Comparative Analysis of Technical Indicators in Machine Learning Applications

By: Leigh Badua, John Batarse, Catherine Croft, Jing Pu, Jason Rossi, Lari Rupp

Executive Summary - User Story

For our project, we created a Streamlit dashboard, which, given a specific stock, performs a comparative analysis of machine-learning model design in a testing environment.

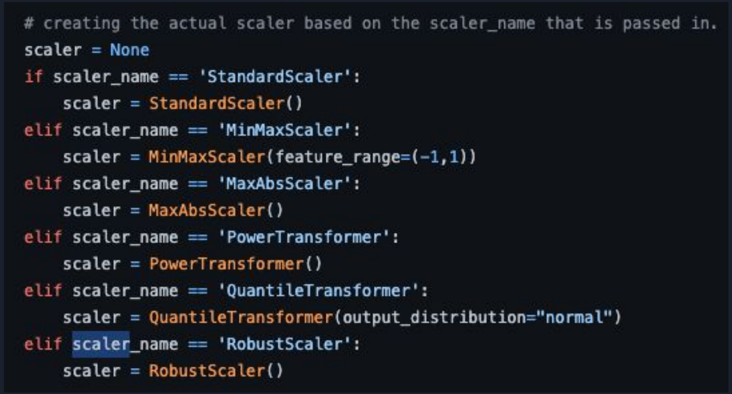
Our minimum viable product demonstrates the framework for a more powerful terminal that could potentially take a stock and design a machine learning model for it.

Selected Machine Learning Models

- SVM
- Random Forest
- Naive Bayes
- AdaBoost

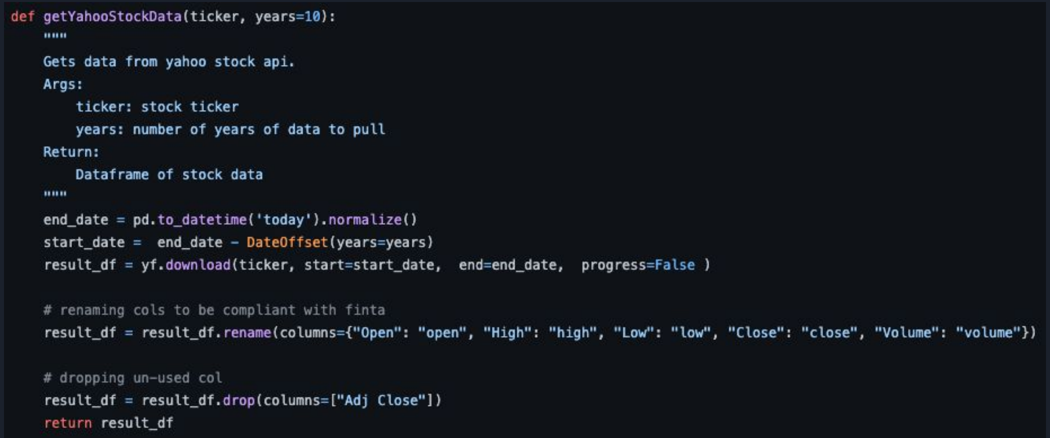
Common supervised machine learning models with binary classification systems

Data Preparation Process



```python
# creating the actual scaler based on the scaler_name that is passed in.
scaler = None
if scaler_name == 'StandardScaler':
    scaler = StandardScaler()
elif scaler_name == 'MinMaxScaler':
    scaler = MinMaxScaler(feature_range=(-1,1))
elif scaler_name == 'MaxAbsScaler':
    scaler = MaxAbsScaler()
elif scaler_name == 'PowerTransformer':
    scaler = PowerTransformer()
elif scaler_name == 'QuantileTransformer':
    scaler = QuantileTransformer(output_distribution="normal")
elif scaler_name == 'RobustScaler':
    scaler = RobustScaler()
```

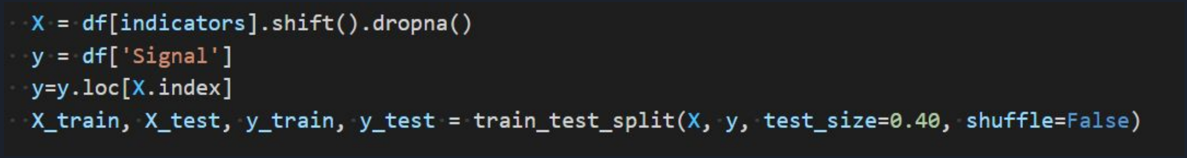
- Data source: Yahoo finance API
- How much data: 10 years
- Libraries utilized: scikit-learn, finta
- Finta map
- Cleanup: OHLCV, drop adjusted close column
- Preparation: Rename columns to lowercase for yfinance
- Scaling the X data



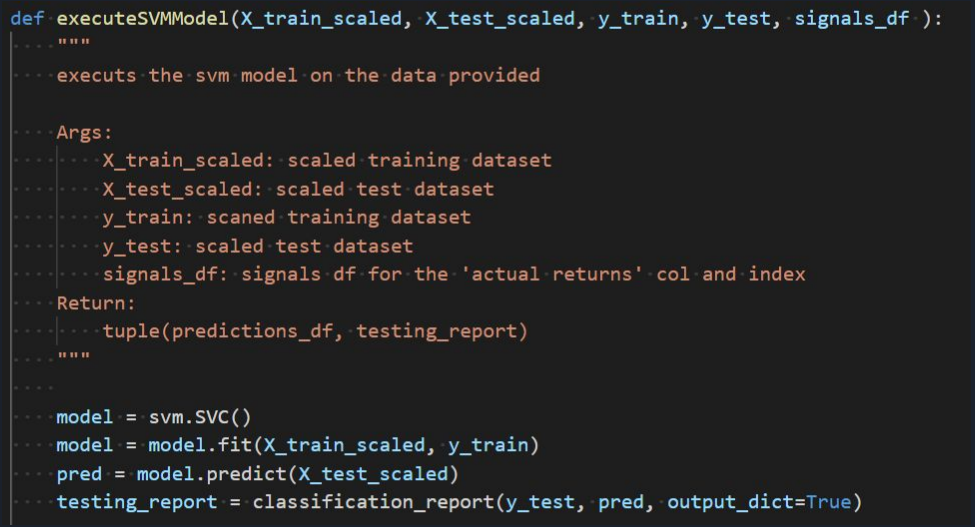
```python
def getYahooStockData(ticker, years=10):
    """
    Gets data from yahoo stock api.
    Args:
        ticker: stock ticker
        years: number of years of data to pull
    Return:
        Dataframe of stock data
    """
    end_date = pd.to_datetime('today').normalize()
    start_date =  end_date - DateOffset(years=years)
    result_df = yf.download(ticker, start=start_date,  end=end_date,  progress=False )

    # renaming cols to be compliant with finta
    result_df = result_df.rename(columns={"Open": "open", "High": "high", "Low": "low", "Close": "close", "Volume": "volume"})

    # dropping un-used col
    result_df = result_df.drop(columns=["Adj Close"])
    return result_df
```

Model Training Process

- Training window: 60%
- Testing window: 40%
- Train, fit, predict
    - For each ML model



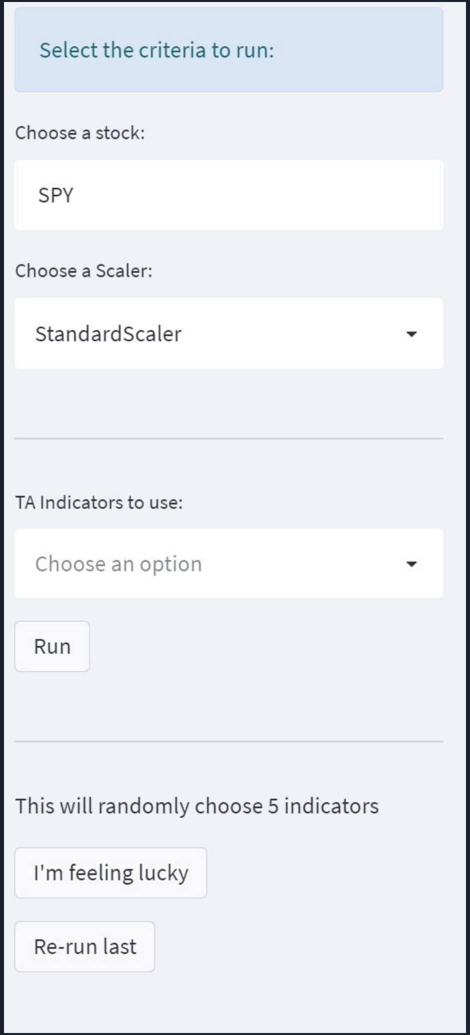
```
X = df[indicators].shift().dropna()
y = df['Signal']
y=y.loc[X.index]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40, shuffle=False)
```



```
def executeSVMModel(X_train_scaled, X_test_scaled, y_train, y_test, signals_df ):
    """
    executs the svm model on the data provided

    Args:
        X_train_scaled: scaled training dataset
        X_test_scaled: scaled test dataset
        y_train: scaned training dataset
        y_test: scaled test dataset
        signals_df: signals df for the 'actual returns' col and index
    Return:
        tuple(predictions_df, testing_report)
    """

    model = svm.SVC()
    model = model.fit(X_train_scaled, y_train)
    pred = model.predict(X_test_scaled)
    testing_report = classification_report(y_test, pred, output_dict=True)
```

Approach

- Evaluate how to improve the algorithm to get better cumulative returns
- Combinations of 5 indicators (or more if your computer can handle it)
  - Test up to 31 different combinations if using 5 indicators
- Top 10 best models
- I'm feeling lucky
  - Randomly chooses 5 indicators
- Re-run last
  - Select a different scaler and see how it compares


Select the criteria to run:

Choose a stock: SPY

Choose a Scaler: StandardScaler

TA Indicators to use: Choose an option

This will randomly choose 5 indicators

Streamlit Demonstration/Results

- SPY
- BND (Vanguard Total Bond Market)
- SPY - I'm feeling lucky, StandardScaler; Re-run with Quantile Transformer
- Indicators for examples: Exponential Moving Average, Relative Strength Index, True Range, Moving Average Convergence Divergence , Bollinger Bands
- Share results during demonstration

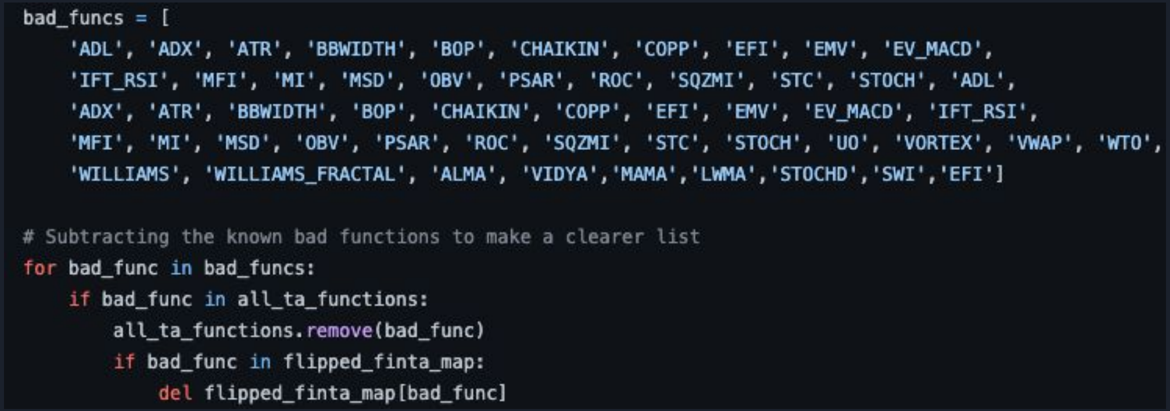
https://share.streamlit.io/lariannrupp/machine_learning_algorithmic_trading_bot/main/streamlit.py

Evaluate Performance

- Classification reports
  - Model accuracy
- Plots with cumulative product returns
  - Percent of strategy returns above actual returns
- Quantile Transformer scaler - best with certain indicators
  - All other scalers similar to Standard Scaler
- Random Forest - use random_state to keep it consistent

Problem Solving

- Choosing a datasource - Alpaca only offering 1000 rows
- Technical indicators - how to trial them all when there’s so many
- Scalers, distribution of data in general
- Calculating percentage of returns
- Deploying to streamlit server - requirements
- “Bad” functions



```
bad_funcs = [
    'ADL', 'ADX', 'ATR', 'BBWIDTH', 'BOP', 'CHAIKIN', 'COPP', 'EFI', 'EMV', 'EV_MACD',
    'IFT_RSI', 'MFI', 'MI', 'MSD', 'OBV', 'PSAR', 'ROC', 'SQZMI', 'STC', 'STOCH', 'ADL',
    'ADX', 'ATR', 'BBWIDTH', 'BOP', 'CHAIKIN', 'COPP', 'EFI', 'EMV', 'EV_MACD', 'IFT_RSI',
    'MFI', 'MI', 'MSD', 'OBV', 'PSAR', 'ROC', 'SQZMI', 'STC', 'STOCH', 'UO', 'VORTEX', 'VWAP', 'WTO',
    'WILLIAMS', 'WILLIAMS_FRACTAL', 'ALMA', 'VIDYA','MAMA','LWMA','STOCHD','SWI','EFI']

# Subtracting the known bad functions to make a clearer list
for bad_func in bad_funcs:
    if bad_func in all_ta_functions:
        all_ta_functions.remove(bad_func)
        if bad_func in flipped_finta_map:
            del flipped_finta_map[bad_func]
```

Next Steps

- Chat bot to discuss machine learning models and algorithmic trading strategies
- Get it consumer-market ready
- Take results from streamlit analysis and create an algorithmic trading strategy to make a trade
- Run all permutations overnight on larger machine (i.e., company computer)

Thank you!