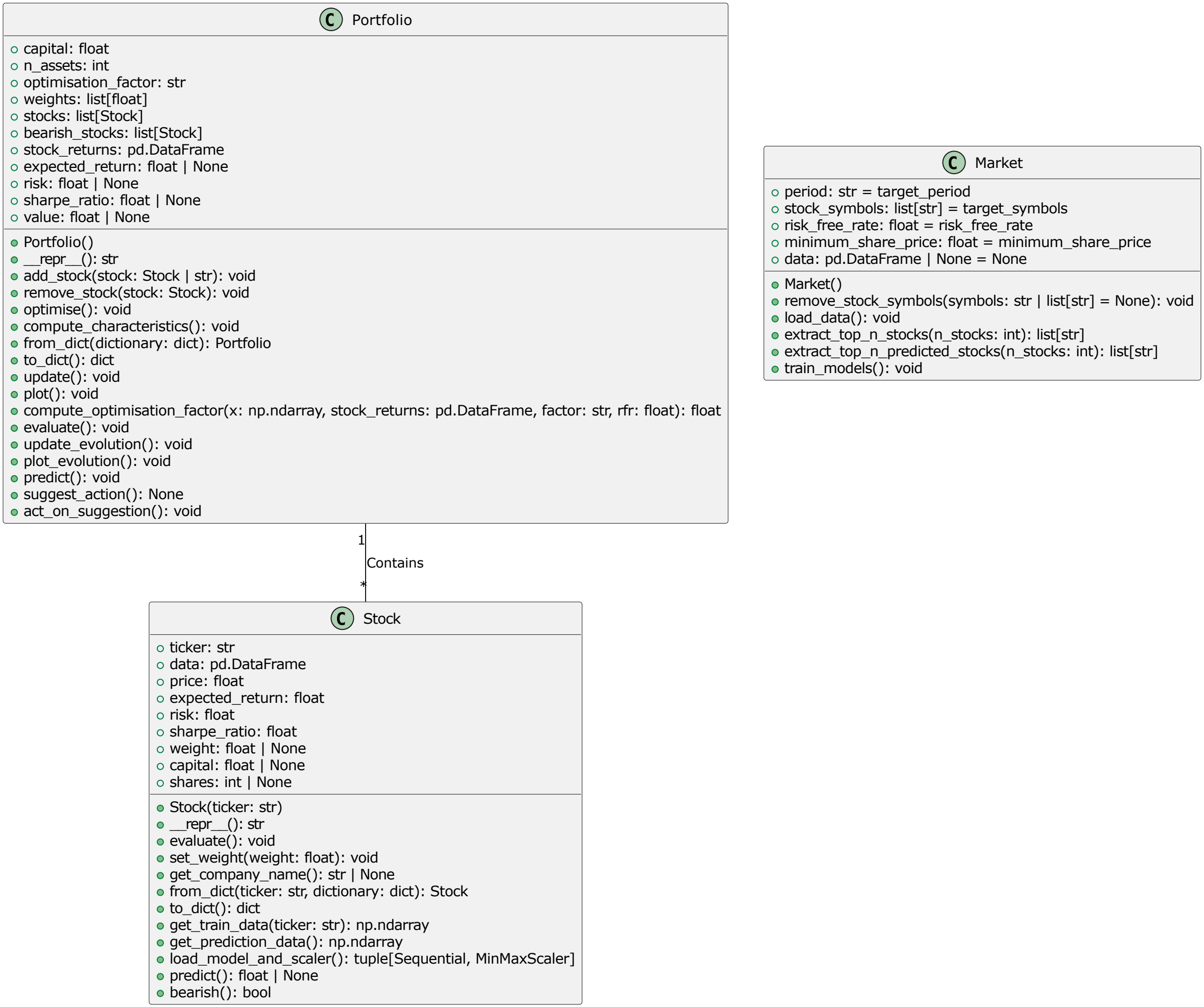
Portfolio
capital: float
n_assets: int
optimisation_factor: str
weights: list[float]
stocks: list[Stock]
bearish_stocks: list[Stock]
stock_returns: pd.DataFrame
expected_return: float | None
risk: float | None
sharpe_ratio: float | None
value: float | None
Portfolio()
__repr__(): str
add_stock(stock: Stock | str): void
remove_stock(stock: Stock): void
optimise(): void
compute_characteristics(): void
from_dict(dictionary: dict): Portfolio
to_dict(): dict
update(): void
plot(): void
compute_optimisation_factor(x: np.ndarray, stock_returns: pd.DataFrame, factor: str, rfr: float): float
evaluate(): void
update_evolution(): void
plot_evolution(): void
predict(): void
suggest_action(): None
act_on_suggestion(): void
1
Contains
*
Market
period: str = target_period
stock_symbols: list[str] = target_symbols
risk_free_rate: float = risk_free_rate
minimum_share_price: float = minimum_share_price
data: pd.DataFrame | None = None
Market()
remove_stock_symbols(symbols: str | list[str] = None): void
load_data(): void
extract_top_n_stocks(n_stocks: int): list[str]
extract_top_n_predicted_stocks(n_stocks: int): list[str]
train_models(): void
Stock
ticker: str
data: pd.DataFrame
price: float
expected_return: float
risk: float
sharpe_ratio: float
weight: float | None
capital: float | None
shares: int | None
Stock(ticker: str)
__repr__(): str
evaluate(): void
set_weight(weight: float): void
get_company_name(): str | None
from_dict(ticker: str, dictionary: dict): Stock
to_dict(): dict
get_train_data(ticker: str): np.ndarray
get_prediction_data(): np.ndarray
load_model_and_scaler(): tuple[Sequential, MinMaxScaler]
predict(): float | None
bearish(): bool