data-preprocessing

August 30, 2024

nasdaq historical data

[20]:
```python
import pandas as pd

# Step 1: Upload the CSV file
from google.colab import files
uploaded = files.upload()

# Assuming the file is uploaded successfully, you can access it using the file␣
↪name.
file_name = list(uploaded.keys())[0]  # Get the file name

# Step 2: Read the CSV file
df = pd.read_csv(file_name)

# Step 3: Display the first 10 rows
print(df.head(20))
```

```
<IPython.core.display.HTML object>

Saving HistoricalData_1725024094933.csv to HistoricalData_1725024094933 (4).csv
          Date  Close/Last      Open      High       Low
0   08/29/2024    17516.43  17610.57  17789.21  17482.60
1   08/28/2024    17556.03  17738.80  17759.94  17439.40
2   08/27/2024    17754.82  17655.52  17789.72  17573.37
3   08/26/2024    17725.76  17867.85  17909.09  17645.69
4   08/23/2024    17877.79  17772.73  17941.27  17700.27
5   08/22/2024    17619.35  17993.72  18017.69  17589.15
6   08/21/2024    17918.99  17840.51  17963.07  17790.98
7   08/20/2024    17816.94  17849.09  17932.53  17758.20
8   08/19/2024    17876.77  17649.74  17877.44  17585.58
9   08/16/2024    17631.72  17516.40  17674.65  17502.82
10  08/15/2024    17594.50  17394.55  17602.72  17375.41
11  08/14/2024    17192.60  17227.64  17260.73  17032.17
12  08/13/2024    17187.61  16944.74  17192.79  16943.95
13  08-12-2024    16780.61  16793.64  16895.79  16699.39
14  08-09-2024    16745.30  16636.52  16789.22  16574.57
15  08-08-2024    16660.02  16408.27  16694.25  16262.93
16  08-07-2024    16195.81  16622.31  16709.81  16179.53
```

```
17  08-06-2024    16366.85  16261.36  16620.31  16137.65
18  08-05-2024    16200.08  15712.53  16453.46  15708.54
19  08-02-2024    16776.16  16780.45  16920.63  16582.79
```

[ ]:
```
!pip install pandas
!pip install xlrd
!pip install openpyxl
```

Data preprocessing: Date Cleaning and Formatting: Removed the time component from datetime entries, ensuring that only the date was retained. This step helped standardize the ‘Date’ column to a consistent format.

Handling Multiple Date Formats: Accounted for different date formats within the column, converting them into a uniform format (‘%m/%d/%Y’). This ensures that all dates are consistent, making further analysis easier.

[16]:
```python
import pandas as pd
from datetime import datetime

# Path to the Excel file
file_path = '/content/HistoricalData_1725024094933.csv'
updated_file_path = '/content/HistoricalData_NASDAQ_modified_file.csv'

def convert_date(date):
    # If the date is a datetime object, format it directly
    if isinstance(date, datetime):
        return date.strftime('%m/%d/%Y')
    elif isinstance(date, str):
        try:
            # If the date is a string, try to parse and format it
            return datetime.strptime(date, '%m-%d-%Y').strftime('%m/%d/%Y')
        except ValueError:
            try:
                return datetime.strptime(date, '%m/%d/%Y').strftime('%m/%d/%Y')
            except ValueError:
                return date
    else:
        return date

try:
    # Read the Excel file
    df = pd.read_csv(file_path)

    # Apply the conversion function to the 'Date' column
    df['Date'] = df['Date'].apply(convert_date)

    # Save the updated DataFrame back to an Excel file
    df.to_csv(updated_file_path, index=False)
    print(f"Dates converted and file saved to {updated_file_path}")

except FileNotFoundError:
    print(f"The file at {file_path} was not found.")
except ValueError as e:
    print(f"Error processing the file: {e}")
```

```
Dates converted and file saved to
/content/HistoricalData_NASDAQ_modified_file.csv
```

Stock news preprocessing and Filtering the news related to the Nifty IT 50 Companies

[ ]:
```
!pip install regex
```

```
Requirement already satisfied: regex in /usr/local/lib/python3.10/dist-packages
(2024.5.15)
```

Filtering the news of Nifty it 50 companies This code processes a CSV file containing stock news to identify and count mentions of Nifty IT 50 companies in the news descriptions. It uses:

- **pandas**: For loading, manipulating, and saving CSV data.
- **re (regex)**: For searching company names in news descriptions.
- **collections.Counter**: For counting the occurrences of each company's mentions.

The script creates a new column indicating which companies are mentioned in each news entry and outputs a summary of the total mentions per company.

[21]:
```
import pandas as pd
import re
from collections import Counter

# Example company names list
company_names = [
    ("tata consultancy services", "tcs"),
    ("infosys", "infy"),
    ("wipro technologies", "wipro"),
    ("hcl technologies", "hcl"),
    ("ltimindtree", "ltimindtree"),
    ("tech mahindra", "techm"),
    ("persistent systems", "psys"),
    ("l&t technology services", "ltts"),
    ("mphasis", "mphasis"),
    ("coforge", "coforge")
]

# Load your CSV file
df = pd.read_csv('stock_news.csv')

# Initialize a counter to count occurrences of each company
```

```
company_count = Counter()

# Function to search for company names in description and update counter
def find_companies(description):
    found_companies = []
    for full_name, short_name in company_names:
        if re.search(rf'\b{re.escape(full_name)}\b', description, re.
 ↪IGNORECASE) or \
           re.search(rf'\b{re.escape(short_name)}\b', description, re.
 ↪IGNORECASE):
            found_companies.append(full_name)
            company_count[full_name] += 1
    return ', '.join(found_companies) if found_companies else ''

# Apply the function to the 'Description' column and create a new 'Company'␣
 ↪column
df['Company'] = df['Description'].apply(find_companies)

# Save the updated DataFrame to a new CSV file
df.to_csv('filtered_news.csv', index=False)

# Create a DataFrame from the company_count dictionary for the summary
company_summary_df = pd.DataFrame(company_count.items(), columns=['Company',␣
 ↪'Count'])

# Sort the summary DataFrame by count in descending order
company_summary_df = company_summary_df.sort_values(by='Count', ascending=False)

# Print the summary DataFrame
print(company_summary_df)

# Optionally, save the summary to a CSV file
company_summary_df.to_csv('company_news_count.csv', index=False)
```

```
                     Company  Count
4              tech mahindra    198
2                    infosys    157
0          hcl technologies    156
1        wipro technologies    138
5                ltimindtree    114
3  tata consultancy services     82
6                    mphasis     28
7    l&t technology services      9
```

Creating new csv file that have the news related to Nifty it 50 companies

```
[ ]: # Load your CSV file
     df = pd.read_csv('filtered_news.csv')

     # Remove rows with blank entries in the 'Company' column
     df_filtered = df.dropna(subset=['Company'])

     # Save the updated DataFrame to a new CSV file
     df_filtered.to_csv('nifty_it_50_stock_news.csv', index=False)

     print("Filtered news with blank entries removed has been saved to␣
      ↪'filtered_news_no_blank.csv'.")
```

```
Filtered news with blank entries removed has been saved to
'filtered_news_no_blank.csv'.
```

Import Libraries: The script imports pandas for data manipulation and re for regex operations.

Define Keywords: Two lists of keywords are defined: pos_words_to_search for positive sentiment and neg_words_to_search for negative sentiment. These words are associated with potential stock price increases or decreases.

Compile Regex Patterns: Positive and negative word lists are compiled into regex patterns using re.compile() with word boundaries to ensure accurate matching.

Label Sentiment: A custom function, label_sentiment, is applied to each row. It combines the Title and Description fields, converts them to lowercase, and checks for the presence of positive or negative keywords:

If a positive keyword is found, it labels the news as 1 (positive). If a negative keyword is found, it labels the news as 0 (negative). If no keywords are found, it returns None (neutral or no match).

```
[ ]: import pandas as pd
     import re

     # Load the CSV file
     df = pd.read_csv('nifty_it_50_stock_news.csv')

     # Define positive and negative words/phrases
     pos_words_to_search = [
         r'jump', r'rise', r'up', r'soar', r'surge', r'leap', r'climb', r'increase',␣
      ↪r'grow', r'boost',
         r'rocket', r'skyrocket', r'advance', r'gain', r'improve', r'strengthen',␣
      ↪r'bullish', r'remarkable',
         r'outstanding', r'healthy', r'strong', r'optimistic', r'upward',␣
      ↪r'exceeds', r'outperforms'
     ]

     neg_words_to_search = [
         r'plunge', r'drop', r'fall', r'decline', r'slump', r'slide', r'tumble',␣
      ↪r'crash', r'collapse', r'plummet',
    r'sink', r'downturn', r'decrease', r'reduction', r'loss', r'dip',␣
  ↪r'retreat', r'reversal', r'setback',
    r'weakness', r'volatility', r'uncertainty', r'risk', r'concerns', r'fears',␣
  ↪r'worries', r'caution',
    r'warning', r'alert', r'red flag', r'headwinds', r'challenges',␣
  ↪r'obstacles', r'hurdles', r'unclear',
    r'downward', r'bearish', r'negative', r'soft', r'weak', r'sluggish',␣
  ↪r'stagnant', r'flat', r'lackluster',
    r'disappointing', r'underwhelming', r'unimpressive', r'uninspiring',␣
  ↪r'gloomy', r'bleak', r'recession',
    r'contraction', r'slowdown', r'stagnation', r'depression', r'crisis',␣
  ↪r'turmoil', r'instability',
    r'depreciation', r'devaluation', r'write-down', r'impairment', r'losses',␣
  ↪r'discount', r'hit', r'blow',
    r'blowback', r'backlash', r'fallout', r'consequences', r'ramifications',␣
  ↪r'implications', r'repercussions'
]

# Compile regex patterns for positive and negative words
pos_pattern = re.compile(r'\b(?:' + '|'.join(pos_words_to_search) + r')\b', re.
  ↪IGNORECASE)
neg_pattern = re.compile(r'\b(?:' + '|'.join(neg_words_to_search) + r')\b', re.
  ↪IGNORECASE)

# Function to label sentiment based on Title and Description columns
def label_sentiment(row):
    text = f"{row['Title']} {row['Description']}".lower()
    if pos_pattern.search(text):
        return 1  # Positive
    elif neg_pattern.search(text):
        return 0  # Negative
    else:
        return None  # Neutral or no match

# Apply the sentiment labeling function
df['label'] = df.apply(label_sentiment, axis=1)

# Save the labeled data to a new CSV file
df.to_csv('nifty_it_50_stock_news_labeled.csv', index=False)

# Display a sample of the labeled data
print(df[['Title', 'Description', 'label']].head())

# Show the count of each label
print(df['label'].value_counts())
```

```
                                               Title  \
0  Nifty, Sensex plunge as all sectors slip in th…
1  Taking Stock: Market reacts to mixed macroecon…
2  Closing Bell: Nifty below 24,150, Sensex plung…
3  Stock Radar: Vodafone Idea, Orchid Pharma, JSW…
4  Taking Stock: Sensex, Nifty end flat amid Hind…

                                         Description  label
0  Titan Company, Apollo Hospitals, Dr Reddy's La…    0.0
1  Titan Company, Apollo Hospitals, Dr Reddy's La…    NaN
2  Titan Company, Apollo Hospitals, Dr Reddy's La…    NaN
3  NMDC, Housing & Urban Development Corporation,…    NaN
4  Hero MotoCorp, Axis Bank, ONGC, Infosys and JS…    0.0
label
1.0    150
0.0    124
Name: count, dtype: int64
```

TCS and Infosys income sheet

Removing the whitespaces and creating new csv file

```
[ ]:
import pandas as pd

# Load your CSV file
df = pd.read_csv('merged_output.csv')

print(df.columns)

# Remove trailing white spaces from column names
df.columns = df.columns.str.strip()

# Verify the column names
print(df.columns)

# Save the updated DataFrame to a new CSV file
df.to_csv('merged_output_updated.csv', index=False)

# Download the updated CSV file
# from google.colab import files
# files.download('updated_file.csv')

# Display the first row
print(df.head(1))
```

```
Index(['Date ', 'series ', 'OPEN ', 'HIGH ', 'LOW ', 'PREV. CLOSE ', 'ltp ',
       'close ', 'vwap ', '52W H ', '52W L ', 'VOLUME ', 'VALUE ',
       'No of trades '],
      dtype='object')
Index(['Date', 'series', 'OPEN', 'HIGH', 'LOW', 'PREV. CLOSE', 'ltp', 'close',
       'vwap', '52W H', '52W L', 'VOLUME', 'VALUE', 'No of trades'],
      dtype='object')
          Date series      OPEN      HIGH       LOW PREV. CLOSE       ltp  \
0  31-Dec-2020     EQ  2,900.00  2,905.00  2,845.00    2,909.30  2,864.95

      close      vwap     52W H     52W L     VOLUME               VALUE  \
0  2,862.75  2,874.36  2,952.00  1,506.05  40,40,956  11,61,51,70,401.55

  No of trades
0     1,30,170
```