```python
from airflow import DAG
from airflow.operators.python import PythonOperator

from datetime import datetime, timedelta
import yfinance as yf
import pickle
import pandas as pd
from sklearn.linear_model import LinearRegression
from pathlib import Path
import logging


COMPANIES = ['AAPL', 'GOOGL', 'FB', 'MSFT', 'AMZN']

DF_DIRECTORY = "hw4/stock_dfs.pkl"
MODELS_DIRECTORY = "hw4/models.pkl"
HISTORY_DIREC = 'hw4/histories.pkl'
RELATIVE_ERRORS_DIREC = "hw4/relative_errors.csv"
PREDICTION_DIREC = 'hw4/predictions.pkl'

FEATURE_COLS = ["Open", "Low", "Close", "Volume", 'High']
PREDICTED_COL = ["High"]

default_args = {
    'owner': 'yewen',
    'depends_on_past': False,
    'email': ['yz4175@columbia.edu'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1
}


def get_date_today():
    """Get date today in str format such as 20201119. """
    return datetime.today().strftime('%Y%m%d')


def get_date_yesterday():
```

```
40      """Return yesterday in str such as 2021-11-19
    . """
41      return (datetime.today() - timedelta(days=1)).
    strftime('%Y-%m-%d')
42
43
44  def get_data():
45      """Get new data in the past 10 days up to
    including yesterday.
46      For example, if today is 11/22/2021, only collect
     data up to
47      including 11/21/2021. Append as a list. Save to
    pickle file.
48      """
49      stock_dfs = []
50      for i, company in enumerate(COMPANIES):
51          # because I run at 7 am everyday, using
    period='10d'
52          # would get the data in the past 10 days
    including
53          # yesterday
54          stock_df = yf.download(company, period='10d')
55          stock_dfs.append(stock_df)
56      with open(DF_DIRECTORY, 'wb') as f:
57          pickle.dump(stock_dfs, f)
58
59
60  def get_historical_data():
61      """Get all historical data up to including
    yesterday. """
62      histories = []
63      for i, company in enumerate(COMPANIES):
64          # because I run at 7 am every day, this would
65          # actually just get me all the historic data
66          history = yf.download(company, period='max')
67          histories.append(history)
68      with open(HISTORY_DIREC, 'wb') as f:
69          pickle.dump(histories, f)
70
71
72  def compute_relative_error():
```

```
 73      """Compute the relative error for today, update
    CSV.
 74      For example, today is 11/22, compute the
    relative error between
 75      prediction and high on 11/21.
 76
 77      On the 1st run, it will pass.
 78      On the 2nd run, it will create a csv.
 79      On the later runs, it will read the csv.
 80      """
 81      with open(DF_DIRECTORY, 'rb') as f:
 82          # this data includes stock up to 11/21 in
    the past
 83          # 10 days
 84          stock_dfs = pickle.load(f)
 85
 86      if Path(PREDICTION_DIREC).is_file():
 87          with open(PREDICTION_DIREC, 'rb') as f:
 88              # a list of 5 predictions made on 11/21.
 89              predictions = pickle.load(f)
 90
 91      if Path(PREDICTION_DIREC).is_file():
 92          relative_errors = []
 93          for i, stock_df in enumerate(stock_dfs):
 94              # high price on 11/21.
 95              high = stock_df.iloc[-1, :]["High"]
 96              # prediction on 11/21.
 97              prediction = predictions[i]
 98              relative_error = (prediction - high) /
    high
 99              relative_errors.append(relative_error)
100
101          if Path(RELATIVE_ERRORS_DIREC).is_file():
102              logging.info("Updating relative_errors
    csv...")
103              relative_errors_df = pd.read_csv(
    RELATIVE_ERRORS_DIREC)
104              relative_errors_df[get_date_yesterday
    ()] = relative_errors
105              relative_errors_df.to_csv(
    RELATIVE_ERRORS_DIREC)
```

```
106         else:
107             # create a CSV
108             logging.info("Creating relative_errors
    csv...")
109             relative_errors_df = pd.DataFrame(data=
    relative_errors,
110 
    columns=[get_date_yesterday()],
111                                               index=
    COMPANIES)
112             relative_errors_df.to_csv(
    RELATIVE_ERRORS_DIREC)
113 
114 
115 def train_models():
116     """Make Linear Regression Models. """
117     with open(HISTORY_DIREC, 'rb') as f:
118         histories_df = pickle.load(f)
119 
120     models = []
121     for history_df in histories_df:
122         n = history_df.shape[0]
123         logging.info(f"History shape: {history_df.
    shape}")
124         reg = LinearRegression()
125         Xs = []
126         ys = []
127         for i in range(0, n-10):
128             slice = history_df.iloc[i:i + 10, :][
    FEATURE_COLS]
129             X = slice.to_numpy().flatten().tolist()
130             Xs.append(X)
131             y = history_df.iloc[i + 10]['High']
132             ys.append(y)
133         reg.fit(Xs, ys)
134         models.append(reg)
135 
136     with open(MODELS_DIRECTORY, 'wb') as f:
137         pickle.dump(models, f)
138 
139 
```

```
140 def make_prediction():
141     """Make a prediction for today. """
142     with open(MODELS_DIRECTORY, 'rb') as f:
143         models = pickle.load(f)
144 
145     with open(DF_DIRECTORY, 'rb') as f:
146         stock_dfs = pickle.load(f)
147 
148     predictions = []
149     for i, stock_df in enumerate(stock_dfs):
150         model = models[i]
151         X = stock_df[FEATURE_COLS].to_numpy().
    flatten().reshape(1, -1)
152         logging.info(X)
153         prediction = model.predict(X)[0]
154         predictions.append(prediction)
155 
156     with open(PREDICTION_DIREC, 'wb') as f:
157         pickle.dump(predictions, f)
158 
159 
160 with DAG('stock',
161          default_args=default_args,
162          description='Stock Price Analysis for HW4',
163          catchup=False,
164          start_date=datetime(2021, 1, 1),
165          schedule_interval='0 7 * * *',
166          ) as dag:
167     get_data_task = PythonOperator(task_id='get_data
    ',
168                                    python_callable=
    get_data)
169 
170     get_historical_data_task = PythonOperator(
    task_id='get_historic_data',
171 
    python_callable=get_historical_data)
172 
173     make_prediction_task = PythonOperator(task_id="
    make_prediction",
174 
```

```
174 python_callable=make_prediction)
175
176     train_model_task = PythonOperator(task_id='
    train_model',
177
    python_callable=train_models)
178
179     compute_relative_error_task = PythonOperator(
    task_id="compute_relative_error",
180
    python_callable=compute_relative_error)
181
182     # task dependencies
183     # (get_data_task >> get_historical_data_task >>
    train_model_task
184     #  >> make_prediction_task >>
    compute_relative_error_task)
185     (get_data_task >> get_historical_data_task >>
186      train_model_task >> make_prediction_task >>
    compute_relative_error_task)
187
```