

Creating Large Dataset for Model Training

This is comprised of momentum indicators of several handpicked stocks that I think represent a good space in the U.S.A market, indices that the U.S.A market depend on, and commodities that the U.S.A depends on.

```
In [1]: ### Importing Needed Packages  
import pandas as pd  
import yfinance as yf  
import pandas_ta as ta
```

```
In [2]: ### Function SPC  
  
# This on will append all needed momentum indicators to the dataframe provided  
def momentum_indicators(df):  
    df.ta.rsi(append=True)  
    df.ta.macd(append=True)  
    df.ta.roc(append=True)  
    df.ta.willr(append=True)  
    df.ta.stochrsi(append=True)  
    df.ta.inertia(append=True)  
    df.ta.dm(append=True)  
    return df  
  
# This on will add the indicators from the above function and then remove the un  
def add_indicators(data):  
    for key in data.keys():  
        data[key] = momentum_indicators(data[key])  
        drops = ["Adj Close", "High", "Open", "Low", "Close", "Volume"]  
        data[key].drop(columns=drops, inplace=True)  
    return data  
  
# This one will create the target data from the S&P500 index  
def make_target(SP500, ema_len=30):  
    SP500.ta.ema(length = 30, append=True)  
    SP500["EMA_30_FT"] = SP500.EMA_30.shift(periods=-30)  
    SP500["Diff"] = (SP500.EMA_30_FT - SP500.EMA_30)  
    SP500["Diff_ratio"] = SP500.Diff / SP500.EMA_30  
    SP500["Diff_shift"] = SP500.Diff_ratio.shift(periods=-30)  
    SP500["Target"] = SP500.Diff_shift.apply(lambda x: -1 if x < -0.01 else (1 if  
    return SP500.Target
```

Downloading Data

Packages and Functions are loaded and now to grab the data. The code below will download all the **ticker** symbols using a yahoo finance scraper called yfinance. Its a great package to have. I highly recommend it. The data will come in a as dataframe with columns for Open, High, Low, Close, Volume, and Adjusted Close. My code will the pandas dataframes into a dictionary for storage.

```
In [3]: data = {}
tickers = {}
tickers["company_tickers"] = ["JNJ", "PFE", "PG", "MSFT", "AMZN", "CVX", "XOM", "
tickers["index_tickers"] = ["^GSPC", "^IXIC", "^DJI", "000001.SS", "^N225"]
tickers["commodities_tickers"] = ["CL=F", "HG=F", "GC=F", "SI=F", "RB=F", "NG=F"]

print(f"Loading Companies")
for company in tickers["company_tickers"]:
    data[company] = yf.download(company, start="2000-01-01", end="2021-11-30", interval="1d")

print(f"Loading Indecies")
for index in tickers["index_tickers"]:
    data[index] = yf.download(index, start="2000-01-01", end="2021-11-30", interval="1d")

print(f"Loading Commodities Futures")
for commodity in tickers["commodities_tickers"]:
    data[commodity] = yf.download(commodity, start="2000-01-01", end="2021-11-30", interval="1d")
```

```
Loading Companies
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
Loading Indecies
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
```

The Data is downloaded now we need the indicators!

Creating the Dataset

Adding the indicators

```
In [4]: data = add_indicators(data)
```

Gluing together all the dataframes created.

```
In [5]: dataframe = pd.concat([data[key] for key in data.keys()], axis=1)
dataframe.shape
```

Out[5]: (5699, 253)

Downloading the Target Data!

```
In [6]: SPY = yf.download("^GSPC", start="2000-01-01", end="2021-11-30", interval="1D")
[*****100%*****] 1 of 1 completed
```

Making the Target column

```
In [7]: target = make_target(SPY)
```

```
In [8]: target.shape
```

Out[8]: (5513,)

Appending the Target Column!

```
In [9]: dataframe = pd.concat([dataframe, target], axis=1)
dataframe.dropna(axis=0, inplace=True)
dataframe.shape
```

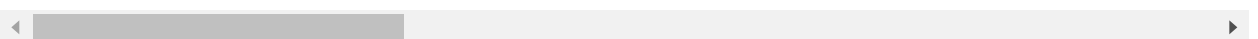
Out[9]: (4620, 254)

```
In [10]: dataframe.head(5)
```

Out[10]:

	RSI_14	MACD_12_26_9	MACDh_12_26_9	MACDs_12_26_9	ROC_10	WILLR_14	STOC
Date							
2001-01-09	47.930677	0.196231	-0.312678	0.508910	-3.017241	-69.364162	
2001-01-10	41.186181	0.035816	-0.378475	0.414291	-6.349206	-93.063584	
2001-01-11	34.228196	-0.227382	-0.513338	0.285957	-10.254854	-98.181818	
2001-01-12	40.877630	-0.346242	-0.505759	0.159517	-9.725537	-82.727273	
2001-01-16	40.107042	-0.450377	-0.487915	0.037538	-10.350982	-85.454545	

5 rows × 254 columns



This is the Data! Now I'll save it as a csv file.

```
In [11]: dataframe.to_csv("data/momentum_market_data.csv")
```

