AKTIENPREIS VORHERSAGE

Einleitung

Dieser Code wurde im Rahmen eines IC Kurses an der Universität St. Gallen geschrieben und ist somit nicht für tatsächliche Investitionen an der Börse zu verwenden, sondern dient der Demonstration was mit einem simplen Code machbar ist.

Der Code besteht aus den folgenden Phasen:

1. Auslesen der S&P 500 Ticker von Wikipedia (modifizierter Code)
2. Auslesen der historischen Daten für alle S&P 500 Firmen mit QUANDL (selbst erstellt in Anlehnung an diverse Vorlagen)
3. Vorhersage der zukünftigen Bewegungen mit statistischen Analysen (LSTM Model) (selbst erstellt in Anlehnung an diverse Vorlagen)
4. Weitere Datensammlung und -analyse
    a. Auslesen zusätzlicher relevanter Daten z.B. Gold-, Dollar- und Ölpreise. (selbst erstellt)
    b. Sentiment-Analyse und Summarizer von Zeitungsartikeln (selbst erstellt, jedoch viel vom VaderSentiment Package vorgegeben)

Das Ziel dieses Projektes ist es, relevante Börsendaten zu sammeln und zu analysieren um so ein Dashboard zusammenzustellen, welches private Investoren bei Investitionen mit kurzfristigem Zeithorizont unterstützen soll.

Anleitung

Um die oben genannten Resultate zu erreichen, müssen folgende Schritte verfolgt werden.

1. Installation sämtlicher Python Packages welche unter dem letzten Punkt «Requirements» dieses Dokuments aufgeführt sind
2. GitHub Repository mit GitKraken öffnen und somit eine **lokale Kopie** des GitHubs erstellen.
3. Dateipfad der lokalen Github Kopie herauslesen bis und mit «...\Short»
4. Sämtliche Python Dokumente («Additional_Data_extraction.py», «Prediction_for_all 500_tickers_as_loop.py", "Prediction_single_ticker.py", "SP500_historische_Daten_abrufen.py" & "Sentiment_Analysis_and_Summarizer.py") öffnen und im Code die Variable «DATEIPFAD», welche jeweils am Anfang des Dokuments steht mit dem lokalen GitHub Pfad ersetzen.
5. «SP500_historische_Daten_abrufen.py» laufen lassen.

6. «Prediction_for_all 500_tickers_as_loop.py ” laufen lassen, wenn alle Resultate gebraucht werden. Es dauert mehrere Stunden, alle Ticker durchlaufen zu lassen.
    a. Alternativ «Prediction_single_ticker.py” im Code die Variable «COLUMN” (am Anfang des Codes suchen) und abändern, um den gewünschten S&P 500 Ticker einzugeben.
7. «Additional_Data_extraction.py» laufen lassen um weitere Informationen zu ziehen. In der Liste «lst» können hier weitere QUANDL Ticker hinzugefügt werden je nach Bedarf.
8. «Sentiment_Analysis_and_Summarizer» laufen lassen, um vorher manuell erstellte CSV mit Zeitungsartikeln auf Sentiment zu analysieren und zusammenzufassen. CSV Dokument muss im Code ausgewählt werden. Drei Beispiele sind vorhanden, wobei immer zwei als Kommentar markiert werden müssen.
    a. Beispieldokumente unter «…\Short\Data\Manual News Data\»

Erklärungen

Auslesen der S&P500 Ticker

Als erstes wird als Teil des Pythondokuments **S&P_historische_Daten_abrufen.py** die S&P500 Liste aus Wikipedia herausgelesen. Hierfür wird der Webscraper BeautifulSoup verwendet. Alternativ wurde auch ein Code mit den Pandas und Read_HTML Packages geschrieben. Mit diesen beiden Anwendungen werden die Ticker der 500 Firmen ausgelesen und in ein CSV Dokument gespeichert mit dem Namen SP500.csv, welches in den GitHub Ordner gespeichert wird (unter «…\Short\Data\S&P 500 lists\»).

Auslesen der historischen Daten

Die zweite Phase ist erneut im Pythondokument **S&P_historische_Daten_abrufen.py** enthalten. In dieser Phase wird die eigentliche Datenbank, mit welcher später gearbeitet wird, erstellt. Mit der Quandl API werden für sämtliche Ticker die Daten herausgelesen (Funktion: «get_data_from_quandl()») und in einzelne CSV Dokumente gespeichert (unter «…\Short\Data\Historical data\by_ticker\»). Diese einzelnen CSV Dokumente werden in einem zweiten Schritt mit den Informationen Adjusted Close, Date & Ticker zusammengefügt (Funktion: «compile_data()»). Es wurde Adjusted Close gewählt (anstatt Close, Open, Volume, etc.), da der Adjusted Close bzgl. Aktiensplits angepasst wurde und somit über die Jahre hinweg vergleichbar ist. Dieser zweite Schritt speichert die Daten in das neu kreierte CSV Dokument sp500_joined_closes.csv als Matrix, mit dem Datum auf der y-Achse und den Ticker auf der x-Achse (unter «…\Short\Data\Historical data\»).

Zusätzlich wurde die Funktion «visualize_data()» geschrieben, welche die Korrelation der Daten visuell darstellt, damit man schnell sehen kann, dass z.B. bei einzelnen Firmen Daten fehlen. Dies dient lediglich zur Kontrolle und beeinflusst das Endresultat nicht.

Die Funktionen “extract_featuresets(ticker)» und «do_ml(ticker)» wurden als Kommentare belassen, da diese eine Machine Learning Methode wären um die zukünftigen Preise vorherzusagen. Jedoch waren die Resultate nicht ausreichend, womit diese von uns für das Endprodukt nicht weiter verwendet wurden.

Vorhersage der zukunftigen Preise

Für die Vorhersage wurden zwei Pythondokumente erstellt:
Prediction_for_all_500_tickers_as_loop.py & **Prediction_single_ticker.py**. Wobei das erste sämtliche S&P 500 Firmen durchgeht und für alle die Vorhersage erstellt, während das zweite Dokument für den Fall ist, wenn nur für eine Firma der Code laufen gelassen werden soll. In diesem Falle muss der Ticker jedoch manuell eingegeben werden. Da beide Dokumente im Aufbau sonst sehr ähnlich sind, gilt folgende Beschreibung für beide.

Der Code beginnt damit, dass das CSV Dokument sp500_joined_closes.csv geladen wird. Anschliessen werden die Daten für den gewählten Ticker ausgelesen. Anschliessend müssen die relevanten Daten für die weiteren Schritte vorbereitet werden. Hierzu werden zum einen mit dem «MinMaxScaler()» die Daten «normalized» und zum anderen aufgeteilt in 70% Trainingsdaten und 30% Testdaten.

Als nächstes wird das eigentliche Model vorbereitet. Für dieses Projekt wird ein Deep Learning Ansatz von Keras verwendet. Das sequentielle Model besteht aus einem LSTM Layer und zwei Dense Layern. Die resultierenden Vorhersagen werden anschliessend als CSV pro Ticker gespeichert (unter «…\Short\Data\Prediction data\by_ticker\» für loop und unter «…\Short\Data\Prediction data\test_single prediction\» für einzelne Ticker).

Die resultierenden CSV Dokumente sind nicht formatiert und benötigen eine manuelle Formatierung bevor sie für weitere Schritte verwendet werden können. Damit man die Qualität der Resultate einfach einsehen kann, wird - wenn alle Ticker durchlaufen wurden - noch ein zusätzliches Dokument erstellt (unter «…\Short\Data\Prediction data\mean_sqrd_error.csv»). Diese listet den Mean Squared Error (MSE) für jeden Ticker einzeln auf Je tiefer dieser Wert, desto höher ist die Genauigkeit der Vorhersagen im Vergleich zu den historischen Daten.

Weitere Datensammlung und Analyse

Um das Ziel eines umfangreichen und übersichtlichen Dashboards zu erreichen werden weitere Daten benötigt. Deshalb, wurde der Code **Additional_Data_extraction.py** aufgestellt. Dieser Code basiert auf einen ähnlichen Aufbau wie die bereits gesehenen. Die Daten werden mittels Quandl herausgelesen und in einem CSV Dokument ausgegebn (unter "...\Short\Data\Historical data\additional data\").

Zudem wurden Zeitungsartikel manuell herausgesucht. Diese werden mit dem Code im Pythondokument **Sentiment_Analysis_und_Summarizer.py** analysiert. Mit dem VaderSentiment Paket wird das Stimmungsniveau (=Sentiment) der einzelnen Zeitungsartikel analysiert. Mit dem Gensim Summarizer Paket werden dann noch Zusammenfassungen der Artikel kreiert. Die Resultate dieser Analyse werden wiederum als CSV gespeichert (unter «...\Short\Data\Manual News Data\Sentiment scores\»). Die Sentiment Resultate werden verwendet um zusätzliche Informationen aus den Nachrichten zu lesen, um auf einen steigenden oder sinkenden Kurs hindeuten. Des Weiteren wären die Zusammenfassungen ein möglicher Weg, die Zeitungsartikel in Kurzform den Nutzern verfügbar zu machen.

Es wurde auch ein Code geschrieben, welcher zuerst die Anmeldung bei Linkedin durchführt und anschliessend die Anzahl an Jobinserate für eine spezifische Firma (variabler Eintrag) und spezifische Region (variabler Eintrag) herauslesen lässt. Dieser Code kann im Ordner «LinkedinCode» gefunden werden, wobei der **loginweb.py** die Anmeldung macht und **CodeLinkedin.py** die Daten herauslesen soll.

REQUIREMENTS (Python packages)

- gensim
- vaderSentiment
- pandas
- numpy
- sklearn
- matplotlib
- bs4
- quandl
- pandas_datareader
- keras
- tensorflow
- importlib