Ranking companies

I modeled the trends in H1B visa (The primary visa granted in Tech companies when hiring foreign nationals) to identify companies that hire a robust number of foreign nationals.

Generally, there are two parts in total.

For the first part, I did some data wrangling on companies' information especially about the stock price. This is because I want to check whether there is relationship between stock price and number of H1B approvals in the following analysis.

For the second part, I concatenated the table of companies and table of H1B and applied two linear models of # of H1B approvals on different variables. Finally, I chose the better one from then based on Residual Sum of Square and visulized the result.

The H1B visa data is from the US government but its quality is not guaranteed. We will NOT address most data quality issues in this exam.

Part 1

Step 1 - Getting company performance

Context: choosing a company should factor in their compensation package. Companies often offer stock as a part of their compensation to align the incentives between the employee and company performance.

To get the stock information, we will leverage the API from AlphaVantage. Please obtain the desired data for the listed companies under the variable `stock_symbols` below. The desired data from AlphaVantage is:

- The data under `"TIME_SERIES_MONTHLY_ADJUSTED"`
- The data type should be in the JSON format
- You should use the `requests` package
- You code should use a for-loop to iterate through companies
- HINT: You will need to apply for an API Key, you should feel free to paste your key into the exam.

In [1]:
```
import requests

stock_symbols = ["INFY", "MSFT"]
```

In [2]:
```
import requests
import json
```

```
# replace the "demo" apikey below with your own key from https://www.alphavanta
stock_symbols = ["INFY", "MSFT"]
url = "https://www.alphavantage.co/query"
stock = {}
for symbol in stock_symbols:
    payload = {"function": "TIME_SERIES_MONTHLY_ADJUSTED",
           "symbol": symbol,
           "apikey": "W6TGOO1BSKWNDLXR"
        }
    resp = requests.get(url, params=payload)
    stock[symbol] = resp.json()
print(resp)
```

```
<Response [200]>
```

In [3]:
```
stock.keys()
```

Out[3]:
```
dict_keys(['INFY', 'MSFT'])
```

Step 2 - Data wrangling

You'll also need this dictionary:

```
employer2stock = {
    "INFOSYS LIMITED": "INFY",
    "ACCENTURE LLP": "ACN",
    "NTT DATA": "NTDTY",
    "MICROSOFT": "MSFT",
    "AMAZON WEB SERVICES": "AMZN",
    "INTEL": "INTC",
    "NVIDIA": "NVDA",
    "ORACLE AMERICA": "ORCL",
    "QUALCOMM TECHNOLOGIES": "QCOM",
    "VMWARE": "VMW"
}
```

The goal is to wrangle the data into a data frame where the rows correspond to different companies and the columns are:

- `name` : The company name as in `employer2stock`, these should be strings
- `symbol` : The stock symbol as in `employer2stock`, these should be strings
- `adj_close` : The monthly adjusted close price, these should be floats
- `date` : The end date of the adjusted close price in that month, these should be datetime objects
- `year` : The year for that corresponding record, these should be integers

In [4]:
```
import json
with open('stocks_monthly_adjusted.json','r') as f:
    data = json.load(f)
```

In [5]:
```
import pandas as pd
import numpy as np
```

In [6]:
```
keys = list(data.keys())
employer2stock = {
    "INFOSYS LIMITED": "INFY",
    "ACCENTURE LLP": "ACN",
    "NTT DATA": "NTDTY",
    "MICROSOFT": "MSFT",
    "AMAZON WEB SERVICES": "AMZN",
    "INTEL": "INTC",
    "NVIDIA": "NVDA",
    "ORACLE AMERICA": "ORCL",
    "QUALCOMM TECHNOLOGIES": "QCOM",
    "VMWARE": "VMW"
}
```

In [ ]:
```

```

In [7]:
```
import datetime
stocks = []

format = '%Y-%m-%d'

for key in keys:
    stock = {}
    for employer in employer2stock:
        if key == employer2stock[employer]:
            stock['name'] = employer
            stock['symbol'] = key
    temp = data[key]['Monthly Adjusted Time Series']
    stock['date'] = []
    stock['adj_close'] = []
    stock['year'] = []
    for date in temp:
        stock['adj_close'].append(temp[date]['4. close'])
        stock['date'].append(datetime.datetime.strptime(date, format).date())
        stock['year'].append(int(datetime.datetime.strptime(date, format).date(
    stocks.append(stock)

new_stocks = []
for x in stocks:
    for i in range(len(x['date'])):
        new_stock = {}
        new_stock['name'] = x['name']
        new_stock['symbol'] = x['symbol']
        new_stock['adj_close'] = float(x['adj_close'][i])
        new_stock['date'] = x['date'][i]
        new_stock['year'] = x['year'][i]
        new_stocks.append(new_stock)




stocks_df = pd.DataFrame(new_stocks)

stocks_df['date'] = pd.to_datetime(stocks_df['date'])


print('The number of rows is {}.'.format(stocks_df.shape[0]))
print('\n')
```

```
print(stocks_df.head(3))
print('\n')
print(stocks_df.dtypes)
```

```
The number of rows is 2533.


              name symbol  adj_close       date  year
0  INFOSYS LIMITED   INFY      19.71 2022-11-29  2022
1  INFOSYS LIMITED   INFY      18.73 2022-10-31  2022
2  INFOSYS LIMITED   INFY      16.97 2022-09-30  2022


name                 object
symbol               object
adj_close           float64
date         datetime64[ns]
year                  int64
dtype: object
```

Step 3 - Summarize data

Summarize `stocks_df` into a data frame, called `stocks_summ`, where the rows correspond to a year and company pair and the columns correspond to:

- `name`: The company name as in `employer2stock`
- `symbol`: The stock symbol as in `employer2stock`
- `year`: The year
- `avg_adj_close`: The average monthly adjusted close price in the corresponding year
- `std_adj_close`: The standard deviation of the monthly adjusted close price in the corresponding year

In [8]:
```
stocks_summ = stocks_df.groupby(by = ['name','symbol','year']).agg({'adj_close'
```

In [9]:
```
stocks_summ.columns = ['name', 'symbol', 'year','avg_adj_close', 'std_adj_close
```

In [10]:
```
print('The number of rows is {}.'.format(stocks_summ.shape[0]))
print('\n')
print(stocks_summ.head(3))
```

```
The number of rows is 220.


            name symbol  year  avg_adj_close  std_adj_close
0  ACCENTURE LLP    ACN  2001      18.948000       5.779063
1  ACCENTURE LLP    ACN  2002      20.109167       4.179773
2  ACCENTURE LLP    ACN  2003      19.713333       3.828017
```

Step 4 - Filtering

Given most of students are foreign nationals, the number of H1B visas assigned to each company is useful. This is the primary work visa applied by Tech companies to hire foreign nationals.

Please find `h1bexports.csv`, a processed version of data from H1B DataHub. Each row corresponds to a unique fiscal year and employer combination and the columns are:

- `Fiscal Year`: Oct 1 of the year to Sept 30 the next year.
- `Employer`: the employer filing for the H1B visa
- `Initial Approval`: the number of newly applied H1B visas that are approved
- `Initial Denial`: the number of newly applied H1B visas that are denied
- `Continuing Approval`: the number of H1B visas approved in the past that is again approved this year
- `Continuing Denial`: all remaining H1B visas applications not included in the 3 categories above

- Please create a data frame call `friendly` that filters the data such that we only have companies that satisfy the following conditions:
  - They have at least 10 years worth of H1B visas being newly approved, these years do NOT need to be consecutive
  - They have at least 1 newly approved H1B visas in both 2021 and 2022
- Please report how many companies satisfy these criteria in a human readable message (You do not need to articulate the condition these companies satisfy).

In [11]:
```
h1b = pd.read_csv('h1bexports.csv')
h1b.dtypes
```

Out[11]:
```
Fiscal Year              int64
Employer                object
Initial Approval         int64
Initial Denial           int64
Continuing Approval      int64
Continuing Denial        int64
dtype: object
```

In [12]:
```
h1b_temp = h1b.groupby(by = ['Employer']).agg({'Fiscal Year': ['count']}).reset
```

In [13]:
```
h1b_temp.columns = ['Employer', 'Counts']
```

In [14]:
```
#Companies 10 yesrs more h1b visa
h1b_temp1 = h1b_temp[h1b_temp['Counts']>= 10].reset_index()
```

In [15]:
```
h1b_temp2 = h1b.groupby(by = ['Employer', 'Fiscal Year']).agg({'Initial Approva
h1b_temp2.columns = ['Employer', 'Year', 'Sum']
h1b_temp2 = h1b_temp2[(h1b_temp2['Year'] == 2021) | (h1b_temp2['Year'] == 2022)
h1b_temp2 = h1b_temp2[h1b_temp2['Sum']>=1]
h1b_temp2 = h1b_temp2.groupby(by = ['Employer']).agg({'Year':['count']}).reset_

h1b_temp2.columns = ['Employer', 'Counts']
h1b_temp2 = h1b_temp2[h1b_temp2['Counts'] == 2].reset_index()
```

In [16]:
```
lst1 = list(h1b_temp1['Employer'])
lst2 = list(h1b_temp2['Employer'])
```

```
def intersection(lst1, lst2):
    lst3 = [value for value in lst1 if value in lst2]
    return lst3

employers = intersection(lst1, lst2)
friendly = pd.DataFrame(employers)
friendly.columns = ['Employer']
```

In [17]:
```
print('There are {} number of employers satisfy the criteria.'.format(friendly.
```

```
There are 3043 number of employers satisfy the criteria.
```

Part 2

Step 5 - Joining

- Please join `friendly` and `stocks_summ` by year and company name into a data frame called `jdf`, keeping only records that exist in both sources. Please print out the number of rows in `jdf` in a human readable message.
  - Fiscal year 2021 in `friendly` should be joined with `year` 2021 in `stocks_summ` for the sake of the exam. Don't overthink this.
- Please make `jdf` only have the columns `year`, `name`, `avg_adj_close`, `std_adj_close`, and `Initial Approval`.

In [18]:
```
fri_back = pd.read_csv('Q4_backup_final_fall2022.csv')
stock_back = pd.read_csv('Q3_backup_final_fall2022.csv')
fri_back.head()
```

Out[18]:

	Fiscal Year	Employer	Initial Approval	Initial Denial	Continuing Approval	Continuing Denial
0	2009	22ND CENTURY TECHNOLOGIES	9	4	12	3
1	2009	3I INFOTECH	10	0	10	0
2	2009	3K TECHNOLOGIES	3	1	13	0
3	2009	3M COMPANY	0	0	13	1
4	2009	A T KEARNEY	16	2	13	0

In [19]:
```
stock_back.head()
```

Out[19]:

	name	symbol	year	avg_adj_close	std_adj_close
0	ACCENTURE LLP	ACN	2010	32.860058	2.777684
1	ACCENTURE LLP	ACN	2011	45.010992	2.833929
2	ACCENTURE LLP	ACN	2012	52.013758	3.847300
3	ACCENTURE LLP	ACN	2013	63.956433	3.588126
4	ACCENTURE LLP	ACN	2014	70.829533	3.035421

In [20]:
```
jdf_temp = pd.merge(fri_back, stock_back, how = 'inner', left_on = ['Employer',
```

In [21]:
```
jdf = pd.DataFrame()
jdf['year'] = jdf_temp['Fiscal Year']
jdf['name'] = jdf_temp['name']
jdf['avg_adj_close'] = jdf_temp['avg_adj_close']
jdf['std_adj_close'] = jdf_temp['std_adj_close']
jdf['Initial Approval'] = jdf_temp['Initial Approval']
```

In [22]:
```
print('There are {} number of rows in jdf.'.format(jdf.shape[0]))
```

```
There are 110 number of rows in jdf.
```

In [23]:
```
jdf.head()
```

Out[23]:

	year	name	avg_adj_close	std_adj_close	Initial Approval
0	2010	AMAZON WEB SERVICES	6.965875	1.201915	6
1	2010	INTEL	13.991783	0.959326	502
2	2010	MICROSOFT	20.554533	1.764116	1939
3	2010	NVIDIA	3.046492	0.643316	52
4	2010	ORACLE AMERICA	21.308108	2.584877	61

Step 6 - SQL

If `friendly` and `stocks_summ` were tables in an SQL database called `"h1b"`, how would the join in Q5 look in an SQL query? You should pretend any space in a variable name is replaced with an underscore, i.e. `"Initial Approval"` would be a column named `"Initial_Approval"` in this database.

In [24]:
```
query = """
    select s.year as year,
           s.name as name,
           s.avg_adj_close as avg_adj_close,
           s.std_adj_close as std_adj_close,
           f.'Initial Approval' as 'Initial Approval'
    from stocks_summ as s
    inner join
    friendly as f
    on s.year = f.'Fiscal Year' and s.name = f.'Employer'
    order by year

"""
```

In [ ]:

Step 7 - Modeling trends

- For each employer in `jdf`, please fit 2 models for `Initial Approval` using oridinary least squares regression (sklearn.linear_model.LinearRegression) but limited to data in 2020 and earlier. The models should be:
  - Model 1 should regress on `year`, `avg_adj_close`, and `std_adj_close`
  - Model 2 should regress on `year` alone. Please ignore the issue of forecasting for the sake of the exam.
- Please show which model predicts the outcomes in 2021 better with a reasonable metric and a human-readable message.
- From the winning model, please store the coefficient corresponding to `Fiscal Year` (we'll call this the "trend") and the prediction at year 2021. Store your results in a data frame called `output` with columns: `name`, `trend`, `pred_2021`.
  - For clarification, `pred_2021` is the linear model predictions of `Initial Approval` in 2021

In [25]:
```
import pandas as pd
jdf_2 = pd.read_csv('Q5_backup_final_fall2022.csv')
jdf_2.head()
```

Out[25]:

	year	name	avg_adj_close	std_adj_close	Initial Approval
0	2009	AMAZON WEB SERVICES	4.536542	1.277117	1
1	2009	INTEL	11.427517	1.944207	818
2	2009	MICROSOFT	17.584392	3.698794	1505
3	2009	NVIDIA	2.782425	0.692122	150
4	2009	ORACLE AMERICA	16.948083	2.151346	1

In [26]:
```
jdf_mod = jdf_2[jdf_2['year']<=2020]
jdf_mod.head()
```

Out[26]:

	year	name	avg_adj_close	std_adj_close	Initial Approval
0	2009	AMAZON WEB SERVICES	4.536542	1.277117	1
1	2009	INTEL	11.427517	1.944207	818
2	2009	MICROSOFT	17.584392	3.698794	1505
3	2009	NVIDIA	2.782425	0.692122	150
4	2009	ORACLE AMERICA	16.948083	2.151346	1

In [27]:
```
jdf_val = jdf_2[jdf_2['year'] == 2021]
```

In [28]:
```
from sklearn.linear_model import LinearRegression
```

In [29]:
```
def RMSE(y, y_hat):
    return np.sqrt(sum((y-y_hat)**2)/len(y))
```

In [30]:
```
import numpy as np
name_list = jdf_2['name'].unique()
```

```
rmse = []
coef_list = []
pred_list = []

for name in name_list:
    train = jdf_mod[jdf_mod['name'] == name]
    mod1_x = train[['year','avg_adj_close','std_adj_close']]
    mod_y = train['Initial Approval']
    mod2_x = np.array(train['year']).reshape(-1,1)

    validation = jdf_val[jdf_val['name'] == name]
    val1_x = validation[['year','avg_adj_close','std_adj_close']]
    val_y = validation['Initial Approval']
    val2_x = np.array(validation['year']).reshape(-1,1)

    ols1 = LinearRegression().fit(mod1_x,mod_y)
    ols2 = LinearRegression().fit(mod2_x,mod_y)

    y1_pred = ols1.predict(val1_x)
    y2_pred = ols2.predict(val2_x)

    rmse.append([RMSE(y1_pred,val_y), RMSE(y2_pred,val_y)])
    coef_list.append(ols2.coef_)
    pred_list.append(y2_pred)
```

In [31]:
```
rmse1 = 0
rmse2 = 0
for i in range(len(rmse)):
    rmse1 += rmse[i][0]
    rmse2 += rmse[i][1]
```

In [32]:
```
rmse1
```

Out[32]:
```
8696.011825326392
```

In [33]:
```
rmse2
```

Out[33]:
```
4848.810966810935
```

In [34]:
```
print('According to the metric RMSE, the error of model 2 is smaller, so I can
```

```
According to the metric RMSE, the error of model 2 is smaller, so I can conclu
de model 2 is better.
```

In [35]:
```
output_data = {
    'name': name_list,
    'trend': coef_list,
    'pred_2021': pred_list
}
output = pd.DataFrame(output_data)
```

In [36]:
```
output
```

Out[36]:

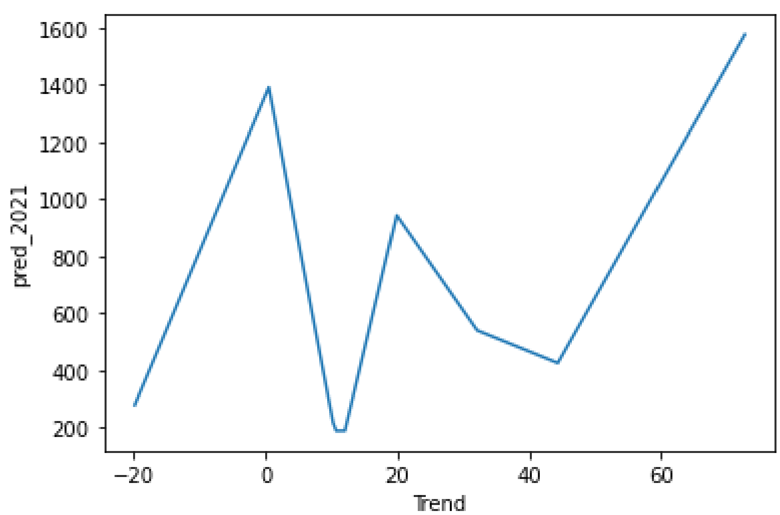
	name	trend	pred_2021
0	AMAZON WEB SERVICES	[44.37062937062935]	[425.7424242424313]
1	INTEL	[19.898601398601393]	[943.2575757575687]
2	MICROSOFT	[0.5104895104895057]	[1391.651515151515]
3	NVIDIA	[10.290209790209786]	[217.9696969696961]
4	ORACLE AMERICA	[32.1118881118881]	[540.2272727272721]
5	VMWARE	[10.762237762237758]	[189.45454545454413]
6	INFOSYS LIMITED	[72.73333333333332]	[1576.444444444467]
7	NTT DATA	[12.083333333333333]	[189.75]
8	QUALCOMM TECHNOLOGIES	[-19.869047619047606]	[279.4642857142826]

Step 8 - Visualization

- Please visualize the relationship between `trend` and `pred_2021` across the companies in `output` .
- Please add informative x and y axis labels in the figure
- Please report the Pearson correlation between `trend` and `pred_2021` in a human-readable message.

In [37]:

```
out = pd.read_csv('Q7_backup_final_fall2022.csv')
import seaborn as sns
scatter = sns.lineplot(data=out, x='trend', y="pred_2021")
scatter.set(xlabel='Trend', ylabel='pred_2021')
import matplotlib.pyplot as plt
my_rho = np.corrcoef(out['trend'], out['pred_2021'])
```



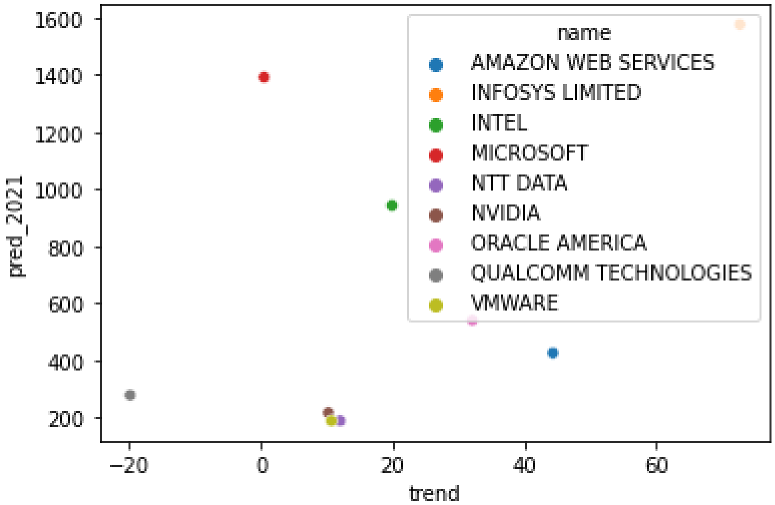
In [38]:

```
print('The pearson coefficient between trend and pred_2021 is {}.'.format(my_rh
```

```
The pearson coefficient between trend and pred_2021 is 0.4740422731080643.
```

In [39]:

```
g = sns.scatterplot(x = 'trend', y = 'pred_2021', data = out, hue = 'name')
```


name
AMAZON WEB SERVICES
INFOSYS LIMITED
INTEL
MICROSOFT
NTT DATA
NVIDIA
ORACLE AMERICA
QUALCOMM TECHNOLOGIES
VMWARE
pred_2021
1600
1400
1200
1000
800
600
400
200
−20
0
20
40
60
trend