JPMC #2 and Index Replication Optimization

AI Studio Final Presentation

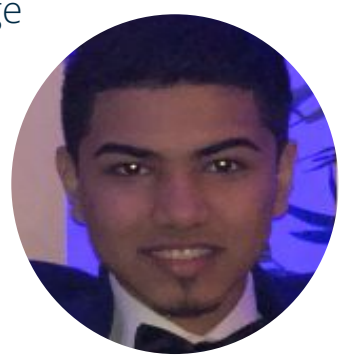
Break Through Tech New York @Cornell Tech
December 15th 2022

Ariana Bhigroog
Queens College

Omina Elsheikh
Baruch College

Andreea Ibanescu
Queens College

Taohid Shadat
Baruch College



Fahad Tahir
Hunter College

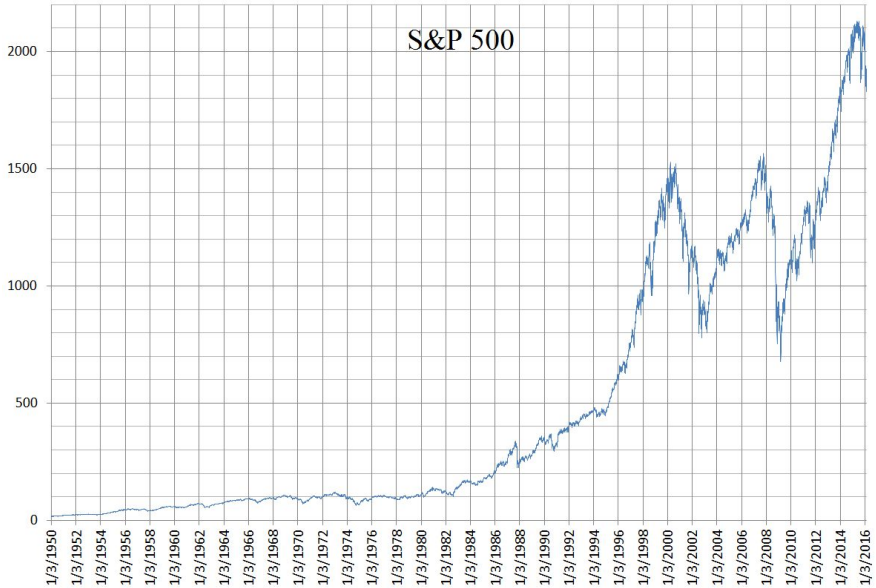
“Our goal is to create an portfolio that replicates the performance of the S&P 500 using Natural Language Constraints and Optimization techniques.”

S&P 500

- The S&P500 is a stock market index tracking the performance of 500 large companies listed on exchanges in the United States
- One of the most followed indices
- Often used as a reference benchmark

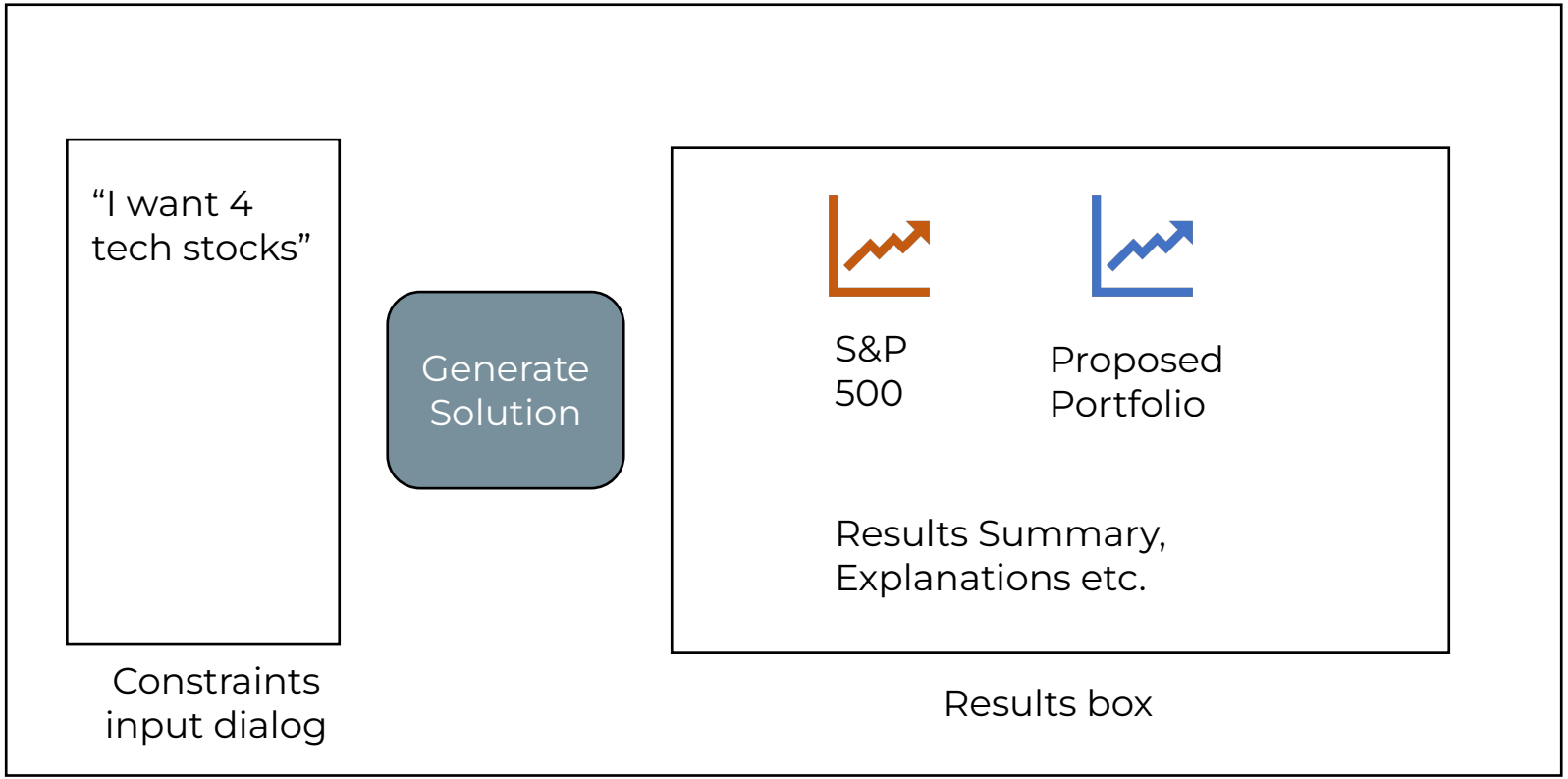
- We would like to create a portfolio to replicate the the S&P 500 performance with:

1. Less companies -> Cost reduction
2. Arbitrary constraints -> User preferences



**historic price movement of S&P 500

Goal


"I want 4 tech stocks"
Generate Solution
S&P 500
Proposed Portfolio
Results Summary, Explanations etc.
Constraints input dialog
Results box

Outcome

User Input:

i want 5 tech stocks

GENERATE SOLUTION

Proposed Allocation

Company	Weight(%)
AAPL	20
ORCL	20
AVGO	20
MA	20
AKAM	20

Correlation to S&P500: 0.9257449380895086

Total Portfolio Return: 75.09%

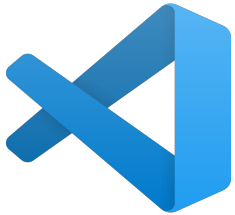
Total S&P500 Return: 46.2%

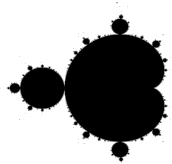
Plan

Task	W1	W2	W3	W4	W5	W6	W7	W8
Create Working Groups								
Get Familiar with Python Libraries and **Data**								
Structure from Natural Language Constraints								
Build, Merge and Compute Similarity between Time Series								
Build MILP **Model**								
NLP and Optimization **Integration**								
Dashboard Integration								

All team members NLP group Optimization group

Resources

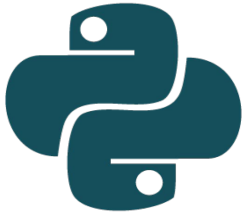




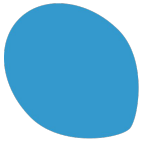
TextBlob

NumPy

plotly



NLTK



Natural Language Processing

The goal of the NLP portion is to extract 3 constraints out of a sentence:

- Number Constraint (3)
- Mathematical Constraint (<=, >=, >, <, =)
- Sector Constraint (Technology, Energy, etc.)

Example Inputs:

- "I want to invest in ***at least 3 tech*** companies" <- 3 (number); >= (mathematical); tech (sector)

Extract Number Constraint

```
def extract_number(inp):
  res = re.findall(r'\d+', inp)
  if len(res) > 0:
    res = [int(res[0])]
  #if it comes here that means the input doesn't have a numeric form and probably word format
  if not res:
    for w in inp.split():
      if not res:
        try:
          res = [w2n.word_to_num(w)]
        except:
          res = []
  return res
```

Extract Mathematical Constraint

```
def extract_mathematical(inp):
  #dictionary of key words for inequalities
  d = {
      "=": ["only", "exactly"],
      ">=": ["at least", "minimum", "no less than", "no fewer than", "greater than or equal to"],
      "<=": ["at most", "maximum", "no more than", "not above", "does not exceed", "less than or equal
to"],
      ">": ["more than", "exceeds", "over", "above", "greater than"],
      "<": ["under", "below", "fewer than", "beneath", "less than"]
    }
  res = []
  for key, value in d.items():
    for w in value:
      if w in inp.lower():
        res.append(key)
  return res
```

Extract Sector Constraint

```
def extract_cat(inp):
  sectors = {
      'Industrials': [],
      'Health Care': [],
      'Information Technology': ['technology', 'tech', 'game'],
      'Communication Services': [],
      'Consumer Staples': [],
      'Consumer Discretionary': [],
      'Utilities': [],
      'Financials': ['bank'],
      'Materials': [],
      'Real Estate': [],
      'Energy': ['green', 'green-energy', 'energy']}
  res = []
  #the parts of speech we want to extract
  pos_wanted = ["NN", "JJ", "JJS", "JJR", "NNS"]
  words = nltk.word_tokenize(inp)
  #tag words with their part of speech
  tagged = nltk.pos_tag(words)
  for word,pos in tagged:
    if pos in pos_wanted and ps.stem(word) != "stock" and ps.stem(word) != "compani":
      for key, val in sectors.items():
        if word.lower() in val:
          res.append(key)
  return res
```

Word2Vec Model

```
model = Word2Vec(sent, min_count=1,size= 50,workers=3, window =5, sg = 1)

model.similarity('Health Care', 'Industrials')
-0.23644204
```

Optimization

Purpose: Understanding our clients needs by taking their numerical and sector constraints from our NLP side, and using them to return the most optimal portfolio (most similarity to S&P500 and best returns)

Example:

- Portfolio 1 (Apple, Tesla, Microsoft) → 70% similar to S&P500 | 50% return
- Portfolio 2 (Google, Palantir, Meta) → 60% similar to S&P500 | 30% return

General Steps For Optimization:

1. Compute log returns of daily stock price
2. Create a correlation matrix of the log returns of each stock
3. Feed correlation matrix into K-Means algorithm to give us the best stocks
4. Creating a portfolio of stocks given market cap weighting

Computing Log Returns

“A sequence of data points that occur in successive order over some period of time.”
- Investopedia

- Computing Log Returns is not only an industry standard but is essential in creating the correlation matrix for our stocks.
- Calculate the log of returns: ***Log((Closing Price (Day2) - Log(Closing Price (Day 1))*** plotted over a course of a 2 year period.

Computing K-Means

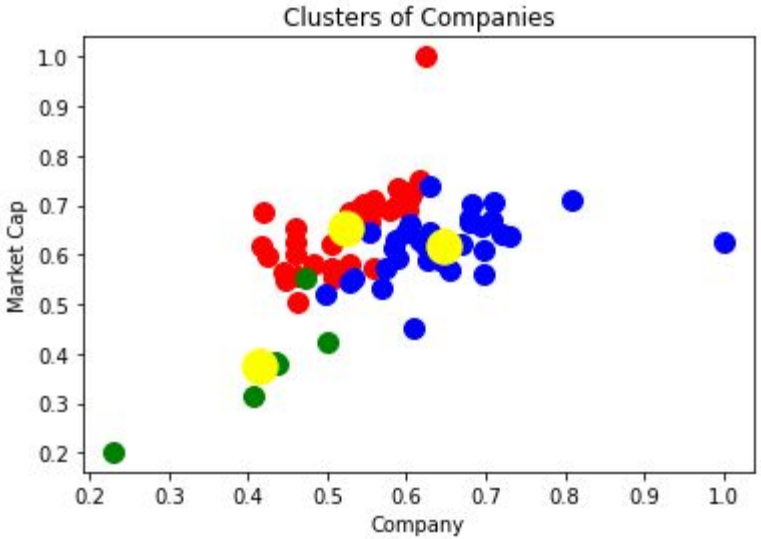
How do we manage to output the highest valued companies from each industry?

K- means: Based on a graph with clustered data points, and using a specific distance formula calculate similarity.

- Works best with scattered data
- Created clusters for each industry and returned the stock with the highest market cap from each cluster.

Computing K-Means Continued

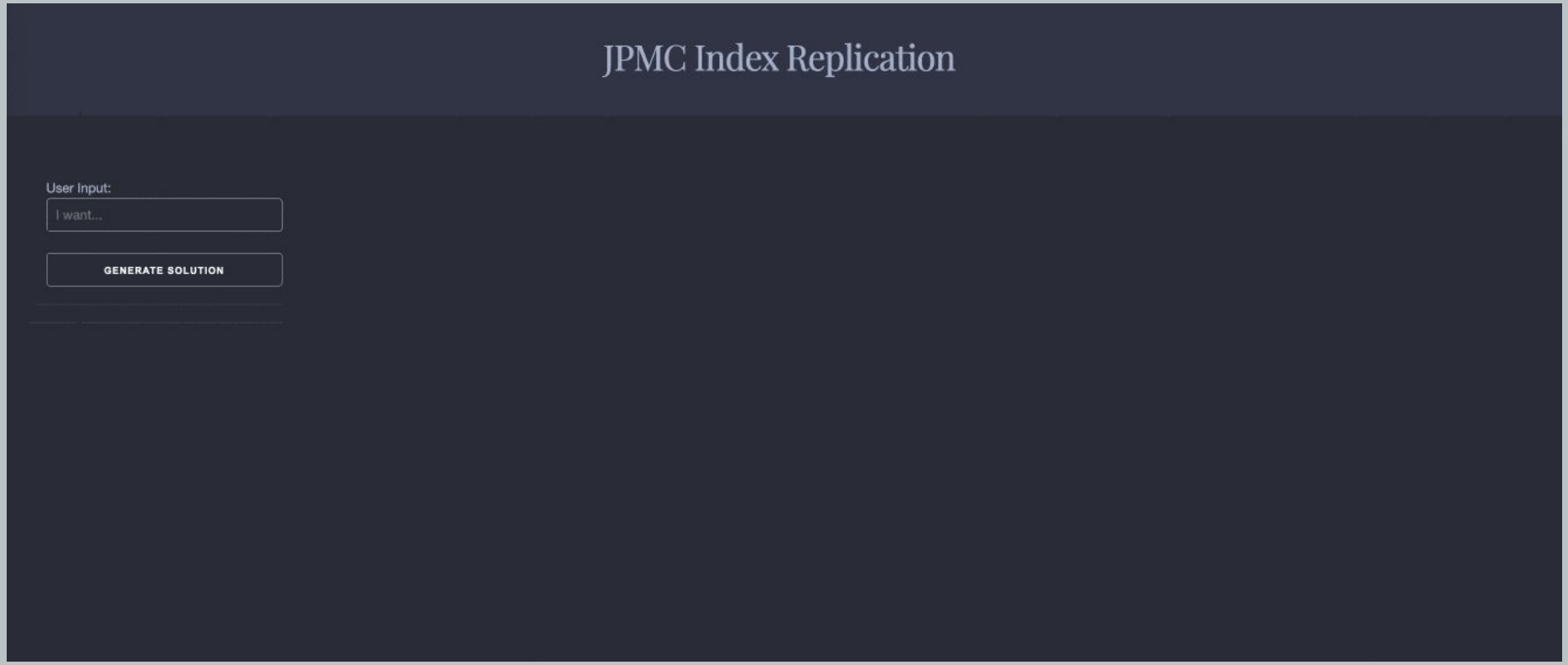
- Each cluster is directly correlated to the number constraint given to us by our client. And will be the number of stocks our k-means algorithm returns.
- The centroids are denoted by the yellow circles. Which basically signifies the center of each cluster.



Optimization Model

```
def optimization(mth, sector = "Information Technology"):
    df = pd.DataFrame()
    #initializing the resulting list of companies
    res = []
    #creating the matrix based on sector
    matrix = create_matrix(sector)
    #this gives us the list of companies within the matrix
    companies = matrix.index.tolist()
    df.index = companies
    #pulling all values from matrix correlation
    X = matrix.values
    #fitting Kmeans model and using math constraint as the number of clusters
    kmeans = KMeans(n_clusters = mth).fit(X)
    df['cluster'] = kmeans.labels_
    for i in range(mth):
        cluster_companies = df[df.cluster == i].index.tolist()
        filt = mkdf[mkdf['Symbol'].isin(cluster_companies)]
        maxx = filt['marketCap'].max()
        res.append(filt[filt['marketCap'] == maxx]['Symbol'].tolist()[0])
    return res, df
```

Demo


JPMC Index Replication
User Input:
I want...
GENERATE SOLUTION

Future Improvements

- **Input:** NLP model to handle more complicated sentence structures
- **Data Processing:** Automate data collection/processing via API
- **Model:** Find/test other ML models to improve optimization process
- **Frontend:** Improve UI/UX of dashboard, incorporate more features (buttons, charts, layout etc.)
- **Code:** Clean up code for more readability


Questions?