EDA_ETL

November 25, 2020

EDA

```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.decomposition import PCA
     from sklearn import linear_model
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestRegressor
     import yfinance as yf
     import numpy as np
     from sklearn.preprocessing import StandardScaler
```

We import the stock price of Starbucks(SBUX) and Apple(AAPL) and the industrial index of S&P 500(SPY).

```
[53]: spy= yf.download(
              tickers = "SPY",
              period = "5d",
              interval = "1m",
              group_by = 'ticker',
              auto_adjust = True,
              prepost = False,
              threads = True,
              proxy = None)
      spy.reset_index(inplace=True,drop=False)
      spy.rename(columns = {'Close':'SPY'}, inplace = True)
      spy=spy.drop(['Open', 'High','Low','Volume'], axis=1)


      sbux= yf.download(
              tickers = "SBUX",
              period = "5d",
              interval = "1m",
              group_by = 'ticker',
              auto_adjust = True,
              prepost = False,
              threads = True,
              proxy = None)
```

```
sbux.reset_index(inplace=True,drop=False)
sbux=sbux.drop(['Open', 'High','Low','Volume','Datetime'], axis=1)
sbux.rename(columns = {'Close':'SBUX'}, inplace = True)

aapl= yf.download(
        tickers = "AAPL",
        period = "5d",
        interval = "1m",
        group_by = 'ticker',
        auto_adjust = True,
        prepost = False,
        threads = True,
        proxy = None)
aapl.reset_index(inplace=True,drop=False)
aapl=aapl.drop(['Open', 'High','Low','Volume','Datetime'], axis=1)
aapl.rename(columns = {'Close':'AAPL'}, inplace = True);
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

[54]:
```
data = pd.concat([spy, sbux,aapl], axis=1, sort=False)
```

[55]:
```
data
```

[55]:
```
                      Datetime         SPY       SBUX        AAPL
0    2020-11-18 09:30:00-05:00  360.760010  98.510002  118.910004
1    2020-11-18 09:31:00-05:00  360.679993  98.565498  118.684998
2    2020-11-18 09:32:00-05:00  360.730011  98.669998  118.620003
3    2020-11-18 09:33:00-05:00  360.660004  98.705002  118.377701
4    2020-11-18 09:34:00-05:00  360.695007  98.644997  118.499901
…                          …         …          …         …
1944 2020-11-24 15:55:00-05:00  363.364990  98.330002  115.170097
1945 2020-11-24 15:56:00-05:00  363.140015  98.300003  115.014999
1946 2020-11-24 15:57:00-05:00  363.359985  98.349998  115.149902
1947 2020-11-24 15:58:00-05:00  363.109985  98.300003  115.025002
1948 2020-11-24 15:59:00-05:00  363.179993  98.320000  115.169998

[1949 rows x 4 columns]
```

ETL

[56]:
```
import pymongo
client = pymongo.MongoClient()
```

[57]:
```
db = client.get_database("stock")
collection = db.get_collection("stock")
update_count = 0
```

```python
for record in data.to_dict('records'):
    result = collection.replace_one(
        filter={'Datetime': record['Datetime']},
        replacement=record,
        upsert=True)
    if result.matched_count > 0:
        update_count += 1
print(f"rows={data.shape[0]}, update={update_count}, "
      f"insert={data.shape[0]-update_count}")
```

```
rows=1949, update=1949, insert=0
```