Table of Contents

Sl no. Chapter Name	Page No.
Abstract	i
Acknowledgment	ii
Table of Contents	iii
List of Figures	v
List of tables	vi
Abbreviations	vii
1 Introduction	01
1.1 Organization / Industry	01
1.1.1 Company Profile	01
1.1.1 Domain / Technology	01
1.2 Problem Statement	01
1.2.1 Existing Systems and their Limitations	01
1.2.2 Proposed Systems	02
1.2.3 Problem Formulation	02
2 Requirement Analysis, Tools and Technologies	03
2.1 Hardware and Software Requirements	03
2.2 Functional Requirements	03
2.3 Tools/ Languages/ Platform	03
3 DESIGN AND IMPLEMENTATION	05
3.1 System Architecture	05
3.2 Functional Modules	05
3.3 Algorithm	06

3.4 Libraries / API 07

3.5 Configuration 09

3.2.1 Scraping 09

3.2.2 Cleaning 09

3.2.3 Storage 10

3.2.4 Model 10

4 Observations and Results 11

4.1 Testing 11

4.2 Results 12

4.3 Snapshots 14

5 Conclusion and Future Work 16

5.1 Conclusion 16

5.2 Future Work 16

References 17

List of Figures

Sl no.	Figure	Page no.
Figure 3.1	System architecture	05
Figure 3.2	Flow Diagram	05
Figure 3.3	Collect constituents’ details	08
Figure 3.4	Collect stock data	09
Figure 3.5	Clean and Pre-processing	09
Figure 3.6	Storage utility	10
Figure 3.7	Linear regression model	10
Figure 4.1	Yearly data trend	13
Figure 4.2	Daily data trend	13
Figure 4.3	Historical data Bar chart	14
Figure 4.4	Sending keys to search box	14
Figure 4.5	Collecting tabular data	15
Figure 4.6	Collecting multiple tables	15

List of Tables

Sl no.	Table	Page no.
Table 4.1	Collection of all constituents	11
Table 4.2	Constituent data collection	12
Table 4.3	Insert data to sqlite3	12

Abbreviations

Abbreviation	**Full Form**
API	Application Program Interface
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
NPM	Node Package Manager
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION/ INDUSTRY

1.1.1 Company Profile

Pecten is a Fintech Start-up that has created a suite of products and services that leverages vast amounts and a wide variety of relevant alternative datasets and effectively recalibrates and combines them with technical and fundamentals datasets using proprietary modes based on Artificial Intelligence and Machine Learning

This helps us to extract greater and concise insights into stocks and companies of your interest, enabling you to make timely informed investment decisions.

Pecten provides an AI-driven dynamic investment approach to achieve wealth growth and long-term value creation.

1.1.2 Domain/ Technology

Equity investing space has 50,000 stocks globally with over 10,000 different types of funds are fishing in the same pond. Pecten's objective is to answer investor's three questions through supervised and unsupervised Machine Learning algorithms which learn from historic data, by identifying patterns, trends and apply it in real-time.

Pecten is not only an uninspired labor-replacement or cost-saving product but also more expansive, thanks to the use of AI as a catalyst for new ways to create high performing funds and create better investment strategies.

Pecten's machine learning engine enables the ability to keep improving its performance without human intervention in highly volatile and dynamic market conditions.

1.2 Problem Statement

1.2.1 Existing Systems and their limitations

The current systems for financial analysis are all black boxes which provide no insight into the trends of the stocks. With deeper knowledge the consumers can involve themselves more diligently in the market transactions. The systems provide a paid platform where the consumers of these products make transactions blindly following the suggestions given by them. This abstracts all the knowledge required by the user which makes the users not to trust the platforms.

1.2.2 Proposed Solution

- Enable business prototyping through technically advanced self-service tool
- Ingest more datasets to derive better insights
- Exploit social media for more insights
- Collaborative view/ report
- Rapid analytics: Quick and dynamic model, test, evaluate, learn, adjust & repeat

1.2.3 Problem formulation

To automate the process of stock data collection from web, pre-process it, show visualization of variance of data and make predictions using machine learning.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 Hardware & Software Requirements

- **Software Requirements**
  1. *Selenium*
  2. *ChromeDriver*
  3. *Operating system: Windows 10/7/8 or Ubuntu 18.02 or higher*
  4. *sqlite3*
  5. *Pandas Library*
  6. *sklearn Library*
  7. *Chart.js*
- **Hardware Requirements**
  1. *Intel i5 or higher*
  2. *Intel 1.8Hz or above 64-bit processor*
  3. *8 GB RAM*
  4. *Standard Display*

2.2 Functional Requirements

1. The platform should be reliable.
2. It should collect, process and store sufficient information to predict stock trends.
3. It should work in real-time scenario and hence it must be fast.

2.3 Tools/ Languages/ Platform

- **Python**

  Python 3.0 broke backward compatibility, and much Python 2 code does not run unmodified on Python 3. Python's dynamic typing combined with the plans to change the semantics of certain methods of dictionaries, for example, made perfect mechanical translation from Python 2.x to Python 3.0 very difficult. A tool called "2to3" does the parts of translation that can be done automatically. At this, 2to3 appeared to be fairly successful, though an early review noted that there were aspects of translation that such a tool would never be able to handle. Prior to the roll-out of Python 3, projects requiring compatibility with both the 2.x and 3.x series were recommended to have one source (for the 2.x series), and produce releases for the Python 3.x platform using 2to3. Edits to the Python 3.x code were discouraged for so long as the code needed to run on Python 2.x. This is no longer recommended as of 2012 the preferred approach is to create a single code base that can run under both Python 2 and 3 using compatibility modules.

- **Selenium**

  Selenium is a web browser automation tool. Primarily, it is for automating web applications for testing purposes, but it is certainly not limited just to that. It allows you to open a browser of your choice and perform tasks as a human being would, such as:

  - Clicking buttons
  - Entering information in forms
  - Searching for specific information on the web pages

CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 System Architecture

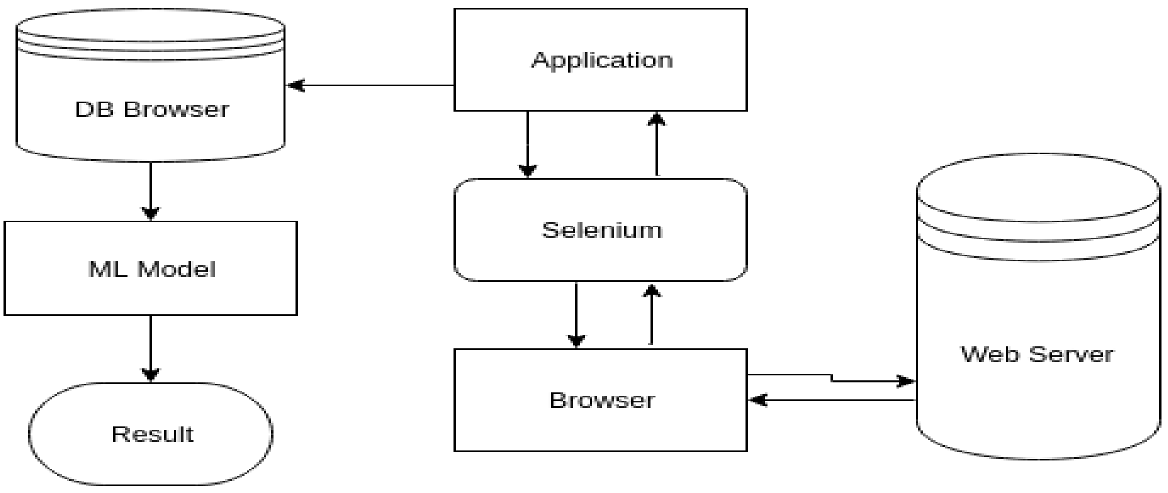


Figure 3.1 System Architecture

A python script using selenium library is used to interact with browser. Then the necessary data is collected, preprocessed and stored in a sqlite3 database. Using this data, visualizations are prepared using the Chart.js API and presented using the Flask server. ML model is built to predict the trend of certain parameters of the data.

3.2 System Flow Diagram

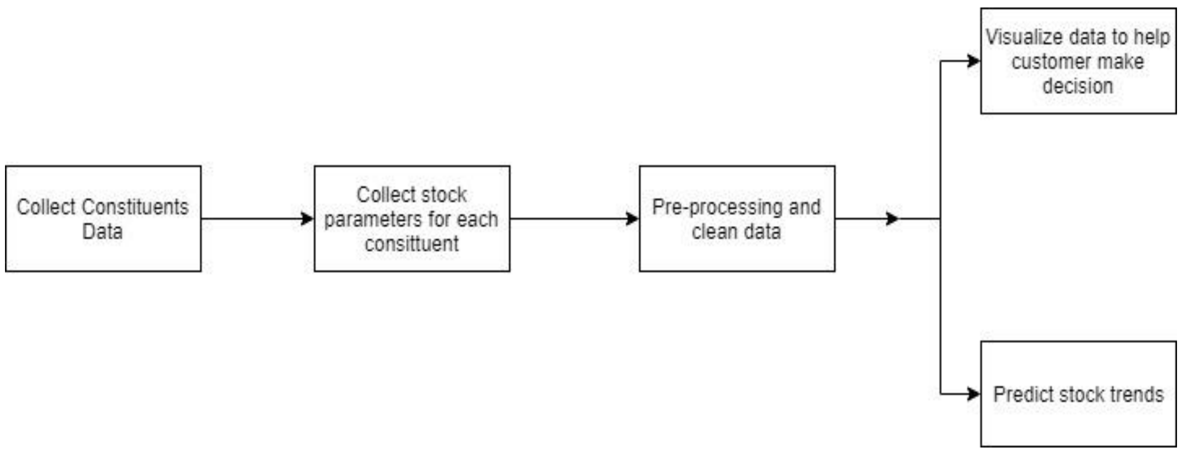

Visualize data to help customer make decision
Collect Constituents Data
Collect stock parameters for each consittuent
Pre-processing and clean data
Predict stock trends

Figure 3.2 System flow diagram

System modules involved in the approach are: -

- ***Constituents data collection***: The data to about company and their unique ID's is collected and stored in the database
- ***Stock data collection***: Then, for each constituent stock data is collected
- ***Pre-processing collected data***: Then, preprocessing is done on collected data.
- ***Predict / Visualize stock trend***: Then, simple linear regression model is developed to predict and Chart,js is used to create visualizations of it.

3.3 Algorithm

A linear model for fitting the data and predicting has been used. Linear regression is one such linear model. Linear regression is an approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

In statistics, originally least squares is a type of linear least squares method for estimating the unknown parameters in a linear regression model.

ALGORITHM

```
Step 1: Take X and Y as input

Step 2: compute the slope and bias using the formula,

          m = $\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sum_{j=1}^{n}(X_j-\bar{X})^2}$

          b = $\bar{Y} - m\bar{X}$

     where $m$ - the slope of hyperplane and b - bias

Step 3: Make predictions using,

        $Y = mX + b$
```

3.4 Libraries/ API's

- **Chart.js**

  It is a simple yet flexible JavaScript charting for designers and developers. The framework can be used in many applications.

This framework has been used because it is:

- Open Source
- HTML5 canvas
- Responsive

The advantages of this framework are:

- ***Mixed chart types*** - Mix and match bar and line charts to provide a clear visual distinction between datasets.
- ***New chart axis types*** - Plot complex, sparse datasets on date time, logarithmic or even entirely custom scales with ease.
- ***Animate everything*** - Out of the box stunning transitions when changing data, updating colors and adding datasets.

- **Pandas**

  Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Library features are:

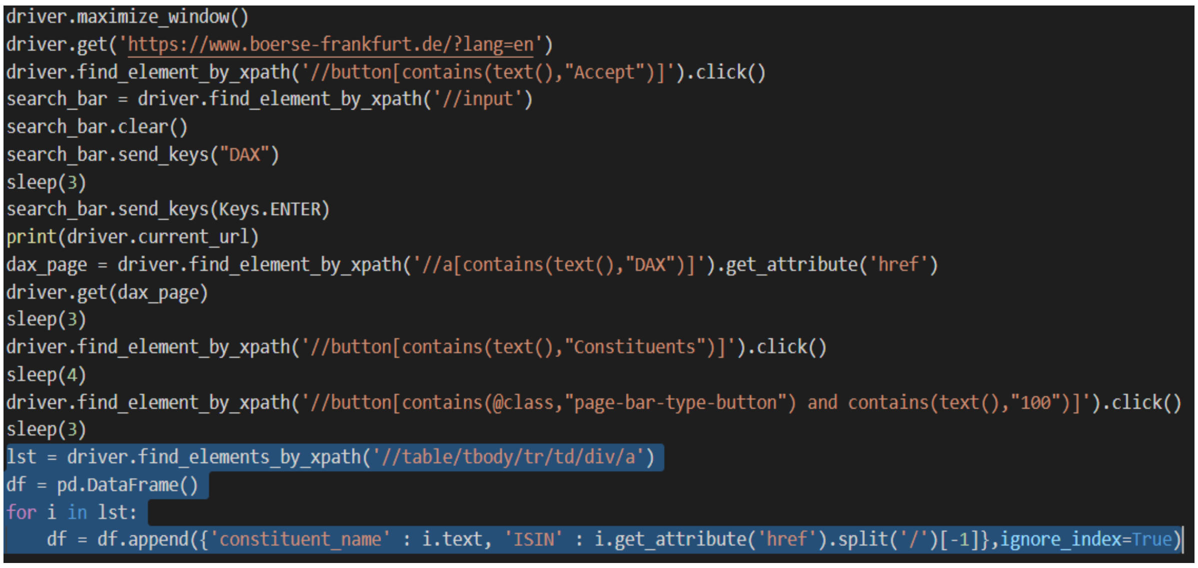
- DataFrame object for data manipulation with integrated indexing.
- Tools for reading writing data between in-memory data structures different file formats. Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets. Label-based slicing, fancy indexing, and getting subsets of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional structure.

- **Scikit-learn**

  It is a free software machine learning library for the Python programming language. The framework is built on top of several popular Python packages, namely NumPy, SciPy, and matplotlib. Scikit-Learn is characterized by a clean, uniform, and streamlined API, as well as by very useful and complete online documentation. A benefit of this uniformity is that once you understand the basic use and syntax of Scikit-Learn for one type of model, switching to a new model or algorithm is very straightforward.

3.5 Configuration

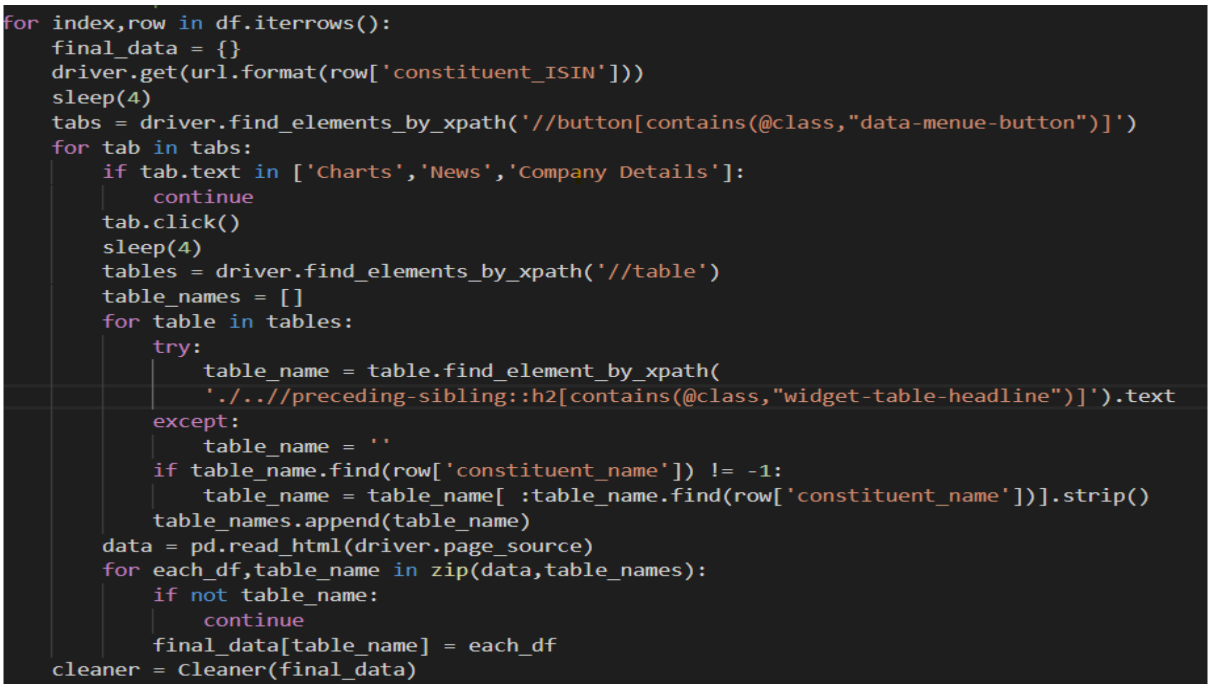
3.5.1 Scraping



```
driver.maximize_window()
driver.get('https://www.boerse-frankfurt.de/?lang=en')
driver.find_element_by_xpath('//button[contains(text(),"Accept")]').click()
search_bar = driver.find_element_by_xpath('//input')
search_bar.clear()
search_bar.send_keys("DAX")
sleep(3)
search_bar.send_keys(Keys.ENTER)
print(driver.current_url)
dax_page = driver.find_element_by_xpath('//a[contains(text(),"DAX")]').get_attribute('href')
driver.get(dax_page)
sleep(3)
driver.find_element_by_xpath('//button[contains(text(),"Constituents")]').click()
sleep(4)
driver.find_element_by_xpath('//button[contains(@class,"page-bar-type-button") and contains(text(),"100")]').click()
sleep(3)
lst = driver.find_elements_by_xpath('//table/tbody/tr/td/div/a')
df = pd.DataFrame()
for i in lst:
    df = df.append({'constituent_name' : i.text, 'ISIN' : i.get_attribute('href').split('/')[-1]},ignore_index=True)
```

Figure 3.3 Collect all the constituents' name and WKN from website

The Figure 3.3 the code for collecting the constituents and its details provided by the website is collected. This data is stored in a sqlite3 database.

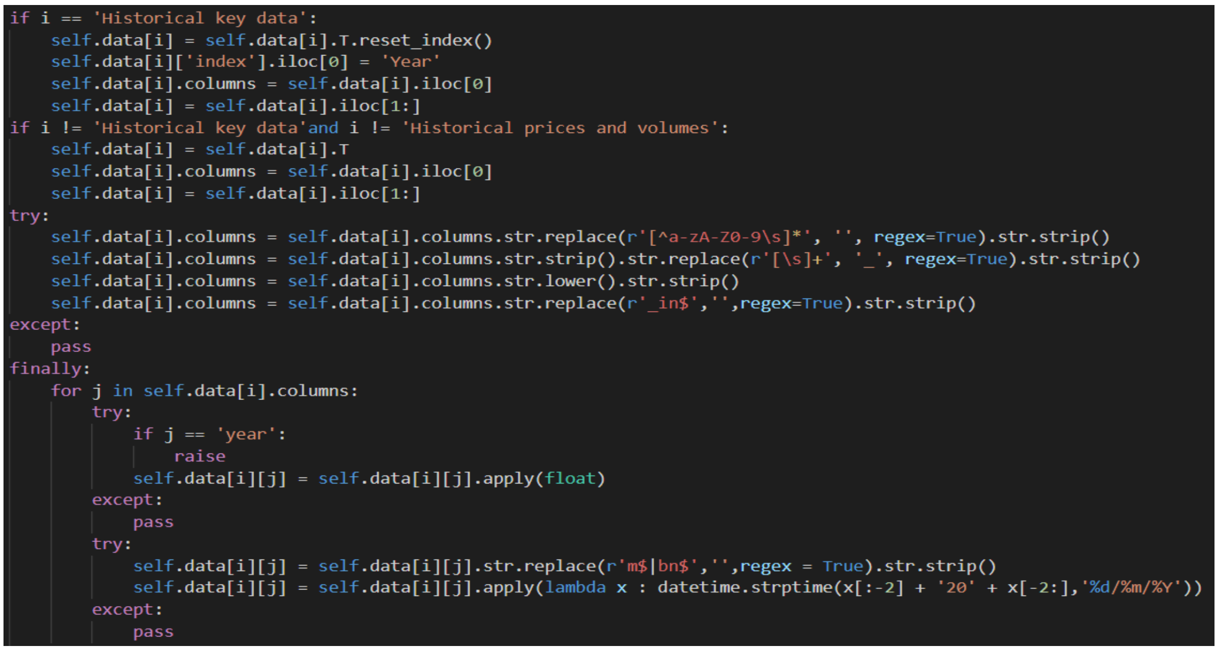


```python
for index,row in df.iterrows():
    final_data = {}
    driver.get(url.format(row['constituent_ISIN']))
    sleep(4)
    tabs = driver.find_elements_by_xpath('//button[contains(@class,"data-menue-button")]')
    for tab in tabs:
        if tab.text in ['Charts','News','Company Details']:
            continue
        tab.click()
        sleep(4)
        tables = driver.find_elements_by_xpath('//table')
        table_names = []
        for table in tables:
            try:
                table_name = table.find_element_by_xpath(
                './../..//preceding-sibling::h2[contains(@class,"widget-table-headline")]').text
            except:
                table_name = ''
            if table_name.find(row['constituent_name']) != -1:
                table_name = table_name[ :table_name.find(row['constituent_name'])].strip()
            table_names.append(table_name)
        data = pd.read_html(driver.page_source)
        for each_df,table_name in zip(data,table_names):
            if not table_name:
                continue
            final_data[table_name] = each_df
    cleaner = Cleaner(final_data)
```

Figure 3.4 Collect stock related data for each constituent

The Figure 3.4 shows code snippet which collects stock related data from each constituent.

3.5.2 Cleaning

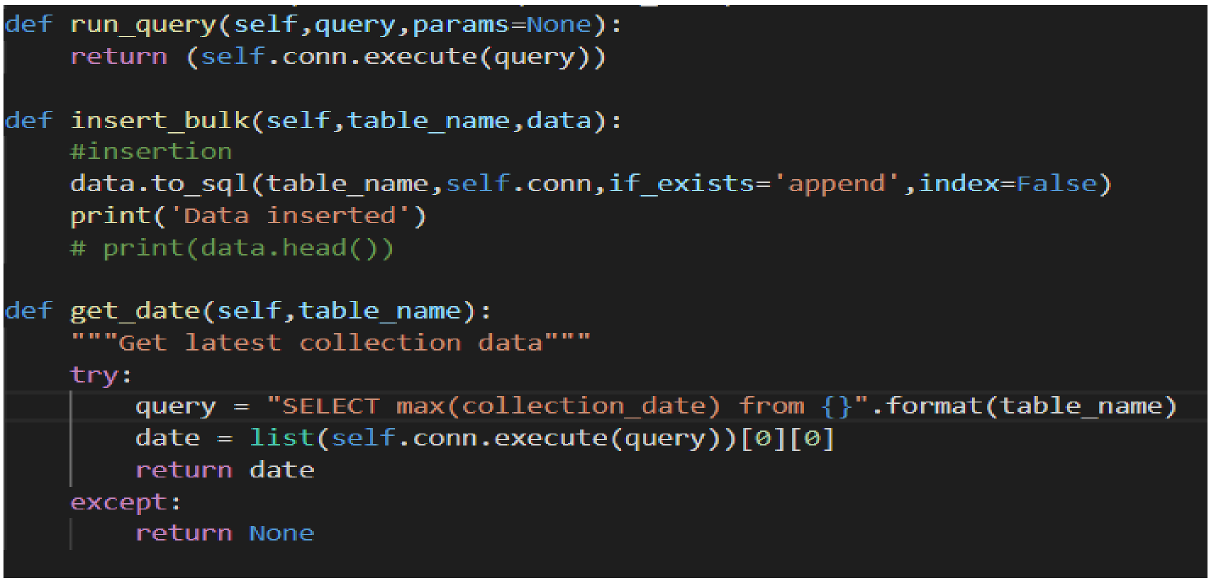


```python
if i == 'Historical key data':
    self.data[i] = self.data[i].T.reset_index()
    self.data[i]['index'].iloc[0] = 'Year'
    self.data[i].columns = self.data[i].iloc[0]
    self.data[i] = self.data[i].iloc[1:]
if i != 'Historical key data'and i != 'Historical prices and volumes':
    self.data[i] = self.data[i].T
    self.data[i].columns = self.data[i].iloc[0]
    self.data[i] = self.data[i].iloc[1:]
try:
    self.data[i].columns = self.data[i].columns.str.replace(r'[^a-zA-Z0-9\s]*', '', regex=True).str.strip()
    self.data[i].columns = self.data[i].columns.str.strip().str.replace(r'[\s]+', '_', regex=True).str.strip()
    self.data[i].columns = self.data[i].columns.str.lower().str.strip()
    self.data[i].columns = self.data[i].columns.str.replace(r'_in$','',regex=True).str.strip()
except:
    pass
finally:
    for j in self.data[i].columns:
        try:
            if j == 'year':
                raise
            self.data[i][j] = self.data[i][j].apply(float)
        except:
            pass
        try:
            self.data[i][j] = self.data[i][j].str.replace(r'm$|bn$','',regex = True).str.strip()
            self.data[i][j] = self.data[i][j].apply(lambda x : datetime.strptime(x[:-2] + '20' + x[-2:],'%d/%m/%Y'))
        except:
            pass
```

Figure 3.5 Clean and pre-processing tabular data collected by collection script

Figure 5.5 shows code snippet to preprocess the tabular stock data collected. The data collected will not conform with the types in database, it may contain unwanted text characters which will is noise according to the model and hence these discrepancies will be removed.

3.5.3 Storing

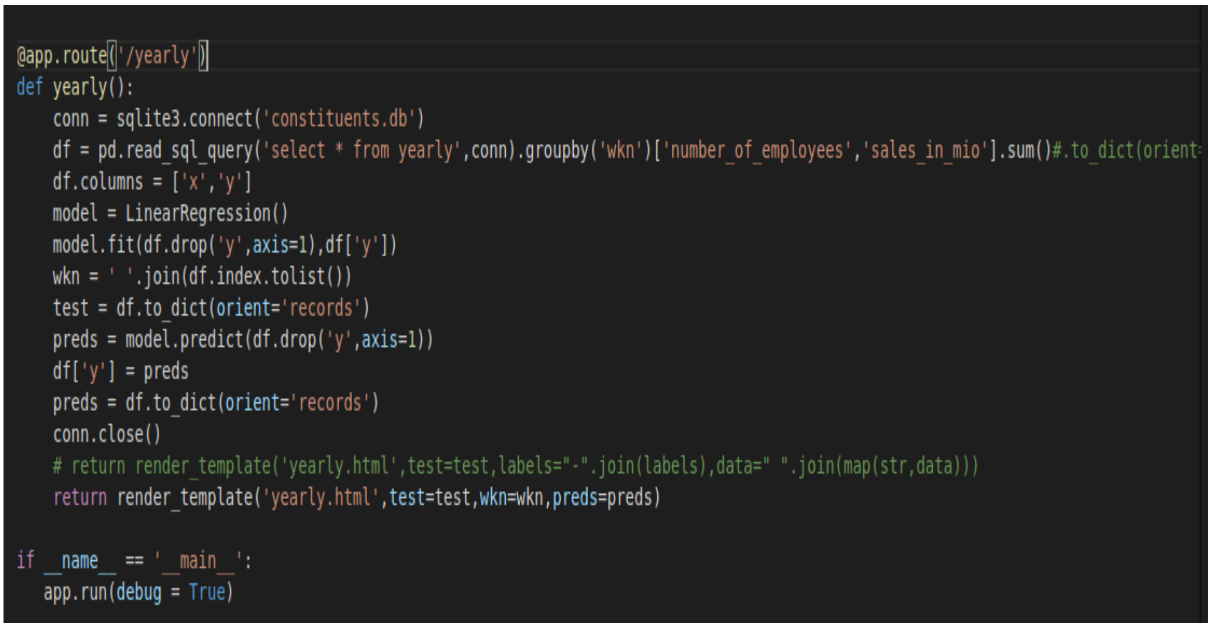


```python
def run_query(self,query,params=None):
    return (self.conn.execute(query))

def insert_bulk(self,table_name,data):
    #insertion
    data.to_sql(table_name,self.conn,if_exists='append',index=False)
    print('Data inserted')
    # print(data.head())

def get_date(self,table_name):
    """Get latest collection data"""
    try:
        query = "SELECT max(collection_date) from {}".format(table_name)
        date = list(self.conn.execute(query))[0][0]
        return date
    except:
        return None
```

Figure 3.6 Storage utility which includes database function

The storage utility handles all data manipulation process on the database end. The code in Figure 3.6 shows the functions used for this manipulation which includes process like running a query to retrieve data and inserting data

3.5.4 Model



```python
@app.route('/yearly')
def yearly():
    conn = sqlite3.connect('constituents.db')
    df = pd.read_sql_query('select * from yearly',conn).groupby('wkn')['number_of_employees','sales_in_mio'].sum()#.to_dict(orient=
    df.columns = ['x','y']
    model = LinearRegression()
    model.fit(df.drop('y',axis=1),df['y'])
    wkn = ' '.join(df.index.tolist())
    test = df.to_dict(orient='records')
    preds = model.predict(df.drop('y',axis=1))
    df['y'] = preds
    preds = df.to_dict(orient='records')
    conn.close()
    # return render_template('yearly.html',test=test,labels="-".join(labels),data=" ".join(map(str,data)))
    return render_template('yearly.html',test=test,wkn=wkn,preds=preds)

if __name__ == '__main__':
   app.run(debug = True)
```

Figure 3.7 Linear regression model

Figure 3.7 shows the code which predicts the stock trends using linear regression.

CHAPTER 4

OBSERVATION AND RESULT

4.1 Testing

Testing Software

Testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. Testing is the process of technical investigation and includes the process of executing a program or application with the intent of finding errors.

Unit Testing

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. Unit testing has been done to check whether the cancer dataset is loaded correctly.

Table 4.1: Collection of all constituents Test Case

Sl No. of test case:	01
Name of test:	Collection test
Item / Feature being tested:	Scraping
Sample Input:	Constituent name
Expected output:	Collect data about the constituent
Observed output:	Collect data about the constituent
Remarks	Test succeeded

The table 4.1 shows unit test case for the collection script. If data has been successfully collected it inserts the same to the database.

Table 4.2: Constituent data collection Test Case

Sl No. of test case:	02
Name of test:	Collection test
Item / Feature being tested:	Scraping
Sample Input:	Constituent name
Expected output:	Data not found
Observed output:	Data not found
Remarks	Test Succeeded

The table 4.2 shows unit test case for the system when it does not find the constituent it is looking for.

Table 4.3: Insert data to sqlite3 Test Case

Sl No. of test case:	03
Name of test:	Data insertion test
Item / Feature being tested:	SQL query
Sample Input:	None
Expected output:	Data inserted successfully
Observed output:	Data inserted successfully
Remarks	Test succeeded

The table 4.3 shows unit test case for insertion of data into database after the collection scripts are done collecting the data.

4.2 Results

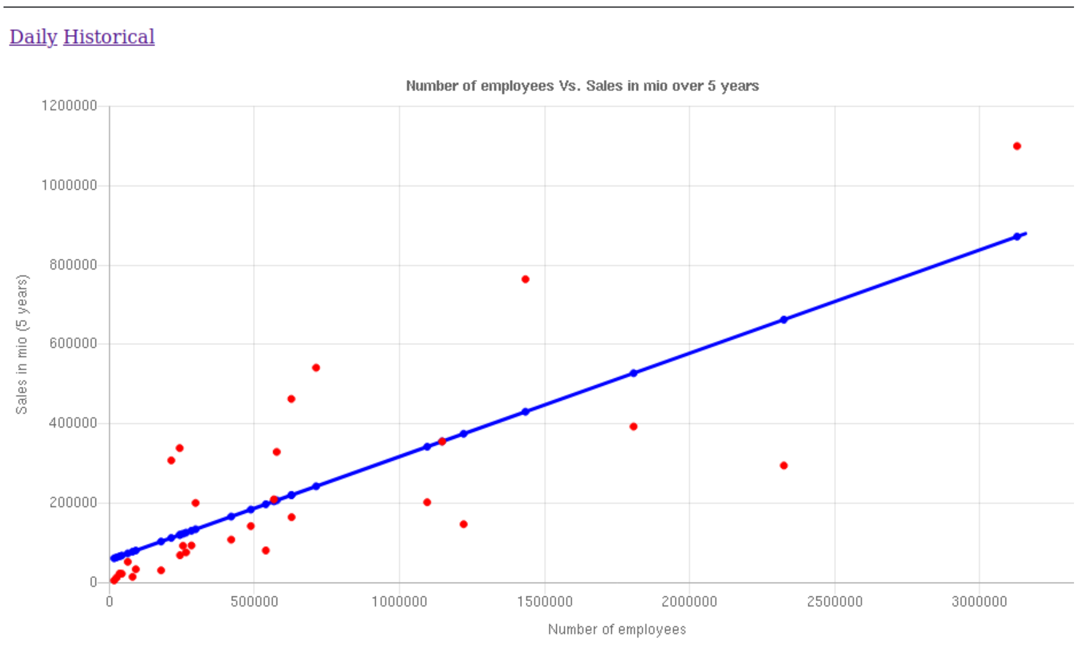


Figure 4.1 Yearly data trend

The figure 4.1 shows the linear model fitted for number of employees vs sales in million.

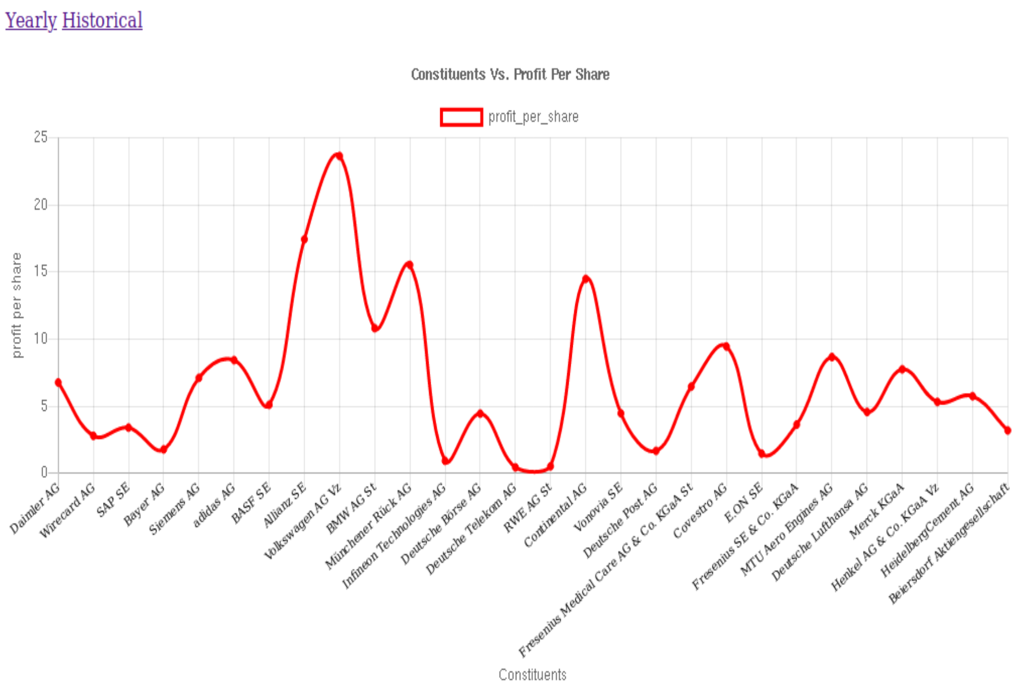


Figure 4.2 Daily data trend

Figure 4.2 shows profit per share for each constituent to help make decisions on which stock to buy or sell.

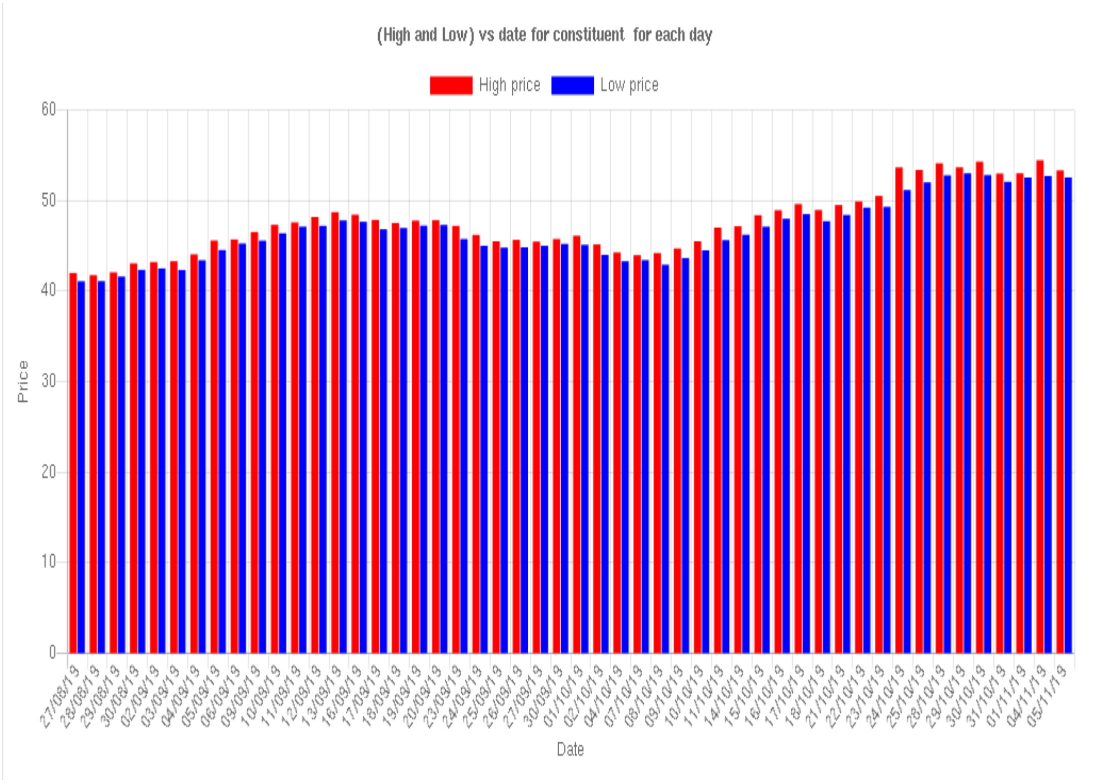
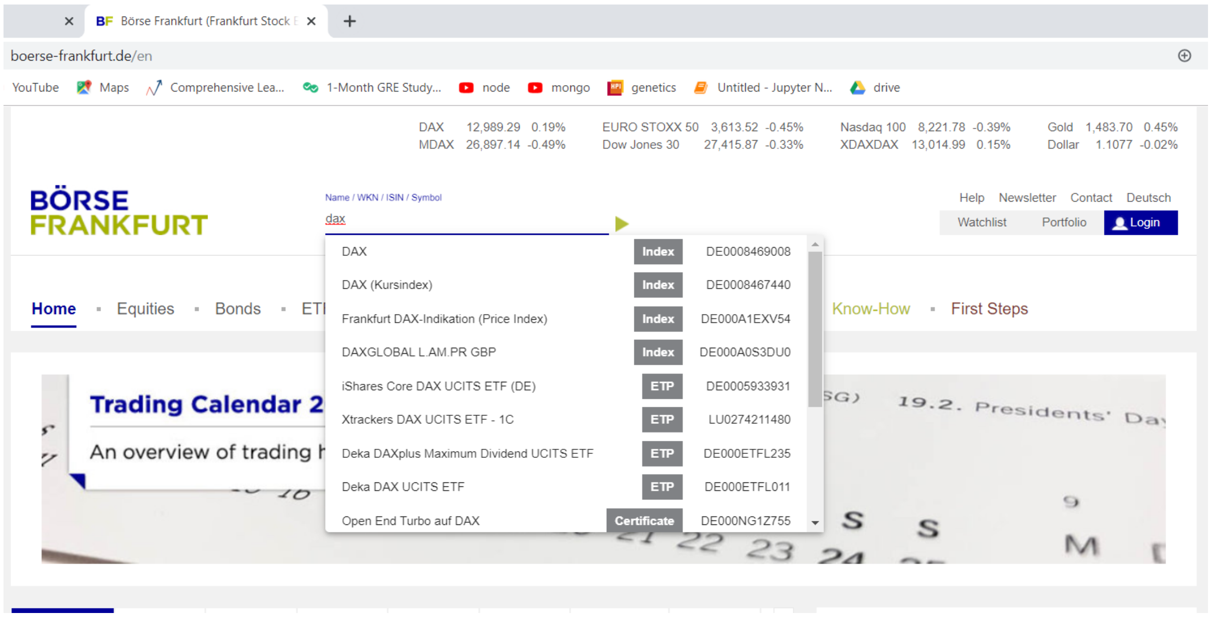


Figure 4.3 Historical data Bar chart

Figure 4.3 shows the high and low prices of the stock of a constituent on each day for the past month.

4.3 Snapshots


BF Börse Frankfurt (Frankfurt Stock E

boerse-frankfurt.de/en

YouTube Maps Comprehensive Lea... 1-Month GRE Study... node mongo genetics Untitled - Jupyter N... drive

DAX	12,989.29	0.19%	EURO STOXX 50	3,613.52	-0.45%	Nasdaq 100	8,221.78	-0.39%	Gold	1,483.70	0.45%
MDAX	26,897.14	-0.49%	Dow Jones 30	27,415.87	-0.33%	XDAXDAX	13,014.99	0.15%	Dollar	1.1077	-0.02%

BÖRSE FRANKFURT

Name / WKN / ISIN / Symbol

dax

Help Newsletter Contact Deutsch

Watchlist Portfolio Login

DAX	Index	DE0008469008
DAX (Kursindex)	Index	DE0008467440
Frankfurt DAX-Indikation (Price Index)	Index	DE000A1EXV54
DAXGLOBAL L.AM.PR GBP	Index	DE000A0S3DU0
iShares Core DAX UCITS ETF (DE)	ETP	DE0005933931
Xtrackers DAX UCITS ETF - 1C	ETP	LU0274211480
Deka DAXplus Maximum Dividend UCITS ETF	ETP	DE000ETFL235
Deka DAX UCITS ETF	ETP	DE000ETFL011
Open End Turbo auf DAX	Certificate	DE000NG1Z755

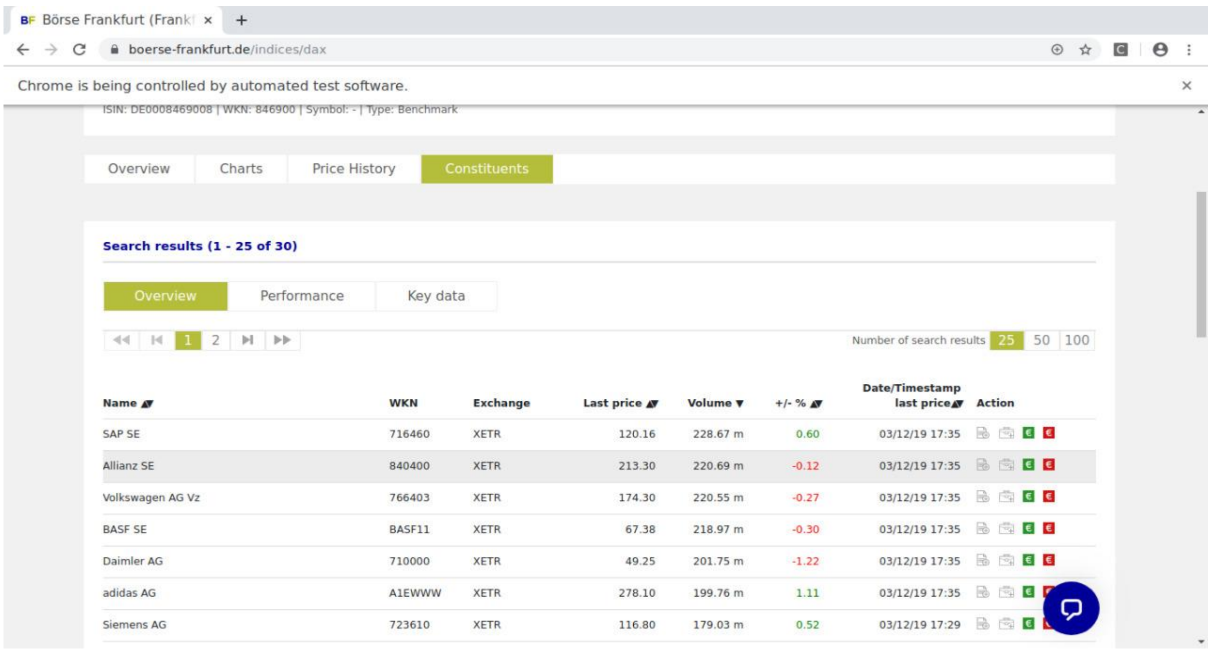
Home Equities Bonds ETI Know-How First Steps

Trading Calendar 2

An overview of trading h

Figure 4.4 Sending keys to search box

The Figure 4.5 shows the selenium tool being used to enter keys into a search box.



Chrome is being controlled by automated test software.

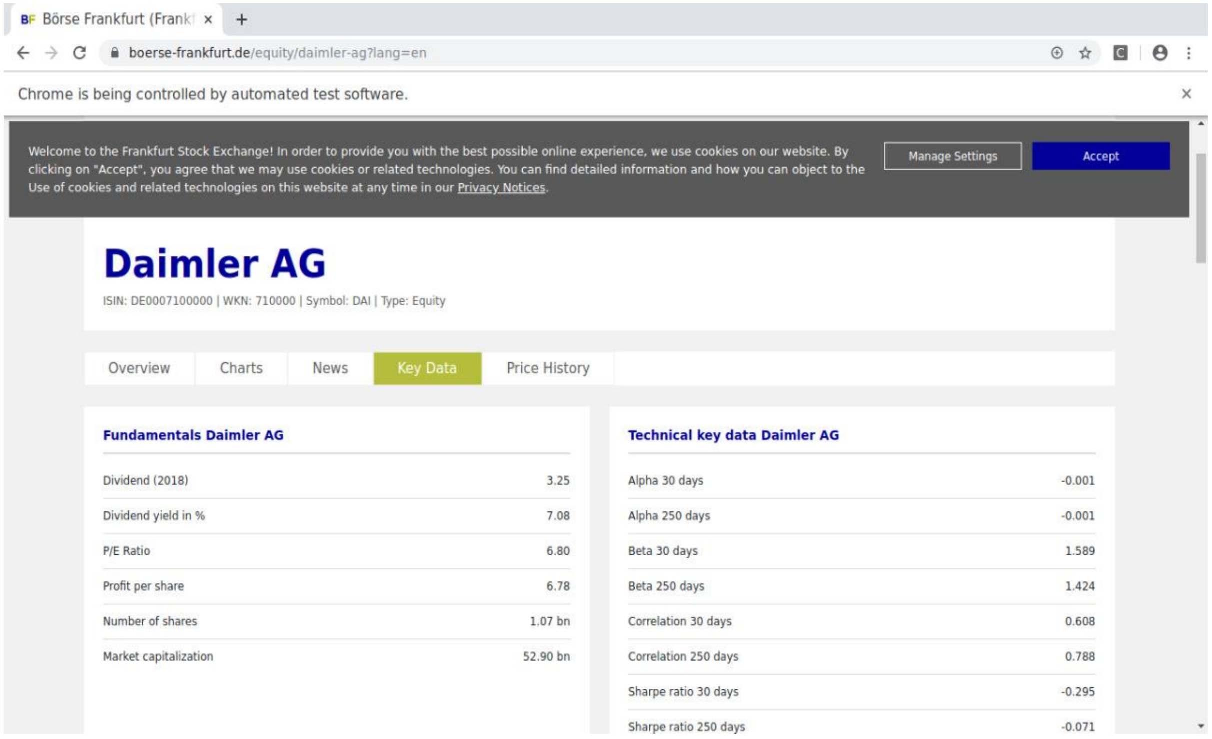
ISIN: DE0008469008 | WKN: 846900 | Symbol: - | Type: Benchmark

Search results (1 - 25 of 30)

Name	WKN	Exchange	Last price	Volume	+/- %	Date/Timestamp last price	Action
SAP SE	716460	XETR	120.16	228.67 m	0.60	03/12/19 17:35	
Allianz SE	840400	XETR	213.30	220.69 m	-0.12	03/12/19 17:35	
Volkswagen AG Vz	766403	XETR	174.30	220.55 m	-0.27	03/12/19 17:35	
BASF SE	BASF11	XETR	67.38	218.97 m	-0.30	03/12/19 17:35	
Daimler AG	710000	XETR	49.25	201.75 m	-1.22	03/12/19 17:35	
adidas AG	A1EWWW	XETR	278.10	199.76 m	1.11	03/12/19 17:35	
Siemens AG	723610	XETR	116.80	179.03 m	0.52	03/12/19 17:29	

Figure 4.5 Collecting tabular data

The Figure 4.5 shows collecting of an entire table using the selenium tool



Chrome is being controlled by automated test software.

Welcome to the Frankfurt Stock Exchange! In order to provide you with the best possible online experience, we use cookies on our website. By clicking on "Accept", you agree that we may use cookies or related technologies. You can find detailed information and how you can object to the Use of cookies and related technologies on this website at any time in our Privacy Notices.

Daimler AG

ISIN: DE0007100000 | WKN: 710000 | Symbol: DAI | Type: Equity

Fundamentals Daimler AG

Dividend (2018)	3.25
Dividend yield in %	7.08
P/E Ratio	6.80
Profit per share	6.78
Number of shares	1.07 bn
Market capitalization	52.90 bn

Technical key data Daimler AG

Alpha 30 days	-0.001
Alpha 250 days	-0.001
Beta 30 days	1.589
Beta 250 days	1.424
Correlation 30 days	0.608
Correlation 250 days	0.788
Sharpe ratio 30 days	-0.295
Sharpe ratio 250 days	-0.071

Figure 4.6 Collecting multiple tables

The Figure 4.6 shows how the selenium tool collects multiple tables of different format and stores in a pandas dataframe.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The financial market is a volatile one. To predict the trends of the market with high accuracy is very difficult. There are many other parameters which influence these trends such as the news, global economy, industry economy, local requirements, etc. The collected information may not be sufficient to predict the trends accurately but in a constant environment it has proved to be good enough to predict with a high accuracy. The visualizations can help a student of this field to find correlations between different parameters of the collected stock data. Based on the correlation factor between parameters models can be built to predict the variations in the parameters. This would help the consumers to decide whether to buy or sell the stocks.

5.2 Limitations and future work

- The application has to be run manually every week. An automated software like Jenkins or airflow can be used to run this weekly automatically.

- Using online storage like GCP or AWS to handle large incoming traffic will help reducing the time spent by accessing and processing data.

- The UI can be improved for easy access and understanding by the users.

- The model used to predict can be improved to provide better accuracy by collecting news information and performing sentiment analysis on this data.

REFERENCES

[1] https://www.w3schools.com/

[2] https://stackoverflow.com/

[3] https://developer.mozilla.org/en-US/

[4] https://selenium-python.readthedocs.io/

[5] https://www.python.org/

[6] https://github.com/pallets/flask

[7] Automate the Boring Stuff with Python by Al Sweigart