Downloading Historical OHLC mid, bid and ask Data for EUR/USD

```python
import warnings
warnings.filterwarnings("ignore")

from api.oanda_api import OandaApi
from infrastructure.instrument_collection import instrumentCollection
from infrastructure.collect_data import run_collection
import pandas as pd

# Initializing our session
api = OandaApi()

# Downloading the instruments available and storing it in a jason file
instruments = api.get_account_instruments()
instrumentCollection.CreateFile(instruments, './data')

instrumentCollection.LoadInstruments('./data')

# Downloading OHLC prices, 2 years worth 5 minute candles (this might take a while)
run_collection(instrumentCollection, api, currencies=["EUR", "USD"], years=2)

df = pd.read_pickle("./data/EUR_USD_M5.pkl")
```

```
EUR_USD M5
EUR_USD M5 2023-01-25 18:46:38.614685+00:00 2023-02-05 04:46:38.614685+00:00 --> 2055 candles loaded
EUR_USD M5 2023-02-05 04:46:38.614685+00:00 2023-02-15 14:46:38.614685+00:00 --> 2218 candles loaded
EUR_USD M5 2023-02-15 14:46:38.614685+00:00 2023-02-26 00:46:38.614685+00:00 --> 2103 candles loaded
EUR_USD M5 2023-02-26 00:46:38.614685+00:00 2023-03-08 10:46:38.614685+00:00 --> 2170 candles loaded
EUR_USD M5 2023-03-08 10:46:38.614685+00:00 2023-03-18 20:46:38.614685+00:00 --> 2151 candles loaded
EUR_USD M5 2023-03-18 20:46:38.614685+00:00 2023-03-29 06:46:38.614685+00:00 --> 2134 candles loaded
EUR_USD M5 2023-03-29 06:46:38.614685+00:00 2023-04-08 16:46:38.614685+00:00 --> 2186 candles loaded
EUR_USD M5 2023-04-08 16:46:38.614685+00:00 2023-04-19 02:46:38.614685+00:00 --> 2085 candles loaded
EUR_USD M5 2023-04-19 02:46:38.614685+00:00 2023-04-29 12:46:38.614685+00:00 --> 2235 candles loaded
EUR_USD M5 2023-04-29 12:46:38.614685+00:00 2023-05-09 22:46:38.614685+00:00 --> 2038 candles loaded
EUR_USD M5 2023-05-09 22:46:38.614685+00:00 2023-05-20 08:46:38.614685+00:00 --> 2282 candles loaded
EUR_USD M5 2023-05-20 08:46:38.614685+00:00 2023-05-30 18:46:38.614685+00:00 --> 1989 candles loaded
EUR_USD M5 2023-05-30 18:46:38.614685+00:00 2023-06-10 04:46:38.614685+00:00 --> 2331 candles loaded
EUR_USD M5 2023-06-10 04:46:38.614685+00:00 2023-06-20 14:46:38.614685+00:00 --> 1942 candles loaded
EUR_USD M5 2023-06-20 14:46:38.614685+00:00 2023-07-01 00:46:38.614685+00:00 --> 2379 candles loaded
EUR_USD M5 2023-07-01 00:46:38.614685+00:00 2023-07-11 10:46:38.614685+00:00 --> 1893 candles loaded
EUR_USD M5 2023-07-11 10:46:38.614685+00:00 2023-07-21 20:46:38.614685+00:00 --> 2424 candles loaded
EUR_USD M5 2023-07-21 20:46:38.614685+00:00 2023-08-01 06:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2023-08-01 06:46:38.614685+00:00 2023-08-11 16:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-08-11 16:46:38.614685+00:00 2023-08-22 02:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2023-08-22 02:46:38.614685+00:00 2023-09-01 12:46:38.614685+00:00 --> 2423 candles loaded
EUR_USD M5 2023-09-01 12:46:38.614685+00:00 2023-09-11 22:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2023-09-11 22:46:38.614685+00:00 2023-09-22 08:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-09-22 08:46:38.614685+00:00 2023-10-02 18:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2023-10-02 18:46:38.614685+00:00 2023-10-13 04:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-10-13 04:46:38.614685+00:00 2023-10-23 14:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2023-10-23 14:46:38.614685+00:00 2023-11-03 00:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-11-03 00:46:38.614685+00:00 2023-11-13 10:46:38.614685+00:00 --> 1837 candles loaded
EUR_USD M5 2023-11-13 10:46:38.614685+00:00 2023-11-23 20:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-11-23 20:46:38.614685+00:00 2023-12-04 06:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2023-12-04 06:46:38.614685+00:00 2023-12-14 16:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2023-12-14 16:46:38.614685+00:00 2023-12-25 02:46:38.614685+00:00 --> 1791 candles loaded
EUR_USD M5 2023-12-25 02:46:38.614685+00:00 2024-01-04 12:46:38.614685+00:00 --> 1906 candles loaded
EUR_USD M5 2024-01-04 12:46:38.614685+00:00 2024-01-14 22:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2024-01-14 22:46:38.614685+00:00 2024-01-25 08:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2024-01-25 08:46:38.614685+00:00 2024-02-04 18:46:38.614685+00:00 --> 1887 candles loaded
EUR_USD M5 2024-02-04 18:46:38.614685+00:00 2024-02-15 04:46:38.614685+00:00 --> 2385 candles loaded
EUR_USD M5 2024-02-15 04:46:38.614685+00:00 2024-02-25 14:46:38.614685+00:00 --> 1935 candles loaded
EUR_USD M5 2024-02-25 14:46:38.614685+00:00 2024-03-07 00:46:38.614685+00:00 --> 2338 candles loaded
EUR_USD M5 2024-03-07 00:46:38.614685+00:00 2024-03-17 10:46:38.614685+00:00 --> 1983 candles loaded
EUR_USD M5 2024-03-17 10:46:38.614685+00:00 2024-03-27 20:46:38.614685+00:00 --> 2302 candles loaded
EUR_USD M5 2024-03-27 20:46:38.614685+00:00 2024-04-07 06:46:38.614685+00:00 --> 2018 candles loaded
EUR_USD M5 2024-04-07 06:46:38.614685+00:00 2024-04-17 16:46:38.614685+00:00 --> 2254 candles loaded
EUR_USD M5 2024-04-17 16:46:38.614685+00:00 2024-04-28 02:46:38.614685+00:00 --> 2067 candles loaded
EUR_USD M5 2024-04-28 02:46:38.614685+00:00 2024-05-08 12:46:38.614685+00:00 --> 2206 candles loaded
EUR_USD M5 2024-05-08 12:46:38.614685+00:00 2024-05-18 22:46:38.614685+00:00 --> 2114 candles loaded
EUR_USD M5 2024-05-18 22:46:38.614685+00:00 2024-05-29 08:46:38.614685+00:00 --> 2123 candles loaded
EUR_USD M5 2024-05-29 08:46:38.614685+00:00 2024-06-08 18:46:38.614685+00:00 --> 2163 candles loaded
EUR_USD M5 2024-06-08 18:46:38.614685+00:00 2024-06-19 04:46:38.614685+00:00 --> 2108 candles loaded
EUR_USD M5 2024-06-19 04:46:38.614685+00:00 2024-06-29 14:46:38.614685+00:00 --> 2211 candles loaded
EUR_USD M5 2024-06-29 14:46:38.614685+00:00 2024-07-10 00:46:38.614685+00:00 --> 2062 candles loaded
EUR_USD M5 2024-07-10 00:46:38.614685+00:00 2024-07-20 10:46:38.614685+00:00 --> 2259 candles loaded
EUR_USD M5 2024-07-20 10:46:38.614685+00:00 2024-07-30 20:46:38.614685+00:00 --> 2013 candles loaded
EUR_USD M5 2024-07-30 20:46:38.614685+00:00 2024-08-10 06:46:38.614685+00:00 --> 2307 candles loaded
EUR_USD M5 2024-08-10 06:46:38.614685+00:00 2024-08-20 16:46:38.614685+00:00 --> 1966 candles loaded
EUR_USD M5 2024-08-20 16:46:38.614685+00:00 2024-08-31 02:46:38.614685+00:00 --> 2354 candles loaded
EUR_USD M5 2024-08-31 02:46:38.614685+00:00 2024-09-10 12:46:38.614685+00:00 --> 1918 candles loaded
EUR_USD M5 2024-09-10 12:46:38.614685+00:00 2024-09-20 22:46:38.614685+00:00 --> 2403 candles loaded
EUR_USD M5 2024-09-20 22:46:38.614685+00:00 2024-10-01 08:46:38.614685+00:00 --> 1870 candles loaded
EUR_USD M5 2024-10-01 08:46:38.614685+00:00 2024-10-11 18:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2024-10-11 18:46:38.614685+00:00 2024-10-22 04:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2024-10-22 04:46:38.614685+00:00 2024-11-01 14:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2024-11-01 14:46:38.614685+00:00 2024-11-12 00:46:38.614685+00:00 --> 1837 candles loaded
EUR_USD M5 2024-11-12 00:46:38.614685+00:00 2024-11-22 10:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2024-11-22 10:46:38.614685+00:00 2024-12-02 20:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2024-12-02 20:46:38.614685+00:00 2024-12-13 06:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2024-12-13 06:46:38.614685+00:00 2024-12-23 16:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2024-12-23 16:46:38.614685+00:00 2025-01-03 02:46:38.614685+00:00 --> 1848 candles loaded
EUR_USD M5 2025-01-03 02:46:38.614685+00:00 2025-01-13 12:46:38.614685+00:00 --> 1849 candles loaded
EUR_USD M5 2025-01-13 12:46:38.614685+00:00 2025-01-23 22:46:38.614685+00:00 --> 2425 candles loaded
EUR_USD M5 2025-01-23 22:46:38.614685+00:00 2025-01-25 18:46:38.614217+00:00 --> 279 candles loaded
*** *** EUR_USD M5 2023-01-25 18:45:00+00:00 2025-01-24 21:55:00+00:00 --> 149167 candles ***
```

Preview Of Our DataFrame

```python
df.head()
```

	time	volume	mid_o	mid_h	mid_l	mid_c	bid_o	bid_h	bid_l	bid_c	ask_o	ask_h	ask_l	ask_c
0	2023-01-25 18:45:00+00:00	245	1.09116	1.09147	1.09112	1.09142	1.09110	1.09140	1.09105	1.09135	1.09123	1.09155	1.09120	1.09148
1	2023-01-25 18:50:00+00:00	309	1.09142	1.09157	1.09102	1.09102	1.09135	1.09150	1.09094	1.09094	1.09149	1.09164	1.09108	1.09109
2	2023-01-25 18:55:00+00:00	181	1.09101	1.09110	1.09084	1.09092	1.09093	1.09102	1.09076	1.09085	1.09109	1.09117	1.09091	1.09098
3	2023-01-25 19:00:00+00:00	317	1.09093	1.09101	1.09068	1.09098	1.09086	1.09094	1.09061	1.09091	1.09100	1.09108	1.09075	1.09106
4	2023-01-25 19:05:00+00:00	359	1.09098	1.09108	1.09088	1.09092	1.09092	1.09101	1.09080	1.09085	1.09105	1.09116	1.09094	1.09099

Dataset Structure and Overview

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149167 entries, 0 to 149166
Data columns (total 14 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   time    149167 non-null  datetime64[ns, tzutc()]
 1   volume  149167 non-null  int64
 2   mid_o   149167 non-null  float64
 3   mid_h   149167 non-null  float64
 4   mid_l   149167 non-null  float64
 5   mid_c   149167 non-null  float64
 6   bid_o   149167 non-null  float64
 7   bid_h   149167 non-null  float64
 8   bid_l   149167 non-null  float64
 9   bid_c   149167 non-null  float64
 10  ask_o   149167 non-null  float64
 11  ask_h   149167 non-null  float64
 12  ask_l   149167 non-null  float64
 13  ask_c   149167 non-null  float64
dtypes: datetime64[ns, tzutc()](1), float64(12), int64(1)
memory usage: 15.9 MB
```

Statistical Summary of the Dataset

```python
df.describe()
```

	volume	mid_o	mid_h	mid_l	mid_c	bid_o	bid_h	bid_l	bid_c	ask_o	ask_h	ask_l	ask_c
count	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000	149167.000000
mean	334.968760	1.080475	1.080638	1.080311	1.080475	1.080394	1.080558	1.080228	1.080394	1.080556	1.080720	1.080391	1.080556
std	307.151709	0.018964	0.018952	0.018975	0.018964	0.018964	0.018953	0.018975	0.018964	0.018964	0.018952	0.018975	0.018964
min	1.000000	1.018430	1.019080	1.017790	1.018440	1.018360	1.019010	1.017710	1.018360	1.018500	1.019160	1.017860	1.018520
25%	151.000000	1.069840	1.070010	1.069700	1.069840	1.069760	1.069930	1.069610	1.069760	1.069930	1.070100	1.069770	1.069930
50%	263.000000	1.082870	1.083020	1.082730	1.082880	1.082800	1.082940	1.082650	1.082800	1.082950	1.083090	1.082810	1.082950
75%	421.000000	1.092660	1.092800	1.092510	1.092660	1.092580	1.092720	1.092430	1.092580	1.092740	1.092880	1.092590	1.092740
max	10707.000000	1.127560	1.127570	1.126750	1.127550	1.127480	1.127490	1.126670	1.127470	1.127650	1.127650	1.126830	1.127630

Transforming Data from Time Bars into Dollar Value Bars

```python
from technicals.dollar_value_bars import generate_dollar_bars

df = generate_dollar_bars(df, 360)
```

Feature Engineering: Adding Technical Indicators

```python
from models.add_indicators import apply_indicators

df = apply_indicators(df)
```

Feature Engineering: Adding Technical Patterns

```python
from technicals.patterns import apply_candle_props

df = apply_candle_props(df)
```

Adding Labels for Model Training using the Triple Barrier Method

```python
from technicals.labeling import tripple_barrier_labeling

df = tripple_barrier_labeling(df, win=4, loss=2).dropna()
```

Overview of Dataset Features

```python
df.columns
```

```
Index(['time', 'volume', 'mid_o', 'mid_h', 'mid_l', 'mid_c', 'bid_o', 'bid_h',
       'bid_l', 'bid_c', 'ask_o', 'ask_h', 'ask_l', 'ask_c', 'spread', 'hour',
       'day_of_week', 'month', 'minute', 'BB_MA10', 'BB_UP10', 'BB_LW10',
       'BB_MA30', 'BB_UP30', 'BB_LW30', 'BB_MA50', 'BB_UP50', 'BB_LW50',
       'ATR_7', 'ATR_14', 'ATR_40', 'EMA20', 'KeUp20_10', 'KeLo20_10', 'EMA50',
       'KeUp50_50', 'KeLo50_50', 'EMA200', 'KeUp200_50', 'KeLo200_50', 'RSI_7',
       'RSI_14', 'RSI_50', 'MACD26_12', 'SIGNAL26_12', 'HIST26_12',
       'MACD52_24', 'SIGNAL52_24', 'HIST52_24', 'direction', 'body_size',
       'body_perc', 'body_lower', 'body_upper', 'body_bottom_perc',
       'body_top_perc', 'mid_point', 'low_change', 'high_change',
       'body_size_change', 'body_size_prev', 'direction_prev',
       'direction_prev_2', 'body_perc_prev', 'body_perc_prev_2',
       'mid_point_prev_2', 'Label', 'trade_duration'],
      dtype='object')
```

Applying Stationarization to Selected Columns

```python
from technicals.stationarize_data import stationarize_data

# Columns that we want to stationarize
stationary_cols = ['volume', 'mid_o', 'mid_h', 'mid_l', 'mid_c', 'bid_o', 'bid_h',
       'bid_l', 'bid_c', 'ask_o', 'ask_h', 'ask_l', 'ask_c', 'BB_MA10', 'BB_UP10', 'BB_LW10',
       'BB_MA30', 'BB_UP30', 'BB_LW30', 'BB_MA50', 'BB_UP50', 'BB_LW50',
       'ATR_7', 'ATR_14', 'ATR_40', 'EMA20', 'KeUp20_10', 'KeLo20_10', 'EMA50',
       'KeUp50_50', 'KeLo50_50', 'EMA200', 'KeUp200_50', 'KeLo200_50', 'RSI_7',
       'RSI_14', 'RSI_50', 'MACD26_12', 'SIGNAL26_12', 'HIST26_12',
       'MACD52_24', 'SIGNAL52_24', 'HIST52_24']

df = stationarize_data(df, stationary_cols).dropna()
```

Selecting Features for Prediction

```python
predictors = ['volume', 'mid_o', 'mid_h', 'mid_l', 'mid_c', 'bid_o', 'bid_h',
       'bid_l', 'bid_c', 'ask_o', 'ask_h', 'ask_l', 'ask_c', 'spread', 'hour',
       'day_of_week', 'month', 'minute', 'BB_MA10', 'BB_UP10', 'BB_LW10',
       'BB_MA30', 'BB_UP30', 'BB_LW30', 'BB_MA50', 'BB_UP50', 'BB_LW50',
       'ATR_7', 'ATR_14', 'ATR_40', 'EMA20', 'KeUp20_10', 'KeLo20_10', 'EMA50',
       'KeUp50_50', 'KeLo50_50', 'EMA200', 'KeUp200_50', 'KeLo200_50', 'RSI_7',
       'RSI_14', 'RSI_50', 'MACD26_12', 'SIGNAL26_12', 'HIST26_12',
       'MACD52_24', 'SIGNAL52_24', 'HIST52_24', 'direction', 'body_size',
       'body_perc', 'body_lower', 'body_upper', 'body_bottom_perc',
       'body_top_perc', 'mid_point', 'low_change', 'high_change',
       'body_size_prev', 'direction_prev',
       'direction_prev_2', 'body_perc_prev', 'body_perc_prev_2',
       'mid_point_prev_2']
```

Splitting Dataset for Hyperparameter Tuning and Model Evaluation

```python
# Will be used for hyperparameter tuning
validation_set = df[:30_000].copy()

# Will be used in the model evaluation faze
test_set = df[30_000:].copy()
```

Hyperparameter Tuning Using Halving Random Search

```python
from sklearn.model_selection import TimeSeriesSplit
from sklearn.ensemble import RandomForestClassifier
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingRandomSearchCV
from scipy.stats import randint


param_distributions = {
    'n_estimators': randint(100, 500),
    'max_depth': randint(5, 20),
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 5),
    'max_features': ['sqrt', 'log2']
}

base_rf = RandomForestClassifier(random_state=42)
tscv = TimeSeriesSplit(n_splits=5)

halving_search = HalvingRandomSearchCV(
    estimator=base_rf,param_distributions=param_distributions,factor=3,
    resource='n_samples', max_resources=10_000, scoring='precision', cv=tscv, verbose=0, random_state=42
)

X = validation_set[predictors]
y = validation_set['Label']
halving_search.fit(X, y)

print("Best Score:", halving_search.best_score_)
print("Best Params:", halving_search.best_params_)
best_rf = halving_search.best_estimator_
```

```
Best Score: 0.45323481116584563
Best Params: {'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 4, 'n_estimators': 224}
```

Evaluating the Optimized Random Forest Model

```python
from sklearn.ensemble import RandomForestClassifier
from technicals.backtesting import model_evaluation

test_set.dropna(inplace=True)
rf_predictions = model_evaluation(test_set, best_rf, predictors, start=10_000, step=10_000, memory='off')
```

```
17.25% there...
34.50% there...
51.75% there...
69.00% there...
86.25% there...
```

Model Precision VS Benchmark Performance

```python
loosing_trades = len(rf_predictions[(rf_predictions['Predictions']==1) & (rf_predictions['Label'] == 0)])
winning_trades = len(rf_predictions[(rf_predictions['Predictions']==1) & (rf_predictions['Label'] == 1)])

precision = (winning_trades / (winning_trades + loosing_trades)) * 100
benchmark = (len(rf_predictions[rf_predictions['Label'] == 1]) / (len(rf_predictions))) * 100

print(f"Precision: {precision:.3f} %")
print(f"BenchMark: {benchmark:.3f} %")
rf_predictions.value_counts()
```

```
Precision: 35.041 %
BenchMark: 34.098 %
```

```
Label  Predictions
0      0.0            26705
1      0.0            13709
0      1.0             4907
1      1.0             2647
Name: count, dtype: int64
```

Saving the model

```python
from joblib import dump

# Save the model
dump(best_rf, 'ML_models/random_forest.pkl')
```

```
['ML_models/random_forest.pkl']
```

Running the Live Trading Bot with our ML Model

```python
from infrastructure.instrument_collection import instrumentCollection
from stream_bot.stream_bot import run_bot

instrumentCollection.LoadInstruments("./data")
run_bot()
```

```
Granularity: M1
EUR_USD: {'pair': 'EUR_USD', 'dollar_threshold': 360, 'model': 'ML_models/random_forest.pkl', 'reward': 4, 'risk': 2, 'probability': 0.6, 'predictors': [['volume', 'mid_o', 'mid_h', 'mid_l', 'mid_c', 'bid_o', 'bid_h', 'bid
_l', 'bid_c', 'ask_o', 'ask_h', 'ask_l', 'ask_c', 'spread', 'hour', 'day_of_week', 'month', 'minute', 'BB_MA10', 'BB_UP10', 'BB_LW10', 'BB_MA30', 'BB_UP30', 'BB_LW30', 'BB_MA50', 'BB_UP50', 'BB_LW50', 'ATR_7', 'ATR_14', 'A
TR_40', 'EMA20', 'KeUp20_10', 'KeLo20_10', 'EMA50', 'KeUp50_50', 'KeLo50_50', 'EMA200', 'KeUp200_50', 'KeLo200_50', 'RSI_7', 'RSI_14', 'RSI_50', 'MACD26_12', 'SIGNAL26_12', 'HIST26_12', 'MACD52_24', 'SIGNAL52_24', 'HIST52_
24', 'direction', 'body_size', 'body_perc', 'body_lower', 'body_upper', 'body_bottom_perc', 'body_top_perc', 'mid_point', 'low_change', 'high_change', 'body_size_prev', 'direction_prev', 'direction_prev_2', 'body_perc_pre
v', 'body_perc_prev_2', 'mid_point_prev_2']]}
```