Laporan Kecerdasan Web dan Big Data

Tugas 1: Scraping & Visualisasi Data

Disusun Oleh:

Azaria Putri Fawnia - 5024221038

Dosen Pengampu:

Arta Kusuma Hernanda

Departemen Teknik Komputer

Institut Teknologi Sepuluh Nopember

Semester Genap 2025

Daftar Isi

Daftar Gambar

Big Data 1

Penjelasan Umum

1.1 Latar Belakang

Di era digital, data telah menjadi aset yang sangat berharga. Kemampuan untuk mengumpulkan, mengolah, dan menganalisis data dalam jumlah besar (Big Data) membuka peluang baru di berbagai bidang, termasuk sektor finansial. Pasar valuta asing (foreign exchange) adalah salah satu pasar finansial terbesar dan paling likuid di dunia, dengan nilai tukar yang berfluktuasi secara konstan akibat berbagai faktor ekonomi dan politik.

Bagi investor dan analis keuangan, kemampuan untuk memprediksi pergerakan harga mata uang di masa depan adalah hal yang krusial untuk pengambilan keputusan. Secara tradisional, analisis ini dilakukan berdasarkan data historis yang disediakan oleh penyedia data finansial. Namun, dengan kemajuan teknologi web, kini data tersebut dapat dikumpulkan secara langsung dan real-time dari berbagai sumber di internet melalui teknik yang dikenal sebagai *web scraping*.

Proyek ini menggabungkan dua bidang ilmu komputer yang relevan: Kecerdasan Web dan Big Data. Pertama, kami menerapkan teknik *web scraping* untuk mengumpulkan data historis harga pasangan mata uang (currency pairs) dari sumber publik seperti Yahoo Finance. Data yang terkumpul kemudian disimpan dan dikelola, membentuk sebuah dataset time-series. Kedua, kami memanfaatkan dataset ini untuk membangun sebuah model prediksi menggunakan *Long Short-Term Memory* (LSTM), salah satu arsitektur *Recurrent Neural Network* (RNN) yang sangat efektif untuk menangani data sekuensial seperti data deret waktu.

1.2 Manfaat Tugas

1. Mahasiswa dapat memahami dan mengimplementasikan konsep-konsep kunci dalam Kecerdasan Web (web scraping) dan Big Data (pengolahan data, machine learning). Proyek ini memberikan pengalaman praktis dalam menggunakan library Python seperti `yahooquery`, `pandas`, `scikit-learn`, dan `tensorflow/keras`.

2. Menghasilkan sebuah sistem prototipe yang mampu memberikan prediksi harga mata

uang. Meskipun bukan untuk tujuan trading riil, model ini dapat menjadi dasar untuk alat bantu analisis yang lebih canggih.

3. Melatih kemampuan dalam seluruh siklus hidup proyek data, mulai dari pengumpulan data, pemrosesan data (preprocessing), pemodelan, hingga evaluasi dan visualisasi hasil.

Big Data 2

Scraping Data

2.1 Tujuan

Laporan ini bertujuan untuk menjelaskan terkait crawling data nilai tukar mata uang dari Yahoo Finance dan menyimpan data tersebut ke dalam file CSV untuk keperluan visualisasi dan analisis lebih lanjut.

2.2 Dasar Teori

- **Web Crawling**: Web crawling adalah proses otomatisasi untuk mengambil data dari halaman-halaman web secara sistematis. Proses ini dilakukan oleh program yang disebut *crawler* atau *spider*, yang akan mengunjungi situs web, membaca isinya, lalu mengambil data yang dibutuhkan. Dalam konteks tugas ini, web crawling digunakan untuk mengakses data nilai tukar mata uang secara langsung dari Yahoo Finance. Web crawling menjadi teknik penting dalam *data mining* dan *big data analysis* karena membantu mengumpulkan data dalam jumlah besar dari berbagai sumber secara cepat dan terstruktur.

- **Yahooquery**: `yahooquery` adalah sebuah pustaka (library) Python yang digunakan untuk mengambil data keuangan secara langsung dari Yahoo Finance API secara resmi maupun tidak langsung. Library ini mendukung berbagai jenis data seperti harga saham, nilai tukar mata uang, laporan keuangan, hingga data historis. Dalam tugas ini, `yahooquery` digunakan untuk mengambil harga pasar terbaru (*regular market price*), perubahan harga (*regular market change*), dan persentase perubahan harga (*regular market change percent*) dari berbagai pasangan mata uang dunia. Dibandingkan scraping manual, `yahooquery` menawarkan akses yang lebih stabil, terstruktur, dan mudah diolah.

- **Dataframe**: DataFrame adalah struktur data tabular dua dimensi yang disediakan oleh pustaka `pandas` di Python. Ia mirip dengan tabel pada basis data atau lembar kerja Excel, yang terdiri dari baris dan kolom. Setiap kolom dalam DataFrame memiliki nama, dan setiap baris memiliki indeks. Dalam proyek ini, DataFrame

digunakan untuk menyimpan hasil crawling data mata uang dengan format yang rapi, di mana setiap baris mewakili data satu pasangan mata uang pada satu waktu tertentu. DataFrame memudahkan manipulasi data seperti filtering, sorting, dan agregasi sebelum data tersebut disimpan ke dalam file CSV.

2.3 Alat dan Bahan

- Bahasa Pemrograman: Python
- Library: `yahooquery`, `pandas`, `datetime`
- Sumber Data: Yahoo Finance
- Software Tambahan: Visual Studio Code

2.3.1 Library yang Digunakan

Dalam implementasi program scraping data mata uang ini, beberapa library eksternal digunakan untuk mempermudah proses pengambilan, pengolahan, dan penyimpanan data. Berikut adalah penjelasan masing-masing library:

1. **yahooquery**

   `yahooquery` adalah library Python yang digunakan untuk mengambil data keuangan secara langsung dari Yahoo Finance melalui API yang disediakan. Dibandingkan metode scraping biasa, `yahooquery` jauh lebih cepat, andal, dan efisien karena bekerja dengan pendekatan berbasis API.

   Pada program ini, `yahooquery` digunakan untuk:

   - Membuat objek `Ticker` untuk daftar pasangan mata uang.
   - Mengambil informasi harga pasar saat ini (*regularMarketPrice*), perubahan harga (*regularMarketChange*), dan persentase perubahan harga (*regularMarketChangePercent*).

   Keunggulan `yahooquery` adalah mampu mengambil banyak instrumen keuangan dalam satu waktu tanpa perlu melakukan scraping manual halaman web.

2. **pandas**

   `pandas` adalah library Python yang sangat populer untuk manipulasi dan analisis data dalam bentuk tabel (struktur data `DataFrame`). Pandas menyediakan fungsi-fungsi yang mudah digunakan untuk membaca, menulis, dan memproses data dalam berbagai format seperti CSV, Excel, SQL, dan lain-lain.

   Dalam program ini, `pandas` berperan untuk:

   - Membuat `DataFrame` dari hasil data scraping agar data lebih terstruktur.
   - Menyimpan `DataFrame` ke dalam file CSV dengan fungsi `to_csv()`.

- Mengecek apakah file CSV sudah ada untuk menentukan apakah header perlu ditulis atau tidak.

Penggunaan `pandas` membuat proses penyimpanan dan pengolahan data menjadi jauh lebih sederhana dan terorganisir.

3. **datetime**

   `datetime` adalah library bawaan Python yang digunakan untuk memanipulasi dan merepresentasikan tanggal serta waktu. Library ini sangat berguna ketika program membutuhkan penandaan waktu (timestamp) dalam data yang dikumpulkan.

   Pada program ini, `datetime` digunakan untuk:

   - Mengambil waktu saat scraping berlangsung menggunakan fungsi `datetime.now()`.
   - Memformat waktu dalam bentuk `YYYY-MM-DD HH:MM:SS` dengan fungsi `strftime()`.

   Penambahan timestamp sangat penting untuk keperluan analisis waktu-seri di masa depan.

2.4 Prosedur Scraping Data

Proses scraping data mengikuti langkah-langkah berikut:

1. **Definisi Target**: Menentukan daftar pasangan mata uang yang datanya akan diambil.
2. **Inisialisasi Ticker**: Membuat objek `Ticker` dari library `yahooquery` dengan daftar pasangan mata uang yang telah ditentukan.
3. **Pengambilan Data**: Memanggil metode `.price` pada objek Ticker untuk mengambil data harga pasar terkini.
4. **Pemrosesan Data**: Melakukan iterasi pada setiap pasangan mata uang, mengekstrak informasi relevan seperti harga pasar, perubahan, dan persentase perubahan. Data ini disimpan bersama dengan stempel waktu saat scraping dilakukan.
5. **Penyimpanan Data**: Mengonversi data yang terkumpul ke dalam Pandas DataFrame dan menyimpannya ke dalam file `currency_rates.csv`. Skrip dirancang untuk menambahkan data baru ke file yang sudah ada (*append mode*) jika file tersebut sudah ada, sehingga data historis dapat terkumpul seiring waktu.

2.5 Instruksi Tugas

- Melakukan crawling data nilai tukar setiap mata uang dari Yahoo Finance.
- Menyimpan hasil crawling ke dalam file CSV.

2.6 Implementasi

2.6.1 Bagian 1: Scraping Data

Implementasi program dilakukan dengan langkah-langkah berikut:

1. **Menentukan Daftar Pasangan Mata Uang**

Code

```
from yahooquery import Ticker
import pandas as pd
from datetime import datetime

# Daftar pasangan mata uang
currency_pairs = [
    "EURUSD=X", "JPY=X", "GBPUSD=X", "AUDUSD=X", "NZDUSD=X",
    "EURJPY=X", "GBPJPY=X", "EURGBP=X", "EURCAD=X", "EURSEK=X",
    "EURCHF=X", "EURHUF=X", "CNY=X", "HKD=X", "SGD=X", "INR=X",
    "MXN=X", "PHP=X", "IDR=X", "THB=X", "MYR=X", "ZAR=X", "RUB=X"
]
```

Langkah pertama adalah menentukan pasangan mata uang (currency pairs) yang ingin diambil datanya. Pasangan mata uang seperti EUR/USD, GBP/JPY, hingga IDR/USD ditentukan di awal dan dimasukkan ke dalam sebuah daftar (`list`) di dalam program. Penentuan pasangan ini penting untuk memperjelas ruang lingkup scraping agar hanya data yang diperlukan yang diambil dari sumber.

2. **Menginisialisasi Objek Ticker dari Library yahooquery**

Code

```
tickers = Ticker(currency_pairs)
prices = tickers.price
```

Library `yahooquery` digunakan untuk berinteraksi dengan Yahoo Finance secara langsung. Dengan menggunakan objek `Ticker` dari `yahooquery`, seluruh pasangan mata uang yang telah ditentukan dapat diakses informasinya secara bersamaan. Hal ini memudahkan pengambilan data karena hanya membutuhkan satu kali inisialisasi untuk banyak instrumen sekaligus.

3. **Mengambil Data Harga Pasar**

Code

```
# Proses data
currency_data = []
timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  # Waktu scraping

for pair in currency_pairs:
    if pair in prices:
        price = prices[pair].get("regularMarketPrice", "N/A")
        change = prices[pair].get("regularMarketChange", "N/A")
        change_percent = prices[pair].get("regularMarketChangePercent", "N/A")
        currency_data.append([timestamp, pair, price, change, change_percent])
```

Setelah objek `Ticker` dibuat, data harga pasar diperoleh dengan memanggil atribut `price`. Data yang diambil untuk masing-masing pasangan mata uang meliputi:

- Harga pasar saat ini (*regularMarketPrice*)
- Nilai perubahan harga (*regularMarketChange*)
- Persentase perubahan harga (*regularMarketChangePercent*)

Data ini dikumpulkan dalam format dictionary, dengan masing-masing pasangan mata uang sebagai kunci.

4. **Menyusun Data dalam Struktur DataFrame**

Code

```
# Konversi ke DataFrame
df = pd.DataFrame(currency_data, columns=["timestamp", "currency_pair", "price", "change_value", "change_percent"])
```

Setiap data hasil scraping diproses menjadi sebuah list yang berisi timestamp pengambilan data, pasangan mata uang, harga, perubahan harga, dan persentase perubahan harga. Semua list kemudian dikumpulkan ke dalam sebuah `pandas DataFrame` agar data lebih terstruktur. DataFrame sangat membantu dalam menyusun data tabular sehingga dapat langsung diolah lebih lanjut atau disimpan ke dalam file.

5. **Menyimpan Data ke dalam File CSV**

Code

```
# Simpan ke CSV (append tanpa header jika sudah ada file)

df.to_csv("C:/zafaa/kuliah/SEMESTER6/BIGDATA/test2/currency_rates.csv",
mode='a',
        header=not
pd.io.common.file_exists("C:/zafaa/kuliah/SEMESTER6/BIGDATA/test2/
        currency_rates.csv"), index=False)

print("Scraping selesai! Data berhasil ditambahkan ke
'currency_rates.csv'.")
```

DataFrame yang telah berisi hasil scraping kemudian disimpan ke dalam file CSV (Comma Separated Values) menggunakan fungsi `to_csv()` dari pandas. File CSV disimpan di direktori lokal yang telah ditentukan. Untuk menghindari penulisan ulang file setiap kali scraping dilakukan, program dirancang untuk menambahkan data (*append*) ke file CSV yang sudah ada, serta hanya menulis header jika file tersebut belum tersedia.

6. **Menambahkan Timestamp**
   Setiap record data scraping dilengkapi dengan kolom `timestamp` yang berisi waktu pengambilan data dalam format `YYYY-MM-DD HH:MM:SS`. Timestamp ini sangat penting untuk menandai kapan data tersebut diambil, sehingga memungkinkan analisis tren atau perubahan harga seiring waktu di masa depan.

Berikut adalah listing program lengkap untuk scraping data:

Code

```
from yahooquery import Ticker
import pandas as pd
from datetime import datetime

# Daftar pasangan mata uang
currency_pairs = [
    "EURUSD=X", "JPY=X", "GBPUSD=X", "AUDUSD=X", "NZDUSD=X",
    "EURJPY=X", "GBPJPY=X", "EURGBP=X", "EURCAD=X", "EURSEK=X",
    "EURCHF=X", "EURHUF=X", "CNY=X", "HKD=X", "SGD=X", "INR=X",
    "MXN=X", "PHP=X", "IDR=X", "THB=X", "MYR=X", "ZAR=X", "RUB=X"
]

# Ambil data dari Yahoo Finance
tickers = Ticker(currency_pairs)
prices = tickers.price
```

```
16
17 # Proses data
18 currency_data = []
19 timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  # Waktu scraping
20
21 for pair in currency_pairs:
22     if pair in prices:
23         price = prices[pair].get("regularMarketPrice", "N/A")
24         change = prices[pair].get("regularMarketChange", "N/A")
25         change_percent = prices[pair].get("regularMarketChangePercent", "N/A")
26         currency_data.append([timestamp, pair, price, change, change_percent])
27
28 # Konversi ke DataFrame
29 df = pd.DataFrame(currency_data, columns=["timestamp", "currency_pair", "price",
     ↪ "change_value", "change_percent"])
30
31 # Simpan ke CSV (append tanpa header jika sudah ada file)
32 df.to_csv("C:/zafaa/kuliah/SEMESTER6/BIGDATA/test2/currency_rates.csv", mode='a',
33           header=not
     ↪ pd.io.common.file_exists("C:/zafaa/kuliah/SEMESTER6/BIGDATA/test2/
34           currency_rates.csv"), index=False)
35
36 print("Scraping selesai! Data berhasil ditambahkan ke 'currency_rates.csv'.")
```

Big Data 3

Visualisasi Data

3.1 Tujuan

Laporan ini bertujuan untuk menjelaskan terkait pengumpulan, penyimpanan, dan visualisasi data harga pasangan mata uang (currency pair) secara periodik, serta menganalisis stabilitas dan volatilitas masing-masing pasangan berdasarkan data tersebut. Visualisasi dilakukan menggunakan Metabase terintegrasi dengan database MySQL.

3.2 Dasar Teori

3.2.1 Currency Pair

Currency pair adalah harga relatif dari satu mata uang terhadap mata uang lainnya. Dalam dunia trading forex, sebuah currency pair, seperti EUR/USD, menunjukkan berapa banyak dolar AS (USD) yang diperlukan untuk membeli satu euro (EUR). Setiap pasangan memiliki nilai tukar yang terus bergerak berdasarkan dinamika pasar seperti suku bunga, tingkat inflasi, pertumbuhan ekonomi, dan kondisi geopolitik.

3.2.2 Data Visualization

Visualisasi data adalah teknik untuk merepresentasikan data secara grafis sehingga pola, tren, dan insight dapat dengan mudah dipahami oleh pengguna. Menurut *Tufte* (1983), visualisasi yang efektif memaksimalkan rasio data-ink, yakni memaksimalkan jumlah data yang ditampilkan relatif terhadap jumlah elemen grafis non-data. Dalam konteks proyek ini, visualisasi data membantu pengguna untuk memahami pergerakan harga currency pair secara lebih intuitif dan efisien.

3.2.3 Database MySQL

MySQL adalah sistem manajemen basis data relasional (RDBMS) open-source yang berbasis pada Structured Query Language (SQL). MySQL digunakan untuk menyimpan data currency pair dalam format tabel yang terstruktur, sehingga data dapat dikelola, diquery,

dan dianalisis dengan efisien. Beberapa fitur penting MySQL yang digunakan dalam proyek ini antara lain:

- **Auto Increment** untuk kolom `id`.
- **Primary Key** untuk menjaga keunikan data.
- **Datetime field** untuk penyimpanan waktu data dengan presisi.

3.2.4 Docker

Docker adalah platform berbasis container yang memungkinkan developer untuk mengemas aplikasi beserta semua dependensinya ke dalam sebuah container yang portabel dan ringan. Penggunaan Docker dalam proyek ini bertujuan untuk menjalankan Metabase tanpa harus melakukan instalasi manual di sistem operasi, meningkatkan portabilitas dan mengurangi kompleksitas setup.

3.2.5 Metabase

Metabase adalah aplikasi open-source untuk business intelligence (BI) yang memudahkan analisis data dan pembuatan dashboard tanpa perlu kemampuan teknis mendalam dalam SQL. Metabase dapat terhubung ke berbagai jenis database termasuk MySQL, dan menyediakan fitur seperti:

- Query Builder visual
- Filter interaktif
- Grafik dan chart otomatis
- Dashboard yang bisa dibagikan

Dalam proyek ini, Metabase digunakan untuk membuat visualisasi harga currency pair, menganalisis volatilitas pasar, dan menampilkan pergerakan harga secara real-time atau historis.

3.2.6 Time Rounding dan Aggregasi Data

Dalam analisis data berkala, penting untuk melakukan normalisasi waktu agar data dapat digabungkan dan dibandingkan dengan adil. Teknik *rounding time* digunakan untuk membulatkan waktu ke interval tertentu, seperti kelipatan 5 menit, sehingga memudahkan pengelompokan data. Selanjutnya, fungsi `AVG()` dalam SQL digunakan untuk menghitung harga rata-rata pada setiap interval waktu tersebut, mengurangi noise akibat fluktuasi harga mikro dan memperjelas tren utama.

3.3 Alat dan Bahan

- XAMPP (MySQL dan phpMyAdmin)
- Docker Desktop
- File CSV hasil scraping data currency pair
- Aplikasi Metabase (open-source, dijalankan lewat Docker)

3.4 Prosedur

3.4.1 Bagian 1: Visualisasi Data

1. Setup MySQL server menggunakan XAMPP.
2. Membuat database `currency_db` dan tabel `currency_rates`.
3. Import data CSV ke dalam tabel, dengan memastikan tidak ada duplikasi berdasarkan kolom `timestamp`.
4. Deploy Metabase menggunakan Docker:
    - Pull image Metabase: `docker pull metabase/metabase`
    - Jalankan container: `docker run -d -p 3000:3000 --name metabase metabase/metabase`
5. Hubungkan Metabase ke database MySQL `currency_db`.
6. Buat dashboard dan visualisasi dengan berbagai macam filter dan chart.

3.5 Skenario

Data scraping dilakukan secara otomatis setiap 5 menit, menghasilkan data harga, perubahan nilai (change value), dan perubahan persentase (change percent) untuk berbagai currency pair. Data yang dikumpulkan digunakan untuk menganalisis stabilitas, volatilitas, dan tren harga.

3.6 Instruksi Tugas

- Melakukan visualisasi data dengan dashboard Metabase dari data yang sudah di crawling sebelumnya.

3.7 Implementasi

3.7.1 Bagian 1: Visualisasi Data

1. Setup Database dan Import Data

1. Membuat database `currency_db`.

2. Membuat tabel `currency_rates` dengan kolom:
    - id (Primary Key, Auto Increment)
    - timestamp (Datetime)
    - currency_pair (Varchar)
    - price (Double)
    - change_value (Double)
    - change_percent (Double)
    - rounded_time (Datetime, hasil pembulatan timestamp ke 5 menit terdekat)
3. Import file CSV menggunakan phpMyAdmin dengan fitur *Import* dan memastikan tidak ada duplikasi menggunakan query SQL untuk membandingkan `timestamp`.

2. Setup Metabase via Docker

- Jalankan perintah: `docker run -d -p 3000:3000 --name metabase metabase/metabase`
- Akses Metabase di browser melalui `localhost:3000`.
- Integrasikan Metabase dengan database MySQL `currency_db`.

3. Pembuatan Dashboard dan Visualisasi

- Membuat **New Dashboard** di Metabase.
- Membuat berbagai **New Question**:
    1. **Grafik Harga per Pair**:
        - Visualisasi: Line Chart.
        - Summarize: Average of price.
        - Group By: rounded_time (per 5 menit atau per jam).
        - Filter: currency_pair dan tanggal.
    2. **Pair Paling Volatil**:
        - Summarize: Standard deviation of price.
        - Group By: currency_pair.
        - Sort: dari nilai standar deviasi tertinggi.
    3. **Pair Paling Stabil**:
        - Summarize: Standard deviation of price.
        - Group By: currency_pair.
        - Sort: dari nilai standar deviasi terendah.
    4. **Tren Perubahan Nilai**:
        - Visualisasi change_value dan change_percent dalam Line Chart.

5. **Insight Summary**:
   - Menggunakan fungsi aggregation: AVG(price), MAX(price) per hari.

- Menambahkan **Global Dashboard Filter** untuk memilih `currency_pair` dan `rounded_time`.

3.8 Analisis

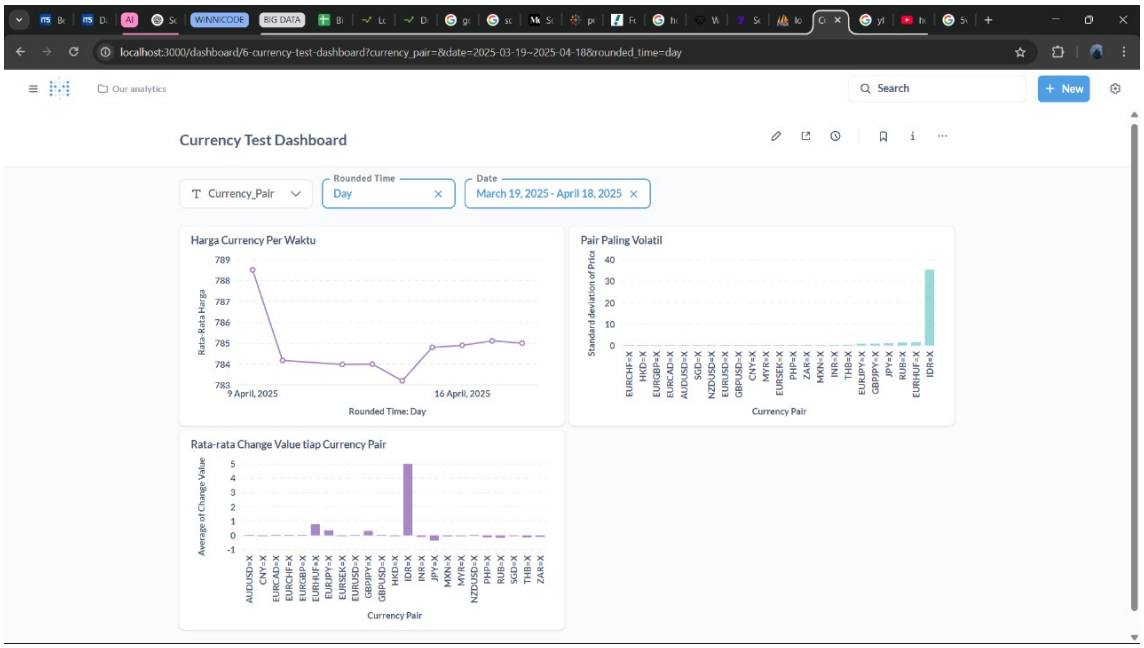

Figure 3.1: Dashboard Metabase Market Forex

Berdasarkan visualisasi yang dihasilkan:

- Pair dengan volatilitas tertinggi menunjukkan potensi peluang trading yang lebih besar namun juga risiko lebih tinggi.
- Pair yang stabil cocok untuk konservasi nilai dan analisis tren jangka panjang.
- Grafik harga per waktu membantu mengidentifikasi pola pergerakan harga.
- Filter tanggal dan pair memungkinkan analisis spesifik per hari atau per event tertentu.

Big Data 4

Implementasi LSTM

4.1 Tujuan

Tujuan dari bab ini adalah untuk mengimplementasikan model Long Short-Term Memory (LSTM) untuk melakukan prediksi harga pasangan mata uang (currency pair) berdasarkan data historis. Model akan dipakai untuk memprediksi harga pada tanggal dan jam tertentu yang dimasukkan oleh pengguna.

4.2 Dasar Teori

- **LSTM**: LSTM adalah salah satu jenis jaringan saraf tiruan dalam keluarga Recurrent Neural Network (RNN) yang dirancang untuk mengatasi permasalahan long-term dependency. LSTM memiliki struktur gate (input gate, forget gate, output gate) yang memungkinkan informasi penting dipertahankan selama proses pembelajaran urutan data. Model ini sangat efektif dalam menangani data time-series seperti pergerakan harga mata uang.

4.3 Library yang Digunakan

Library yang Digunakan antara lain:

- **Pandas**: untuk manipulasi dan analisis data tabular.
- **NumPy**: untuk perhitungan numerik dan array.
- **Matplotlib**: untuk visualisasi grafik hasil training dan prediksi.
- **scikit-learn**: untuk preprocessing dan evaluasi model (scaling, MSE, MAE, R2).
- **TensorFlow (Keras)**: untuk membangun, melatih, dan mengevaluasi model LSTM.
- **datetime**: untuk menangani dan menghitung perbedaan waktu prediksi.

4.4 Prosedur Implementasi

Proses implementasi model mengikuti langkah-langkah berikut:

1. Membaca dan memproses data historis harga mata uang dari file CSV.
2. Melakukan normalisasi data menggunakan MinMaxScaler.
3. Membuat dataset berurutan untuk pelatihan dan pengujian model.
4. Membangun dan melatih model LSTM.
5. Melakukan evaluasi performa model dan membandingkannya dengan naive forecast.
6. Melakukan prediksi harga di masa depan berdasarkan input tanggal dan jam dari pengguna.
7. Menampilkan grafik hasil prediksi dan evaluasi model.

4.5 Implementasi

Model LSTM ini digunakan untuk melakukan prediksi harga pasangan mata uang (seperti USDJPY, EURUSD, dll) berdasarkan data historis yang dikumpulkan secara berkala. Data yang digunakan mencakup timestamp, currency pair, dan harga. Pengguna dapat memilih pasangan mata uang dan tanggal target untuk memprediksi harga di waktu yang ditentukan. Model LSTM dipilih karena kemampuannya dalam menangkap dependensi jangka panjang dalam data time series.

Implementasi program dilakukan dengan langkah-langkah berikut:

1. **Impor Library dan Pembacaan Data**

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score,
    ↪ mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.callbacks import EarlyStopping
from datetime import timedelta, datetime

df = pd.read_csv('test2\\currency_rates.csv', parse_dates=['timestamp'])
```

Potongan kode ini berfungsi mengimpor seluruh library yang diperlukan:

- `pandas`, `numpy`, dan `datetime` digunakan untuk pengolahan data dan manipulasi waktu.
- `matplotlib.pyplot` digunakan untuk visualisasi grafik.
- `MinMaxScaler` dari `sklearn` digunakan untuk normalisasi data harga ke rentang 0-1.
- `Sequential`, `LSTM`, dan `Dense` adalah komponen dari Keras (TensorFlow) untuk membangun dan melatih model LSTM.
- `EarlyStopping` digunakan agar pelatihan berhenti otomatis saat validasi loss tidak membaik dalam sejumlah epoch.

Dataset kemudian dimuat dari file `currency_rates.csv`, dengan parsing otomatis pada kolom `timestamp` sebagai datetime.

2. **Fungsi Utama Prediksi**

Code

```
def predict_currency(currency_pair='USDJPY', target_datetime=None,
    ↪ look_back=60):
    data = df[df['currency_pair'] == currency_pair].copy()
    data = data.sort_values('timestamp')
    prices = data[['price']].values

    scaler = MinMaxScaler()
    scaled_prices = scaler.fit_transform(prices)

    X, y = [], []
    for i in range(look_back, len(scaled_prices) - 1):
        X.append(scaled_prices[i - look_back:i, 0])
        y.append(scaled_prices[i + 1, 0])
    X, y = np.array(X), np.array(y)
    X = X.reshape((X.shape[0], X.shape[1], 1))

    split = int(len(X) * 0.8)
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]
```

Fungsi `predict_currency` dimulai dengan memilih pasangan mata uang tertentu dari dataset. Setelah disorting berdasarkan waktu, hanya kolom harga yang diambil.

Kemudian data harga dinormalisasi agar lebih stabil saat dilatih oleh model LSTM. Normalisasi ini penting karena neural network sangat sensitif terhadap skala nilai input.

Dataset sekuensial dibentuk dengan parameter `look_back` sebesar 60, artinya setiap

sampel input akan berisi 60 harga sebelumnya untuk memprediksi 1 harga selanjutnya.

Setelahnya, data dibagi menjadi 80% data latih dan 20% data uji untuk proses evaluasi performa model.

3. **Pembangunan dan Training Model**

Code

```
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X.shape[1], 1)),
    LSTM(50),
    Dense(1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
early_stop = EarlyStopping(monitor='val_loss', patience=5,
    ↪ restore_best_weights=True)

history = model.fit(
  X_train, y_train,
  epochs=50,
  batch_size=32,
  validation_split=0.1,
  callbacks=[early_stop],
  shuffle=False,
  verbose=1
)
```

Model LSTM terdiri dari dua lapisan LSTM masing-masing dengan 50 unit. Lapisan pertama menggunakan `return_sequences=True` agar output-nya bisa dipakai oleh lapisan LSTM kedua.

Lapisan akhir adalah `Dense(1)` yang bertugas memetakan output ke satu nilai harga prediksi.

Model menggunakan optimizer `adam` dan fungsi loss `mean squared error (MSE)`, yang umum digunakan dalam regresi time series.

Early stopping diaktifkan agar pelatihan berhenti lebih cepat saat tidak ada perbaikan loss validasi selama 5 epoch berturut-turut.

Model dilatih selama maksimum 50 epoch dengan batch size 32, dan 10% dari data latih digunakan sebagai data validasi.

4. **Evaluasi Model dan Prediksi**

Code

```
y_pred_scaled = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred_scaled.reshape(-1, 1))
y_true = scaler.inverse_transform(y_test.reshape(-1, 1))

mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

naive_pred = scaler.inverse_transform(X_test[:, -1, :])
```

Setelah model selesai dilatih, dilakukan prediksi terhadap data uji. Nilai prediksi dan label sebenarnya dikembalikan ke skala harga asli dengan inverse transform dari `MinMaxScaler`.

Untuk evaluasi performa digunakan tiga metrik:

- **MSE (Mean Squared Error)**: menunjukkan rata-rata error kuadrat.
- **MAE (Mean Absolute Error)**: rata-rata selisih absolut antara prediksi dan nilai sebenarnya.
- **R^2 Score**: mengukur seberapa baik prediksi menjelaskan variasi data.

Sebagai pembanding, juga dihitung `naive prediction`, yaitu prediksi berdasarkan harga terakhir dari sekuens input, tanpa menggunakan model.

5. **Prediksi Hari ke Depan**

Code

```
last_date = data['timestamp'].max()
delta = target_datetime.date() - last_date.date()
horizon_days = delta.days
if horizon_days <= 0:
    print("Tanggal target harus lebih dari tanggal terakhir pada data:",
    ↪ last_date)
    return

last_seq = scaled_prices[-look_back:].reshape(1, look_back, 1)
future_preds_scaled = []

for _ in range(horizon_days):
    pred_scaled = model.predict(last_seq)[0, 0]
    future_preds_scaled.append(pred_scaled)
    last_seq = np.append(last_seq[:, 1:, :], [[[pred_scaled]]], axis=1)
```

```
15
16 future_preds =
       ↪ scaler.inverse_transform(np.array(future_preds_scaled).reshape(-1,
       ↪ 1))
```

Bagian ini digunakan untuk melakukan prediksi ke masa depan berdasarkan tanggal target yang diberikan user.

Sistem melakukan prediksi satu per satu (rekursif) untuk tiap hari ke depan. Output prediksi hari ini akan digunakan sebagai input untuk prediksi hari berikutnya, dan seterusnya hingga mencapai target waktu.

Hasil prediksi kemudian dikembalikan ke skala aslinya menggunakan `inverse transform`.

6. **Visualisasi dan Tampilan Hasil**

Code

```
1 future_dates = [last_date + timedelta(days=i + 1) for i in
      ↪ range(horizon_days)]
2 future_dates = [dt.replace(hour=target_datetime.hour,
      ↪ minute=target_datetime.minute, second=0) for dt in future_dates]
3
4 plt.figure(figsize=(12, 6))
5 plt.plot(data['timestamp'], prices, label='Historical Prices')
6 plt.plot(future_dates, future_preds, label=f'Prediksi Hingga
      ↪ {target_datetime}', linestyle='--')
7 plt.legend()
8 plt.title(f'{currency_pair} - Prediksi Sampai
      ↪ {target_datetime.strftime("%d %B %Y %H:%M")}')
9 plt.xlabel('Tanggal')
10 plt.ylabel('Harga')
11 plt.show()
```

Grafik ditampilkan untuk menunjukkan perbandingan antara harga historis dengan prediksi harga masa depan. Garis putus-putus menandai awal prediksi dari waktu saat ini hingga tanggal target.

7. **Input dari Pengguna**

Code

```
print(df.groupby("currency_pair")["timestamp"].max())
print("Currency pair yang tersedia:", df['currency_pair'].unique())
pair = input("Masukkan currency pair (contoh: USDJPY): ").upper()
if not pair.endswith("=X"):
    pair += "=X"

try:
    tgl_input = input("Masukkan tanggal target prediksi (format: dd mm
    ↪ yyyy): ")
    jam_input = input("Masukkan jam target (format HH:MM, contoh 14:15): ")

    dd, mm, yyyy = map(int, tgl_input.strip().split())
    hh, minit = map(int, jam_input.strip().split(":"))

    target_datetime = datetime(yyyy, mm, dd, hh, minit)
    predict_currency(pair, target_datetime)
except ValueError:
    print("Format tanggal atau jam salah. Contoh: tanggal '11 06 2025',
    ↪ jam '14:15'")
```

Program menampilkan pilihan pasangan mata uang yang tersedia berdasarkan data. Setelah user memasukkan pasangan dan waktu target, program akan memanggil fungsi `predict_currency` untuk memproses prediksi harga hingga waktu tersebut.

4.6 Listing program lengkap implementasi model LSTM prediksi

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.callbacks import EarlyStopping
from datetime import timedelta, datetime

# Load data
df = pd.read_csv('test2\currency_rates.csv', parse_dates=['timestamp'])

def predict_currency(currency_pair='USDJPY', target_datetime=None, look_back=60):
    data = df[df['currency_pair'] == currency_pair].copy()
```

```
    data = data.sort_values('timestamp')
    prices = data[['price']].values

    scaler = MinMaxScaler()
    scaled_prices = scaler.fit_transform(prices)

    # Buat dataset X, y
    X, y = [], []
    for i in range(look_back, len(scaled_prices) - 1):
        X.append(scaled_prices[i - look_back:i, 0])
        y.append(scaled_prices[i + 1, 0])
    X, y = np.array(X), np.array(y)
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # Split data
    split = int(len(X) * 0.8)
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]

    # Model
    model = Sequential([
        LSTM(50, return_sequences=True, input_shape=(X.shape[1], 1)),
        LSTM(50),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    early_stop = EarlyStopping(monitor='val_loss', patience=5,
    ↪ restore_best_weights=True)

    history = model.fit(
        X_train, y_train,
        epochs=50,
        batch_size=32,
        validation_split=0.1,
        callbacks=[early_stop],
        shuffle=False,
        verbose=1
    )

    print("Training selesai.")
    plt.figure(figsize=(10, 4))
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Learning Curve')
    plt.xlabel('Epoch')
    plt.ylabel('Loss (MSE)')
    plt.legend()
    plt.show()
```

```
63
64      # Evaluasi test set
65      y_pred_scaled = model.predict(X_test)
66      y_pred = scaler.inverse_transform(y_pred_scaled.reshape(-1, 1))
67      y_true = scaler.inverse_transform(y_test.reshape(-1, 1))
68
69      mse = mean_squared_error(y_true, y_pred)
70      mae = mean_absolute_error(y_true, y_pred)
71      r2 = r2_score(y_true, y_pred)
72
73      naive_pred = scaler.inverse_transform(X_test[:, -1, :])
74      naive_mse = mean_squared_error(y_true, naive_pred)
75      naive_r2 = r2_score(y_true, naive_pred)
76
77      print(f"Model MSE = {mse:.4f}, MAE = {mae:.4f}, R2 = {r2:.4f}")
78      # print(f"Naive baseline MSE = {naive_mse:.4f}, R2 = {naive_r2:.4f}")
79
80      plt.figure(figsize=(12, 6))
81      plt.plot(data['timestamp'].iloc[split + look_back + 1:], y_true,
        ↪ label='Actual Price (Test)')
82      plt.plot(data['timestamp'].iloc[split + look_back + 1:], y_pred,
        ↪ label='Predicted Price (Model)')
83      plt.plot(data['timestamp'].iloc[split + look_back + 1:], naive_pred,
        ↪ label='Naive Forecast')
84      plt.legend()
85      plt.title(f'{currency_pair} - Evaluasi Model')
86      plt.xlabel('Tanggal')
87      plt.ylabel('Harga')
88      plt.show()
89
90      # Hitung jumlah hari yang perlu diprediksi
91      last_date = data['timestamp'].max()
92      delta = target_datetime.date() - last_date.date()
93      horizon_days = delta.days
94      if horizon_days <= 0:
95          print("Tanggal target harus lebih dari tanggal terakhir pada data:",
        ↪ last_date)
96          return
97
98      # Recursive forecast
99      last_seq = scaled_prices[-look_back:].reshape(1, look_back, 1)
100     future_preds_scaled = []
101
102     for _ in range(horizon_days):
103         pred_scaled = model.predict(last_seq)[0, 0]
104         future_preds_scaled.append(pred_scaled)
105         last_seq = np.append(last_seq[:, 1:, :], [[[pred_scaled]]], axis=1)
106
```

```
    future_preds =
    ↪ scaler.inverse_transform(np.array(future_preds_scaled).reshape(-1, 1))

    # Buat future timestamp
    future_dates = [last_date + timedelta(days=i + 1) for i in
    ↪ range(horizon_days)]
    future_dates = [dt.replace(hour=target_datetime.hour,
    ↪ minute=target_datetime.minute, second=0) for dt in future_dates]

    plt.figure(figsize=(12, 6))
    plt.plot(data['timestamp'], prices, label='Historical Prices')
    plt.plot(future_dates, future_preds, label=f'Prediksi Hingga
    ↪ {target_datetime}', linestyle='--')
    plt.legend()
    plt.title(f'{currency_pair} - Prediksi Sampai {target_datetime.strftime("%d
    ↪ %B %Y %H:%M")}')
    plt.xlabel('Tanggal')
    plt.ylabel('Harga')
    plt.show()

    print(f"Prediksi harga {currency_pair} pada {target_datetime} adalah:
    ↪ {future_preds[-1][0]:.3f}")

# ==================== USER INPUT ====================
print(df.groupby("currency_pair")["timestamp"].max())

print("Currency pair yang tersedia:", df['currency_pair'].unique())
pair = input("Masukkan currency pair (contoh: USDJPY): ").upper()
if not pair.endswith("=X"):
    pair += "=X"

try:
    tgl_input = input("Masukkan tanggal target prediksi (format: dd mm yyyy): ")
    jam_input = input("Masukkan jam target (format HH:MM, contoh 14:15): ")

    dd, mm, yyyy = map(int, tgl_input.strip().split())
    hh, minit = map(int, jam_input.strip().split(":"))

    target_datetime = datetime(yyyy, mm, dd, hh, minit)

    predict_currency(pair, target_datetime)

except ValueError:
    print("Format tanggal atau jam salah. Contoh: tanggal '11 06 2025', jam
    ↪ '14:15'")
```