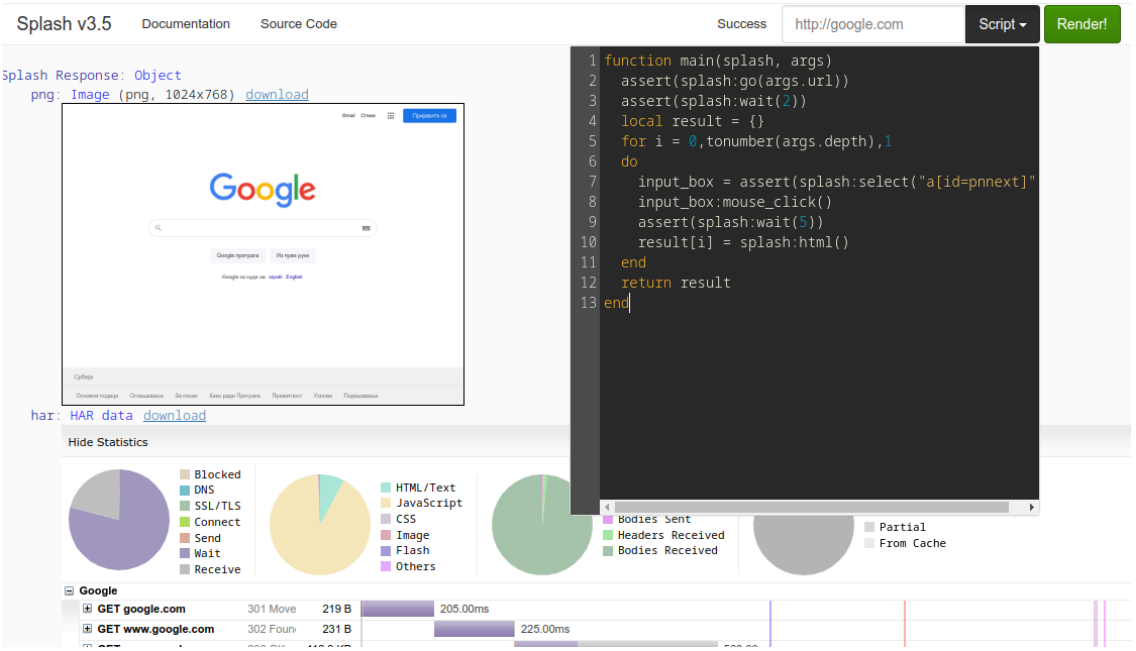
Uobičajeno Scrapy spider, započinje crawling proces, konstruisanjem generatora **scrapy.Request** zahtevom, integrisanog unutar **Scrapy biblioteke**. Ovakav početni zahtev, moguće je koristiti samo nad web sajtovima koji ne zahtevaju Java-Script engine. U suprotnom će povratni rezultat biti ograničen, ili nepostojeći.

```
yield scrapy.Request
```

Scrapy ne poseduje mogućnost učitavanja **Java-Script** koda, pa je kao rešenje korišćena **Scrapy-Splash** biblioteka, sa pratećim **Splash Docker** servisom. **Scrapy-Splash** docker kontejner, se pokreće kao zaseban servis, sledećom komandom:

```
docker run -p 8050:8050 scrapinghub/splash --max-timeout 3600
```

Tako se dobija server na lokalnoj mreži, koji osluškuje na portu 8050. Svaki **SplashRequest** zahtev, koji zapravo predstavlja nameru da se željena web adresa učita sa Java-Script-om, biva obrađen od strane Splash docker servera. Postoji mnoštvo alternativa Splash serveru (playwright, scrapy-selenium…), međutim, zbog najbolje integracije i prateće dokumentacije, u ovom radu, korišćen je Splash server i standardna selenium biblioteka kao rešenje za ograničenja na Cointelegraph sajtu. Splash poseduje lokalni web interfejs, putem kojeg je moguće, u realnom vremenu testirati **LUA skripte** uz **grafički prikaz**.



Slika 2. Splash LUA grafički prikaz

LUA skripte, za potrebe ovog projekta, pisane su za Google i Yahoo News web sajtove. Skripte se definišu kao varijable u Python-u i prosleđuju se **Splash docker serveru**, iniciranjem **SplashRequest zahteva**, u vidu **argumenta**, uz definisanje **endpoint** varijable kao „**execute**".

```
script_google_next_page = """
function main(splash, args)
    assert(splash:go(args.url))
    assert(splash:wait(2))
    -- collect search result links
    local results = {}
    results[#results + 1] = splash:html()
    for i = 0,tonumber(args.depth),1
    do
        local next_button = splash:select("a#pnnext")
        if next_button then
            next_button:mouse_click()
            assert(splash:wait(2))
            results[#results + 1] = splash:html()
        end
    end
    return {html=table.concat(results, '')}
end"""
```

Slika 3. Google LUA skripta

Google LUA skripta nakon učitavanja web strane sa definisanim termnom za petraživanje, pronalazi i inicira dugme Next u podnožju strane, u cilju postizanja definisane dubine pretrage i agregiranja ciljane količine članaka.

```
script_yahoo_next_page = """
function main(splash, args)
    splash:go(args.url)
    assert(splash:wait(2))
    -- bypass consent wall
    local consent_button = splash:select('button[name="agree"]')
    if consent_button then
        consent_button:mouse_click()
        assert(splash:wait(2))
    end
    -- collect search result links
    local results = {}
    results[#results + 1] = splash:html()
    for i = 0,tonumber(args.depth),1
    do
        local next_button = splash:select('a.next')
        if next_button then
            next_button:mouse_click()
            assert(splash:wait(2))
            results[#results + 1] = splash:html()
        end
    end
    return {html=table.concat(results, '')}
end"""
```

Slika 4. Yahoo News LUA skripta za agregaciju linkova pretrage

Yahoo News LUA skripta za agregaciju linkova pretrage funkcioniše nalik Google LUA skripti, uz dodatak zaobilaženja zida saglasnosti, iza kog se skriva HTML sadržaj. Zaobilaženje ove mere za prevenciju crawling-a postiže pronalaskom i iniciranjem dugmeta Agree, a potom uspešno agregira HTML sadržaj.

```
script_yahoo = """
function main(splash, args)
    splash:go(args.url)
    assert(splash:wait(2))
    -- accept cookies on yahoo finance
    local scroll_button = splash:select('button[id="scroll-down-btn"]')
    if scroll_button then
        scroll_button:mouse_click()
        assert(splash:wait(1))
    end
    -- bypass consent wall
    local consent_button = splash:select('button[name="agree"]')
    if consent_button then
        consent_button:mouse_click()
        assert(splash:wait(2))
    end
    return {html=splash:html()}
end"""
```

Slika 5. Yahoo News LUA skripta za učitavanje pojedinačnih linkova

Tokom testiranja Yahoo News LUA skripte za učitavanje pojedinačnih linkova agregiranih prethodnom Yahoo skriptom, Yahoo spider je naišao na još jednu prepreku u vidu novog zida saglasnosti, koji ima za cilj odobrenje korišćenja kolačića. Zaobilaženje ove prepreke je zahtevalo skrolovanje ka dole a zatim klik na dugme Agree.

Cointelegraph je tokom evaluacije ovog rada, implementirao mere za prevenciju web crawling-a, te inicijalno napisana Cointelegraph LUA skripta zapravo neće biti korišćena, a takođe ni Splash. Umesto toga, za iniciranje počenog zahteva korišćen je Selenium, koji se može tretirati kao ultimativno rešenje, ako alternativna rešenja ne daju zadovoljavajuće rezultate.

```
def start_requests(self):
    for ticker in self.TICKERS:
        try:
            driver = create_webdriver()
            driver.get("https://cointelegraph.com/search?query={}".format(ticker))
            time.sleep(5)
            for i in range(0, DEPTH_LIMIT):
                driver.execute_script(
                    "document.querySelectorAll('div.search-nav__load-more a')[0].click()"
                )
                time.sleep(5)
            # Extract Links
            links = []
            link_elements = driver.find_elements(
                By.XPATH, "//h2[@class='header']/a"
            )  # Cointelegraph extract links
            for el in link_elements:
                links.append(el.get_attribute("href"))
            for i, url in enumerate(list(set(links))):
                if url is not None:
                    yield scrapy.Request(
                        url, callback=self.parse, meta={"message": str(ticker)}
                    )
            driver.close()
            driver.quit()
        except Exception as e:
            print(print("Exception {}".format(e)))
```

Slika 6. Cointelegraph Selenium početni zahtev

Selenium je ovde iskorišćen za navigaciju do željene strane Cointelegraph sajt-a, nakon čega se iz HTML koda izdvajaju svi linkovi do članaka. Za svaki link se generiše *scrapy.Request*, čime Scrapy preuzima dalje praćenje linkova, a samim tim i finalno parsiranje istih.

2.3 Metode blokiranja crawler-a

Mnogi moderni web sajtovi, koriste dinamičko učitavanje sadržaja posredstvom Java-Script-a. U slučaju da pretraživač ne poseduje mogućnost renderovanja Java-Script funkcija ili ne uspeva da renderuje sve neophodne Java-Script funkcije, korisniku će pristup sadržaju biti delimično ili u potpunosti ograničen. Vid ograničenja može biti i nemogućnost navigacije, tj. prelaska na sledeću stranu, jer za interakciju sa pojedinim elementima mora se posedovati Java-Script engine. **Twitter** između ostalog, borbu sa crawler-ima i scraper-ima, vodi i pomoću implementacije **Content-Security-Policy** **(CSP) token**-a, koji onemogućava izvršenje Java-Script funkcija.

Blokiranje crawler-a implementacijom **CSP tokena**-a, može se zaobići na dva načina. Prvi način zahteva **kompajliranje Scrapy-Splash** docker kontejner-a, uz isključivanje **CSP tokena** iz zaglavlja izvornog koda. Drugi, jednostavniji način, podrazumeva korišćenje proxy servera, koji bi se nalazio ispred Splash servera, što omogućava modifikaciju zaglavlja, samim tim i uklanjanja **CSP tokena** iz istog. U ovom radu, neće biti implementiran Twitter crawler, iz razloga učestalog ponavljanja članaka.

Mnogi web serveri ne serviraju sadržaj pretraživačima bez definisanog **User-Agent** parametra u zaglavlju, ili blokiraju **IP adresu** koju crawler koristi, zbog previše zahteva u kratkom vremenskom periodu. Korišćenjem scrapy crawler-a uz dodatne alate, moguće je rešiti svaku od pomenutih potencijalnih blokada.

Izbor izvora članaka, sveden je na **Yahoo News**, **Google** i **Cointelegraph** kao jedan od najstarijih i najkredibilnijih izvora vesti, iz sveta kripto valuta. Za svaki od izvora, definisan je poseban **Spider**, zbog drastičnih razlika u pristupu rešavanja problema ograničenja crawling-a i izdvajanja tekstualnog sadržaja. Prilikom crawling-a izabranih izvora, nije moguće otići dalje od prve strane bez Java-Script engine-a. Izbegavanje blokiranja i vremenskog ograničenja, postignuto je implementacijom rotiranja: liste **Proxy IP** adresa i **User-Agent** zaglavlja. Proxy IP adrese se učitavaju iz tekstualnog fajla i to u formatu **Username:Password@IP:PORT**. Izbor User-Agent stringova zasnovan je na statistici korišćenih agenata iz realne baze podataka, ali takođe ima opciju da konfiguriše generator lažnih stringova, kao rezervu. Postoji i mogućnost definisanja fall-back User-Agent string-a. Rotacija se vrši pri svakom novom zahtevu. Sledi prikaz konfiguracije Google Spider-a:

```
class GoogleSpider(scrapy.Spider):
    name = "GoogleSpider"
    custom_settings = {
        "LOG_ENABLED": False,
        "DEPTH_LIMIT": DEPTH_LIMIT,
        "DOWNLOADER_MIDDLEWARES": {
            "scrapy_splash.SplashCookiesMiddleware": 723,
            "scrapy_splash.SplashMiddleware": 725,
            "scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware": 810,
            "scrapy.downloadermiddlewares.useragent.UserAgentMiddleware": None,
            "scrapy.downloadermiddlewares.retry.RetryMiddleware": None,
            "scrapy_fake_useragent.middleware.RandomUserAgentMiddleware": 400,
            "scrapy_fake_useragent.middleware.RetryUserAgentMiddleware": 401,
            #'rotating_proxies.middlewares.RotatingProxyMiddleware': 610,
            #'rotating_proxies.middlewares.BanDetectionMiddleware': 620,
        },
        "SPIDER_MIDDLEWARES": {
            "scrapy_splash.SplashDeduplicateArgsMiddleware": 100,
        },
        "DUPEFILTER_CLASS": "scrapy_splash.SplashAwareDupeFilter",
        "FAKEUSERAGENT_PROVIDERS": [
            "scrapy_fake_useragent.providers.FakeUserAgentProvider",  # this is the first provider we'll try
            "scrapy_fake_useragent.providers.FakerProvider",  # if FakeUserAgentProvider fails, we'll use faker to generate a user-agent
            "scrapy_fake_useragent.providers.FixedUserAgentProvider",  # fall back to USER_AGENT value
        ],
        "USER_AGENT": "Mozilla/5.0 (Linux; x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36",
        "HTTPCACHE_STORAGE": "scrapy_splash.SplashAwareFSCacheStorage",
        "REQUEST_FINGERPRINTER_IMPLEMENTATION": "2.7",
        #'ROTATING_PROXY_LIST_PATH':'../Lists/proxy_list',
    }
```

Slika 7. Google Spider

Konfiguracija **Cointelegraph** i **Yahoo** Spider-u, identična je priloženoj konfiguraciji **Google** spider-a.

2.4 Izdvajanje tekstualnog sadržaja

Princip izdvajanja sadržaja, iako različit zavisno od infrastrukture web sajta, moguće je uopšteno generalizovati. Najpre je potrebno generisati početni zahtev. Odgovor na početni zahtev biva obrađen u metodi parser-a, u kojoj se obično vrši izdvajanje pronađenih adresa, a koju je moguće definisati po nahođenju unutar klase Spider-a. Definisanje ove funkcije u mnogome zavisi od strukture web-sajta. Metoda *parse(self, response)* služi za izdvajanje linkova ka vestima, nakon pretrage, a zatim za slanje novih zahteva kako bi se ti linkovi pratili. Sledi prikaz metoda za finalno parsiranje tekstualnog sadržaja svih Spider-a.

```
def parse(self, response):
    # Extract clean URLS
    links = list(
        set(
            response.xpath(
                "//div[contains(@class, 'NewsArticle')]/ul/li/h4/a/@href"
            ).extract()
        )
    )
    for url in links:
        if (
            url is not None
            and not str(url).__contains__("cointelegraph.com")
            and not str(url).__contains__("cnyes.com")
            and not str(url).__contains__("beincrypto.com")
            and (str(url).__contains__(".com/") or str(url).__contains__(".net/"))
        ):
            yield SplashRequest(
                url,
                meta={"message": str(response.meta["message"])},
                callback=self.parse_content,
                endpoint="execute",
                args=self.splash_args_script_yahoo,
            )
```

Slika 8. Yahoo metoda parsiranja

Yahoo metoda parsiranja koristi *XPath* da pronađe URL-ove u okviru <*div*> elementa sa klasom *"NewsArticle"*. URL adrese su filtrirane tako da ne sadrže određene domene kao što su *"cointelegraph.com"*, *"cnyes.com"* ili "*beincrypto.com"*, zarad elimisanja neželjenih i dpulih izvora. Za svaki validan URL, šalje se novi *SplashRequest*, koji inicira renderovanje dinamičkog sadržaja. Ovaj zahtev prosleđuje URL, meta-podatke, i argumente, koristeći povratni poziv *self.parse_content*, kako bi Spider izvukao celokupni tekstualni sadržaj renderovanog HTML-a.

```python
def parse(self, response):
    links = []
    urls = []
    try:
        [
            links.append(r)
            for r in response.xpath("//div/a/@href").extract()
            if r not in links
        ]  # Google extract links and remove duplicates.
        for url in links:
            if "https://" in url:
                res = (
                    re.findall(r"(https?://\S+)", url)[0]
                    .split("&")[0]
                    .rstrip('\\"')
                )
                if (
                    not str(res).__contains__(".google.")
                    and not str(res).__contains__("cointelegraph.com")
                    and not str(res).__contains__("beincrypto.com")
                    and not str(res).__contains__("cnyes.com")
                    and (
                        str(res).__contains__(".com/")
                        or str(res).__contains__(".net/")
                    )
                    and not urls.__contains__(res)
                ):
                    urls.append(res)
                    yield scrapy.Request(
                        res,
                        callback=self.parse_content,
                        meta={"message": str(response.meta["message"])},
                    )
    except Exception as e:
        print(print("Exception {}".format(e)))
```

Slika 9. Google metoda parsiranja

Google metoda parsiranja je jako slična Yahoo metodi, s tim da se nakon prikupljanja članaka, za svaki od njih šalje ugrađeni *scrapy.Request*. Google nema dodatne mere zaštite, tako da nema potrebe za korišćenjem *SplashRequest-a*, koji omogućava renderovanje dinamičkog sadržaja.

Cointelegraph, slično Google pretraživaču, nema dodatne mere zaštite, te nakon prikupljanja članaka uz pomoć *selenium-a*, za svaki od njih takođe šalje ugrađeni *scrapy.Request*, nakon čega sledi finalno parsiranje teksta.

```python
def parse(self, response):
    global RESULTS_LIST
    try:
        text = [
            " ".join(
                line.strip()
                for line in p.xpath(".//text()").extract()
                if line.strip()
            )
            for p in response.xpath(
                "//body//p[not(ancestor::header) and not(ancestor::footer) and not(ancestor::nav)]"
            )
        ]
        text_str = " ".join([str(item) for item in text])
        if text_str not in RESULTS_LIST and text_str.strip():
            RESULTS_LIST.append(
                {
                    "Ticker": str(response.meta["message"]),
                    "Link": response.url,
                    "Text": text_str,
                }
            )
    except Exception as e:
        print(print("Exception {}".format(e)))
```

Slika 10. Cointelegraph metoda parsiranja

Iz odgovora ciljanog web sajta, moguće je izdvojiti celokupnu listu linkova koristeći **XPath** ili **CSS** upite. Zatim sledi generisanje zahteva za svaki od izdvojenih linkova i naposletku raščlanjivanje tekstualnog sadržaja unutar svakog **<p> HTML tag**-a, isključivo unutar tela HTML-a. Ovaj proces biva ponovljen onoliki broj puta, zavisno od broja web strana koje se obrađuju. Dovoljno je definisati **DEPTH_LIMIT**, a Scrapy crawler, automatski prestaje sa radom nakon postignute dubine pretraživanja članaka. **DEPTH_LIMIT** sa vrednošću 2, na primeru Google pretraživanja, označava da će Spider prikupiti sve članke sa prve i druge strane generisanih rezultata Google pretrage.

Spider-i startuju u isto vreme, što je omogućeno **CrawlerRunner procesom**. Postoji još jedan proces koji služi istoj svrsi, a naziva se **CrawlerProcess**. Međutim, korišćenjem **CrawlerProcess**-a nije bilo moguće startovati sve spider-e **istovremeno**, zbog grešaka koje su prijavljivane od strane **Reactor**-a. Zbog istih grešaka, nije bilo moguće čak, ni startovanje narednog spider-a nakon završetka prethodnog. Pomenuti problemi su rešeni implementacijom **CrawlerRunner procesa**.

```
def run_crawlers(TICKERS, TIMESPAN_NEWS_SEARCH, SPIDERS):
    print("=>>> Crawlers gathering articles... <<<=")
    configure_logging(install_root_handler=False)
    s = get_project_settings()
    s.update({"LOG_ENABLED": "False"})
    runner = CrawlerRunner(s)

    @defer.inlineCallbacks
    def crawl():
        for spider in SPIDERS:
            yield runner.crawl(
                getattr(eval(spider), str(spider)),
                TICKERS=TICKERS,
                TIMESPAN_NEWS_SEARCH=TIMESPAN_NEWS_SEARCH,
            )
        reactor.stop()

    crawl()
    reactor.run()

    # Get results of all crawlers.
    for spider in SPIDERS:
        CRAWLER_RESULTS[spider] = getattr(eval(spider), "get_" + str(spider))()
```

Slika 11. Startovanje Spider-a

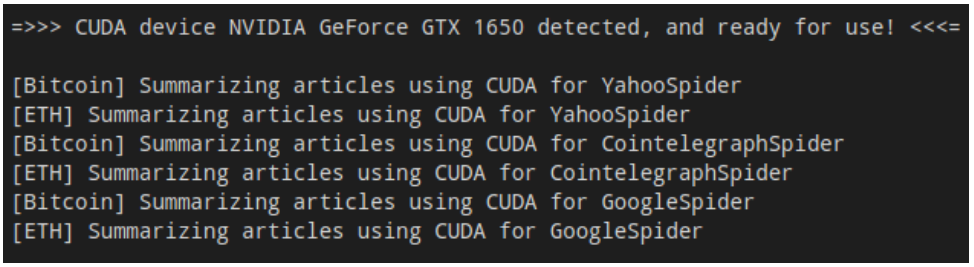
Po okončanju svih definisanih Spider-a, rezultati se agregiraju u varijablu, koja je tipa rečnik (*mapa*, u nekim programskim jezicima). Običnom filtracionom petljom agregiranih rezultata, dobijaju se ulazni podaci za proračun sentimenta.

3 Analiza sentimenta

3.1 NLP model

Transformeri, predstavljeni od strane Google AI tima, doveli su do ogromnog napretka u obradi prirodnog jezika (NLP). Inovacijom, istovremenog posmatranja rečenice sa obe strane, postignuta je znatno veća preciznost tumačenja teksta. Dubina razumevanja teme, o kojoj se kroz tekst govori, podignuta je do impozantnih visina.

Zbog velike količine teksta, model će biti treniran posredstvom **Nvidia** i **MAC silicon** grafičkim kartama. U slučaju da **PyTorch** biblioteka ne detektuje niti jednu od pomenutih grafičkih karti, ili je ista zauzeta drugim operacijama, definisan je fall-back na korišćenje isključivo procesora (zavisno od količine prikupljenih podataka, treniranje modela na procesoru može potrajati i nekoliko sati).



Slika 12. Analiza sentimenta

Tekstualni sadržaj kao izlaz iz svakog spider-a, za celokupnu listu kripto valuta od interesa, šalje se dalje na obradu **Pegasus NLP transformers** finansijskom modelu, treniranom na ogromnom setu podataka, isključivo iz sveta finansija (human-centered-summarization/financial-summarization-pegasus). Obzirom da model razume samo cele brojeve, najpre, pomoću **Tokenizer**-a, svaka reč u tekstu biva enkodirana kao niz celih brojeva. Model vrši sumarizaciju teksta, pri čemu na izlazu generiše tekst u vidu jedne rečenice, koja je takođe enkodirana kao niz celih brojeva. Pomoću Tokenizer-a, niz celih brojeva biva dekodiran nazad u čitljiv tekst.

Pegasus model, sumarizacijom velike količine teksta, zapravo interpretira suštinu o kojoj se govori. Primenom već treniranog modela sentimentalne analize na tako sažetu suštinu, takođe treniranog na skupu finansijskih članaka, vrši se procena sentimentalnog osećaja, a rezultat je formatiran na sledeći način:

'label': 'NEGATIVE', 'score': 0.97

Dakle, sentiment proračunat od strane izabranog modela, može biti pozitivan ili negativan, uz procenu nivoa sigurnosti u tačnost rezultata (score).

3.2 Rezultat

Izlazni rezultati modela, za svaki pronađeni članak, bivaju sačuvani unutar CSV i XLSX fajla. Rezultati se dakle čuvaju tabelarno, sortirani najpre po Spider-u koji je pronašao članak, a zatim i po pojedinačnom Ticker-u (simbolu) iz predefinisane liste kripto valuta.

Spider	Ticker	Summary	Sentiment	Sentim	URL
Yahoo	Bitcoin	CoinJoin, CoinSwap see	POSITIVE	0.976	https://bitcoinmagazine.cor
Yahoo	Bitcoin	'Enemy No. 1' is 'the so	NEGATIVE	0.999	https://thehill.com/blogs/bl
Yahoo	Bitcoin	Cathie Wood says Bitco	NEGATIVE	0.616	https://seekingalpha.com/n
Yahoo	Bitcoin	Conference in Miami la	NEGATIVE	0.999	https://www.fastcompany.c
Yahoo	Bitcoin	Thiel targets Buffett, Di	NEGATIVE	0.957	https://finance.yahoo.com/
Yahoo	Bitcoin	Dogecoin, Algorand, ar	POSITIVE	0.997	https://finance.yahoo.com/
Yahoo	Bitcoin	Cathie Wood says Bitco	NEGATIVE	0.978	https://ca.investing.com/ne
Yahoo	Bitcoin	Airport Warns Covid-Re	NEGATIVE	0.989	https://www.bloomberg.co
Yahoo	Bitcoin	Recovery phrase is a lis	NEGATIVE	0.827	https://www.fool.com/cryp
Yahoo	ETH	Vitalik Buterin donates	POSITIVE	0.991	https://www.benzinga.com
Yahoo	ETH	Ether was burned at a r	NEGATIVE	0.965	https://uk.investing.com/ne
Yahoo	ETH	Bitcoin, Ether fall sharp	NEGATIVE	0.997	https://finance.yahoo.com/
Yahoo	ETH	Gary Vaynerchuck's NF	NEGATIVE	0.995	https://finance.yahoo.com/
Yahoo	ETH	Ethereum rallied all the	POSITIVE	0.934	https://www.investing.com
Yahoo	ETH	Minutes indicated more	NEGATIVE	0.979	https://finance.yahoo.com/
Yahoo	ETH	Ethereum's value has su	POSITIVE	0.999	https://investorplace.com/2
Yahoo	ETH	Ethereum Classic set to	NEGATIVE	0.989	https://investorplace.com/2
Yahoo	ETH	Ethereum price formed	NEGATIVE	0.995	https://www.fxstreet.com/c
Cointelegraph	Bitcoin	BTC/USD drops below n	NEGATIVE	0.999	https://cointelegraph.com/r
Cointelegraph	Bitcoin	Veteran trader Melker	NEGATIVE	0.99	https://cointelegraph.com/r
Cointelegraph	Bitcoin	U.S. users will be able t	NEGATIVE	0.556	https://cointelegraph.com/r
Cointelegraph	Bitcoin	Fight of the Night and	POSITIVE	0.995	https://cointelegraph.com/r
Cointelegraph	Bitcoin	Roatn, Madeira, Portug	POSITIVE	0.969	https://cointelegraph.com/r

Slika 13. Sentiment - rezultati

Od interesa za dalje proračune, jeste i **totalni sentiment** zajedno sa odnosom **pozitivnog sentimenta** po Spider-u, za svaki simbol. Iz tog razloga, proračun totalnog sentimenta, biva sačuvan unutar zasebnog lista XLSX fajla:

Spider	Ticker	Positive ratio %	Total sentiment
YahooSpider	Bitcoin	0.29	NEGATIVE
YahooSpider	ETH	0.5	NEGATIVE
CointelegraphSpider	Bitcoin	0.66	NEGATIVE
CointelegraphSpider	ETH	0.62	NEGATIVE
GoogleSpider	Bitcoin	0.54	NEGATIVE
GoogleSpider	ETH	0.33	NEGATIVE

Slika 14. Totalni sentiment

4 Tehnička analiza

4.1 Obrada podataka

Problem koji treba rešiti je zapravo problem klasifikacije, jer se javljaju dve diskretne grupe, **up_candle** i **down_candle**. Cilj je, uzeti nove vrednosti (nove cene) i klasifikovati ih u pomenute dve grupe. Metodom transformacije, primenjena je lambda funkcija na grupe. Lambda funkcija korišćenjem funkcije **diff()**, poredi trenutnu cenu sa prethodnom cenom. Zatim se tako dobijeni rezultati umotavaju u funkciju **numpi.sign()**, koja na izlazu daje, **-1** za negativne vrednosti (pad cene) ili **1** za pozitivne vrednosti (rast cene). Ovi rezultati su predstavljeni u novo dodatoj koloni, **Prediction**.

Date	Open	High	Low	Close	Adj Close	Volume	Price_change	Prediction
2004-09-28	60.6336936950684	63.7617225646973	60.1632385253906	63.4914588928223	63.4914588928223	16877610	4.30416488647461	1
2004-09-29	63.3262977600098	67.5754089355469	63.176155090332	65.6035003662109	65.6035003662109	30358548	2.11204147338867	1
2004-09-30	65.0129318237305	66.2140960693359	64.5624923706055	64.8627853393555	64.8627853393555	13444942	-0.740715026855469	-1
2004-10-01	65.4633712768555	67.1850357055664	64.512451171875	66.3542327880859	66.3542327880859	15071372	1.49144744873047	1
2004-10-04	67.9757995605469	68.5013122558594	67.0799331665039	67.5954284667969	67.5954284667969	13002079	1.24119567871094	1
2004-10-05	67.3952407836914	69.3321151733398	66.1840667724609	69.2520370483399	69.2520370483399	14853991	1.65660858154297	1
2004-10-06	68.9016952514649	69.2920761108398	68.0658874511719	68.6064147949219	68.6064147949219	13342553	-0.645622253417969	-1
2004-10-07	68.3461608886719	70.0077667236328	68.3411560058594	69.4922714233398	69.4922714233398	14267999	0.885856628417969	1
2004-10-08	69.4322128295898	69.9076690673828	68.5763778686524	68.9317245483398	68.9317245483398	11041381	-0.560546875	-1

Slika 15. Yahoo podaci

Istorijske podatke ciljane kripto valute, dobijamo sa sajta Yahoo finance (Open, High, Low, Closing price...). Na osnovu njih vršimo proračune indikatora koji će služiti kao ulaz za algoritam. Indikatori od interesa svedeni su na:

['RSI' , 'STOCH' , 'Williams' , 'CMF']

Do ovog, suženog izbora indikatora, došlo se metodom dedukcije, primenom **feature importance** proračuna, koji zapravo govori, koji od korišćenih indikatora ima najveći uticaj u predviđanju buduće cene.

RSI	STOCH	Williams	MACD	ROC	OBV	ADX	CMF
50.4762457390916	82.1868721205709	-33.5261929921845	-1.88893929790265	0.430514454718577	-111839232	32.4339687942486	0.126802876142322
44.431065527564	62.4036203441748	-64.0514455410953	-2.37072716858165	0.355204488320092	-111839232	29.7415013831568	0.126802876142322
43.5712978447286	44.6413403334346	-68.4983404664163	-2.87982774589636	-0.0311728008896255	-111839232	28.0364781398426	0.0121325091811619

Slika 16. Indikatori

Proračunati indikatori se pohranjuju u **Random Forest Classifier** (klasifikator slučajne šume stabala), model veštačke inteligencije sa velikim brojem stabala. **Random Forest** je meta estimator koji uklapa veći broj klasifikatora stabla odlučivanja na različitim poduzorcima skupa podataka i koristi usrednjavanje da bi poboljšao tačnost predviđanja i kontrolisano prekomerno prilagođavanje. Veličina poduzorka se kontroliše pomoću parametra **max_samples** ako je **bootstrap=True** (podrazumevano), inače se ceo skup podataka koristi za pravljenje svakog stabla. Šuma stabala je korišćena u ovom radu iz razloga ostvarene veće preciznosti u odnosu na jedno stablo

odluke. Prilikom testiranja, razlika u preciznosti bila je približno 4% u korist šume stabala. Parametri jednog stabla, kojima je inicijalna preciznost ovog modela poboljšana za skoro 20%, dobijeni su testiranjem raznih vrednosti:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=6, criterion = "gini", random_state = 0)
```

Slučajna šuma stabala odluke se sastoji od 100 stabala odluke, a implementacija izgleda ovako:

```
RandomForestClassifier(n_estimators = 100, oob_score = True, criterion = "gini", random_state = 0)
```

Ulazni podaci bivaju podeljeni, pomoću **Sklearn** biblioteke, na skup za obuku i skup za testiranje. Kolone proračunatih indikatora, predstavljaju varijablu X, a Y kolona je zapravo kolona predikcije pada ili rasta cene, koja precizira da li je sveća zatvorila iznad ili ispod prethodne sveće.

```
X = df[indicatorsToUse]
Y = df['Prediction']
```

Nakon podele, moguće je prilagoditi podatke o obuci u model koristeći **fit** metodu:

```
fit(X_train.values, y_train)
```

Konačno, „obučenim" modelom, mogu se vršiti predviđanja buduće cene:

```
predict(X_test.values)
```

4.2 Rezultat

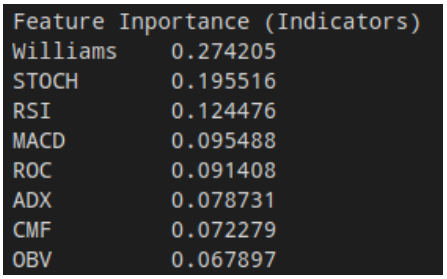
Trenirani **Random Forest Classifier model**, doseže veoma visok **nivo preciznosti predikcije** cene naredne sveće, od čak **83%**:

```
Correct RandomForestClassifier Prediction (%):  83.5676625659051
              precision    recall  f1-score      support
Down Day       0.838768  0.825312  0.831986   561.000000
Up Day         0.832765  0.845754  0.839209   577.000000
accuracy       0.835677  0.835677  0.835677     0.835677
macro avg      0.835766  0.835533  0.835597  1138.000000
weighted avg   0.835724  0.835677  0.835648  1138.000000
```

Slika 17. Predikcija

Prema vrednostima **recall** i **f1-score** parametara, koje su približno jednake tačnosti predikcije (precision), može se zaključiti da je model dobro utreniran i da zapravo uspeva da uči kombinovanjem pohranjenih informacija. U slučaju da se upravo razmotreni parametri znatno razlikuju, zaključak bi bio, da model zapravo ne uči ništa korisno, čime bi postao neupotrebljiv.

Feature Importance proračun omogućava zanemarivanje manje značajnih indikatora, zarad ubrzanja, a samim tim i minimizacije matematičkih proračuna indikatora. Pritom procenat tačnosti ostaje skoro nepromenjen. Udeo i značaj pojedinačnih indikatora u predhodnom predviđanju je sledeći:



Slika 18. Feature Importance

Prilikom pokušaja povećanja broja stabala u šumi stabala sa 100 na 2000 (Random Forest Classifier), uočena je stagnacija procenta tačnosti, uz ogroman porast vremena potrebnog za treniranje modela. Zaključak je da su izvučene maksimalne vrednosti postotka predviđanja pri opisanom pristupu rešavanja problema.

5 The real deal, let me trade

5.1 Proces trgovine

U ovom poglavlju, obrađena je validacija sistema koji je kreiran. Uz pomoć **API** interfejsa **Binance** menjačnice, trgovan je **Ethereum** (ETH) prema **Američkom dolaru** ($) na vremenskom intervalu od **30m**. Kako je kod pisan modularno, moguće je trgovati bilo koju kripto valutu, od ponuđenih na Binance menjačnici. Takođe je moguće izmeniti i vremenski interval.

Pozicije bivaju otvorene zavisno od dobijenih rezultata **tehničke** i **sentimentalne** analize. **Tehnička analiza** daje smer budućeg kretanja cene. Rezultati tehničke analize **filtriraju** se trenutnim **sentimentom**. U slučaju izrazito **negativnog sentimenta**, pozicije bivaju otvarane isključivo u smeru **Short** (pad cene) i to kada tehnička analiza takođe na izlazu pruži **Short signal**. Pozitivan ili blago negativan sentiment filtriraju moguće pozicije isključivo u smeru **Long** (rast cene).

Prilikom startovanja skripte, **Random Forest Classifier model**, najpre biva treniran istorijskim podacima u rasponu od **60 dana**, na odabranom vremenskom intervalu. U model zatim pohranjujemo trenutno proračunate vrednosti indikatora [‘RSI’,’STOCH’,’Williams’,’CMF’] i to pri kraju (isteku) svake sveće intervala od interesa, čime se dobija predikcija cene naredne sveće.

Zatim se vrši **proračun sentimenta** na velikom broju članaka. **Crawler** prikuplja sve članke sa **Google** i **Yahoo News** web sajtova do četvrte strane, što ukupno daje maksimalno 80 članaka po simbolu. Obzirom da **Cointelegraph**, po strani ima oko 50 članaka, **Scrapy zaustavlja navigaciju** na dubini 3. Definisan je filter, koji eliminiše članke koji se ponavljaju, tako da se na kraju dobija sentiment za veliki broj članaka, reda veličine **<= 230** po simbolu.

Na osnovu **predikcije**, a u sprezi sa **totalnim sentimentom** koji se proračunava jednom u toku dana, otvaramo **Short** ili **Long** poziciju na menjačnici. U slučaju **Long** pozicije profitirali bismo prilikom rasta cene, dok u slučaju pada cene gubimo novac (**Short** pozicija je inverzna opisanoj **Long** poziciji). Koristimo **leverage** (sistem poluge) od **100x** pri **profit/gubitak** odnosu **1:1**. Nivo na kome se zatvara pozicija, inkasiranjem **profita** je rast cene od **70%** u odnosu na trenutnu, što je već pomnoženo sa 100x leverage parametrom, tako da pri malom rastu cene ostvarujemo veliki dobitak. **Stop loss** (ograničenje gubitka) nivo, na kome zatvaramo poziciju realizirajući gubitak, definisan je padom cene od **-60%** u odnosu na trenutnu, takođe već pomnoženo sa leverage parametrom.

Dakle u slučaju pada cene od 60%, a po predikciji smo otvotili Long poziciju, zatvaramo poziciju u gubitku i time štitimo ostatak portfolia, kako ne bismo izgubili mnogo više novca u slučaju daljeg pada. U obratnom slučaju, ako cena poraste 70%, a po predikciji smo otvotili Long poziciju, istu zatvaramo u profitu.

5.2 Rezultat

Nakon **3 dana** simulacije trgovanja **ETH/USDT** para na vremenskom intervalu od **30m**, dobijen je rezultat od 8 dobitaka naspram 5 gubitaka, čime je **portfolio uvećan** za **149%**:

```
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: -61.16279553779549 [ TOTAL GAIN ]: -71.16279553779549 [ WIN/LOSS ]: 0/1
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 72.09649074705169 [ TOTAL GAIN ]: -9.0663047907438 [ WIN/LOSS ]: 1/1
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 70.1016745792856 [ TOTAL GAIN ]: 51.0353697885418 [ WIN/LOSS ]: 2/1
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 76.14728142648346 [ TOTAL GAIN ]: 117.18265121502526 [ WIN/LOSS ]: 3/1
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 70.21886531599648 [ TOTAL GAIN ]: 177.40151653102174 [ WIN/LOSS ]: 4/1
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: -60.05678649292747 [ TOTAL GAIN ]: 107.34473003809427 [ WIN/LOSS ]: 4/2
SELL ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: -61.88615589417452 [ TOTAL GAIN ]: 35.45857414391975 [ WIN/LOSS ]: 4/3
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 73.2279744578392 [ TOTAL GAIN ]: 98.68654860175894 [ WIN/LOSS ]: 5/3
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: -60.78912210607683 [ TOTAL GAIN ]: 27.89742649568211 [ WIN/LOSS ]: 5/4
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: -62.09666363714916 [ TOTAL GAIN ]: -44.19923714146705 [ WIN/LOSS ]: 5/5
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 71.00830258773385 [ TOTAL GAIN ]: 16.8090654462668 [ WIN/LOSS ]: 6/5
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 71.8297346754369 [ TOTAL GAIN ]: 78.6388001217037 [ WIN/LOSS ]: 7/5
BUY ][ SYMBOL ]: ETHUSDT [ GAIN NO FEE ]: 81.26469428401748 [ TOTAL GAIN ]: 149.90349440572118 [ WIN/LOSS ]: 8/5
```

Slika 19. Portfolio

Rezultat je zaista **odličan**, ali u cilju dalje validacije pouzdanosti modela i eventualnog rada sa pravim novcem, testiranje mora biti obavljeno u **mnogo dužem vremenskom periodu**, na **raznim vremenskim intervalima** sa **različitim odnosima profit/gubitak** (primer 3:1). Nakon sveobuhvatnog testiranja, parametri mogu biti podešeni tako da gubici budu minimalni, a dobici maksilmalni mogući. U tom slučaju čak i **postotak tačnosti predikcije** koji je **manji od 50%** može dovesti do značajnih finansijskih dobitaka.

Literatura

1. https://docs.scrapy.org/en/latest/intro/overview.html
2. https://github.com/alecxe/scrapy-fake-useragent
3. https://github.com/TeamHG-Memex/scrapy-rotating-proxies
4. https://splash.readthedocs.io/en/stable/faq.html#timeouts
5. https://huggingface.co/human-centered-summarization/financial-summarization-pegasus
6. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0
7. https://github.com/stefan-jansen/machine-learning-for-trading
8. https://ouyangjunaoyun.github.io/file/BigDataInFinanceReport.pdf
9. https://saiconference.com/Downloads/SpecialIssueNo10/Paper_3-A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf
10. Decision Trees - Andrew W. Moore
11. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html