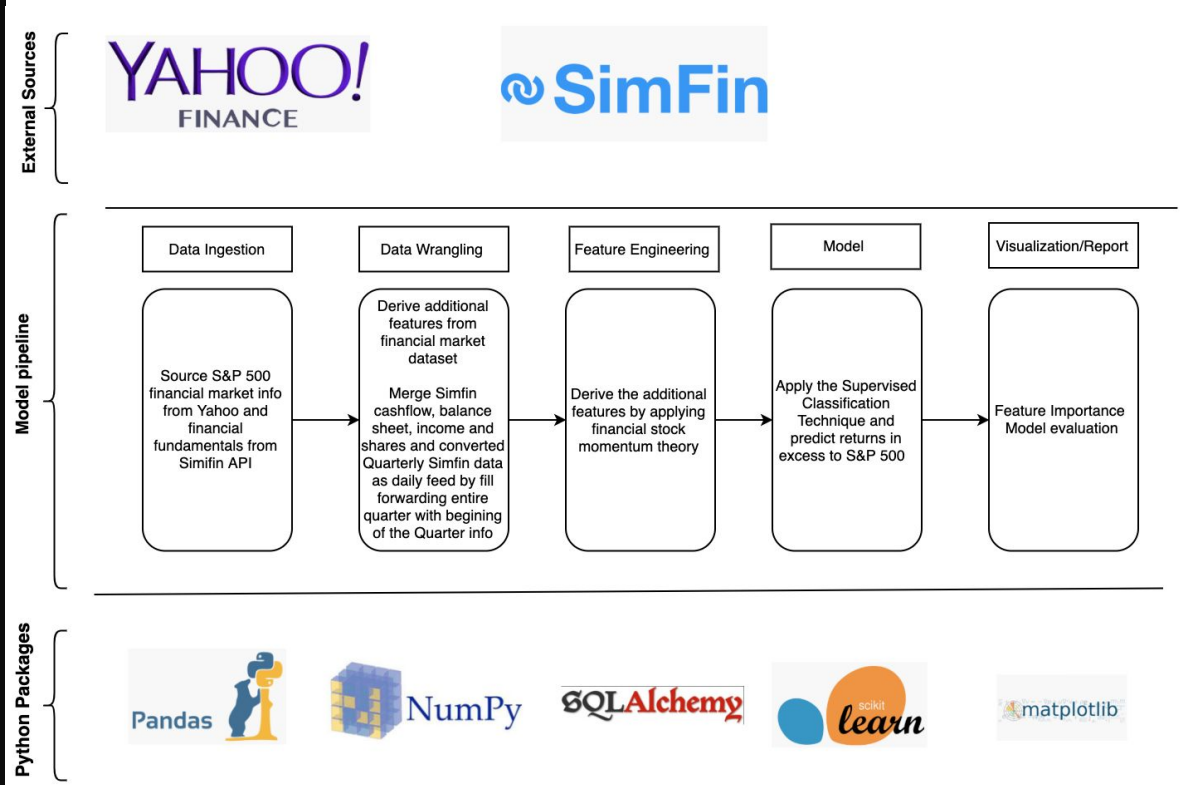
PASSIVE PORTFOLIO MANAGEMENT

Jared Berry, George Krug, Maria Matsira,
and Satya Prakash Yandra

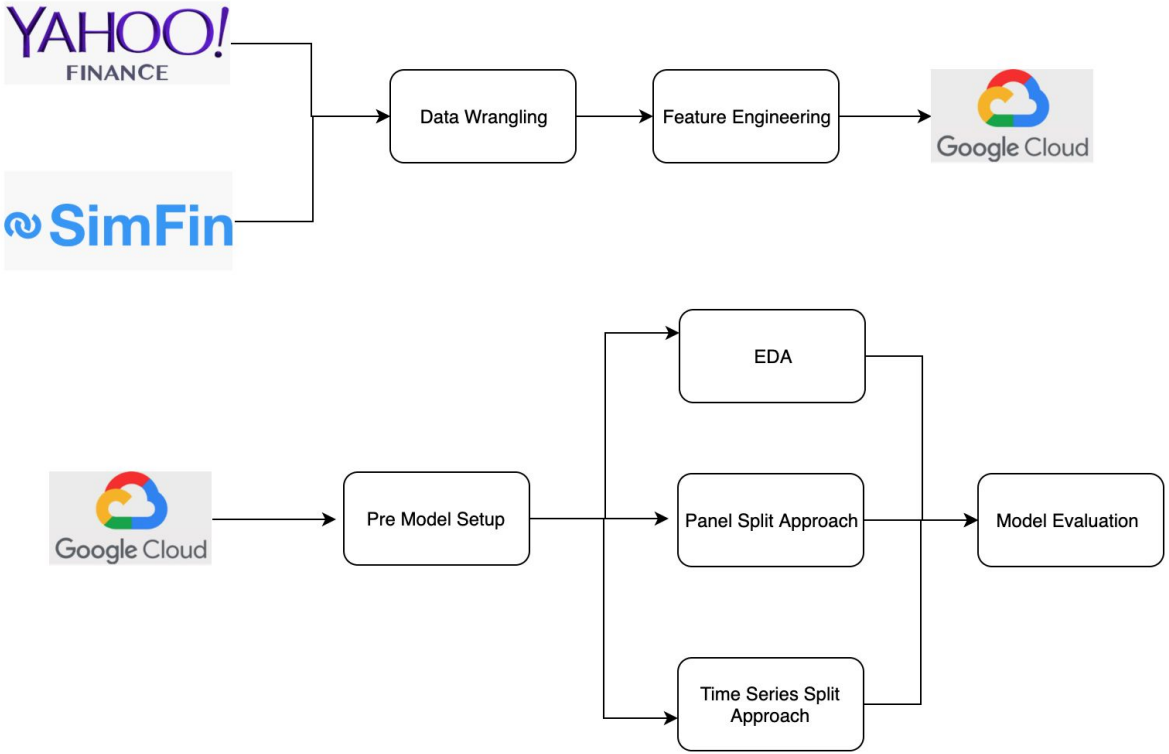
BACKGROUND

- Passive Fund Management
  - The goal is to match, not outperform an index
  - Low cost, efficient
- Active Fund Management
  - The goal is to outperform the benchmark index
  - Higher relative costs; most aren't successful long term
- Why beating the market (S&P 500) is so difficult
- Our approach
  - Forgo predicting raw, absolute returns
  - Frame classification problem: will stock XYZ generate alpha (excess returns relative to the S&P 500)
  - Model driven to detect stocks most likely to generate excess returns

HIGH LEVEL ARCHITECTURE



SOURCING, STORAGE, AND WRANGLING



WRANGLING

```
    # Generate a daily frequency dataframe
    daily_dates = pd.date_range(start=qtr_yr_map['date'].min(),
                                end=qtr_yr_map['date'].max())
    daily_dates = [str(x)[:10] for x in daily_dates]
    daily_df_map = pd.DataFrame(daily_dates, columns=['date'])

    # Forward fill at the ticker-quarter level
    pop_tickers = qtrly_simfin['ticker'].drop_duplicates().tolist()

    daily_dfs = []
    for t in pop_tickers:
        daily_df = pd.merge(daily_df_map,
                            qtrly_simfin[qtrly_simfin['ticker'] == t],
                            how='left',
                            on='date').ffill()

        daily_dfs.append(daily_df)

    # Reduce by row concatenation
    daily_simfin = pd.concat(daily_dfs).reset_index().drop('index', axis=1)
```

- Final post 2011:Q1 dataset contains 1,010,378 unique ticker-date instances as of COB 06/12/2019
- Daily ingestion implies growth over time – we're working with live data

TARGET GENERATION

- Due to time-series considerations, proper calculation of our target is critical
- Returns are calculated as:

$$return_{t+n,i} = \frac{AdjClose_{t+n,i} - AdjClose_{t,i}}{AdjClose_{t,i}}$$

- Relative returns, as:

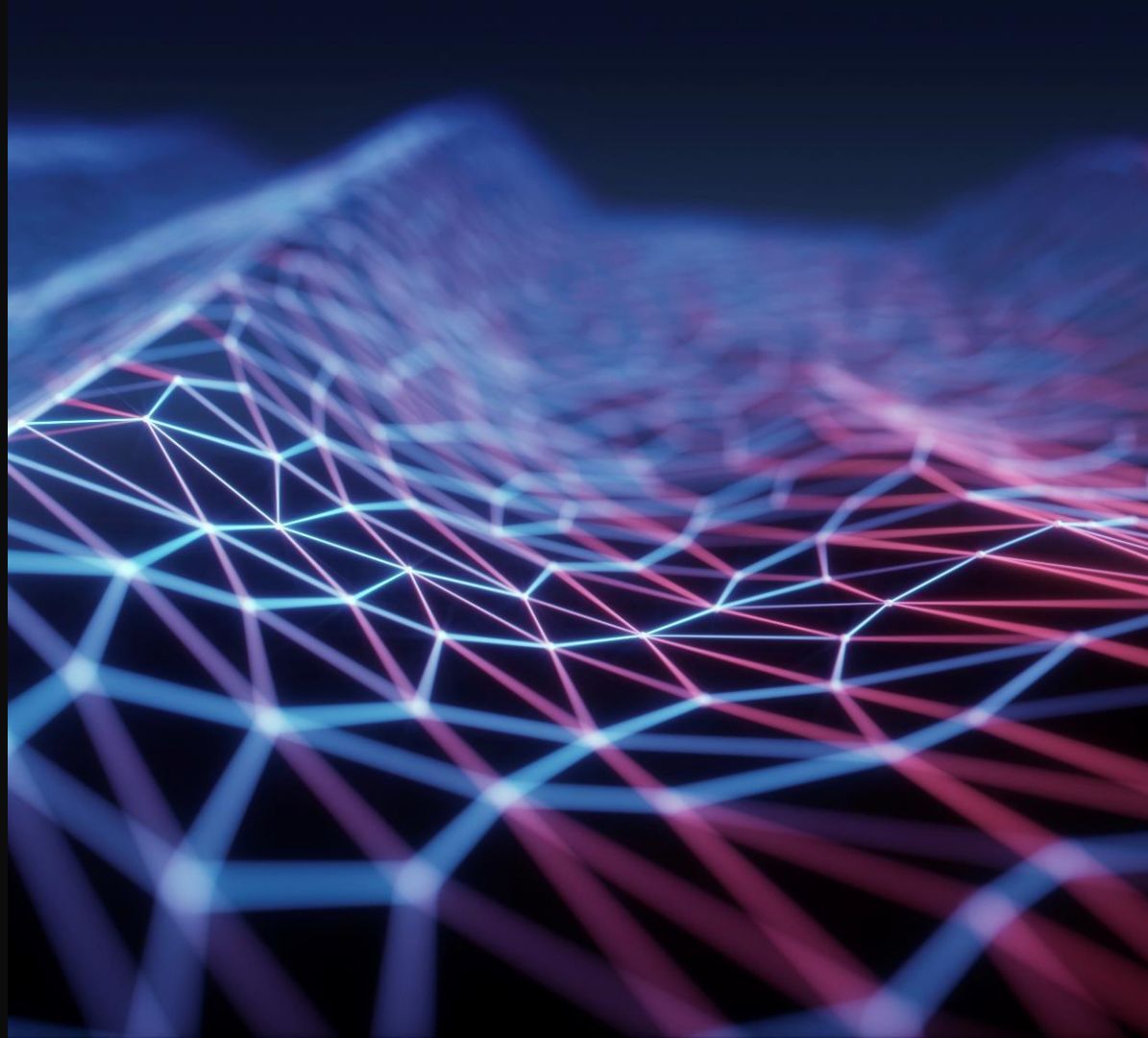
$$relative\_return_{t+n,i} = return_{t+n,i} - return_{(t+n,S\&P\ 500)}$$

- Binary targets, as:

$$target_{t,i} = \begin{cases} 1\ if\ relative\_return_{t+n,i} > 0 \\ 0\ if\ relative\_return_{t+n,i} \leq 0 \end{cases}$$

TARGET GENERATION

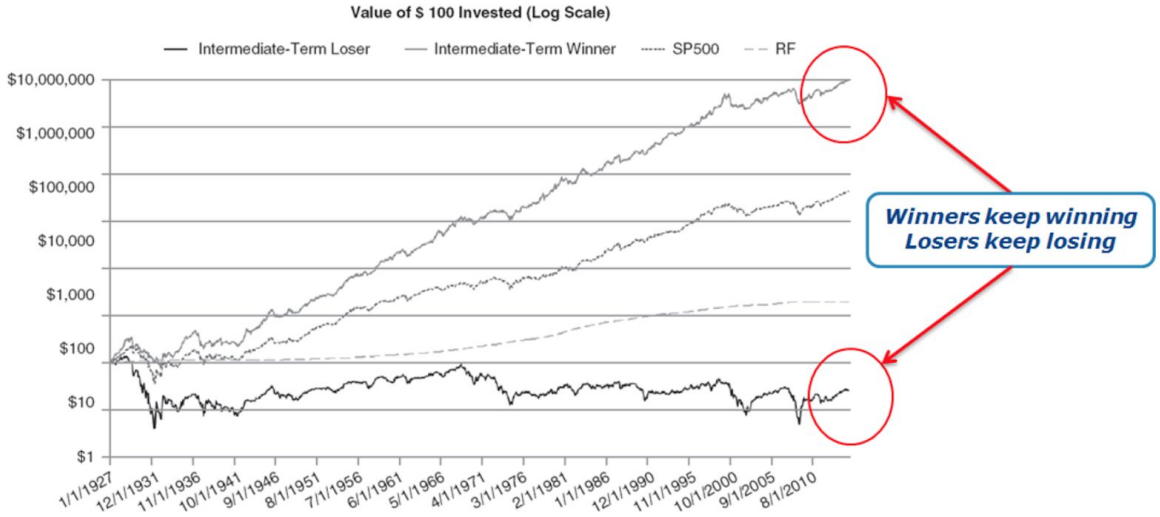
FEATURE ENGINEERING



- Stock Momentum Theory
  - Tendency of strong returning stocks continue to perform well while weak returning stocks continue underperformance
  - Basic momentum vs. investing in the S&P 500
- Momentum Portfolio Construction
  - Buy the best returning stocks over a given period
  - Rebalance regularly

FEATURE ENGINEERING

FEATURE ENGINEERING



https://alphaarchitect.com/2015/12/01/quantitative-momentum-investing-philosop

FEATURE ENGINEERING

- Generate stock price returns

```
def get_price_returns(time_series_df, ticker):
    close = pd.DataFrame(basic.get_time_series_adjusted_close(time_series_df, ticker))
    time_series_df.loc[ticker, 'Pct_Change_Daily'] = close.pct_change(1).values
    time_series_df.loc[ticker, 'Pct_Change_Monthly'] = close.pct_change(basic.MONTHLY_TRADING_DAYS).values
    time_series_df.loc[ticker, 'Pct_Change_Yearly'] = close.pct_change(basic.YEARLY_TRADING_DAYS).values
    return time_series_df
```

- For each time *t* in the dataset (representing a trading day)
  - Sort percent change for all stocks in the S&P 500
  - Each stock is assigned a return rank value – simply the index of the sorted list

FEATURE ENGINEERING

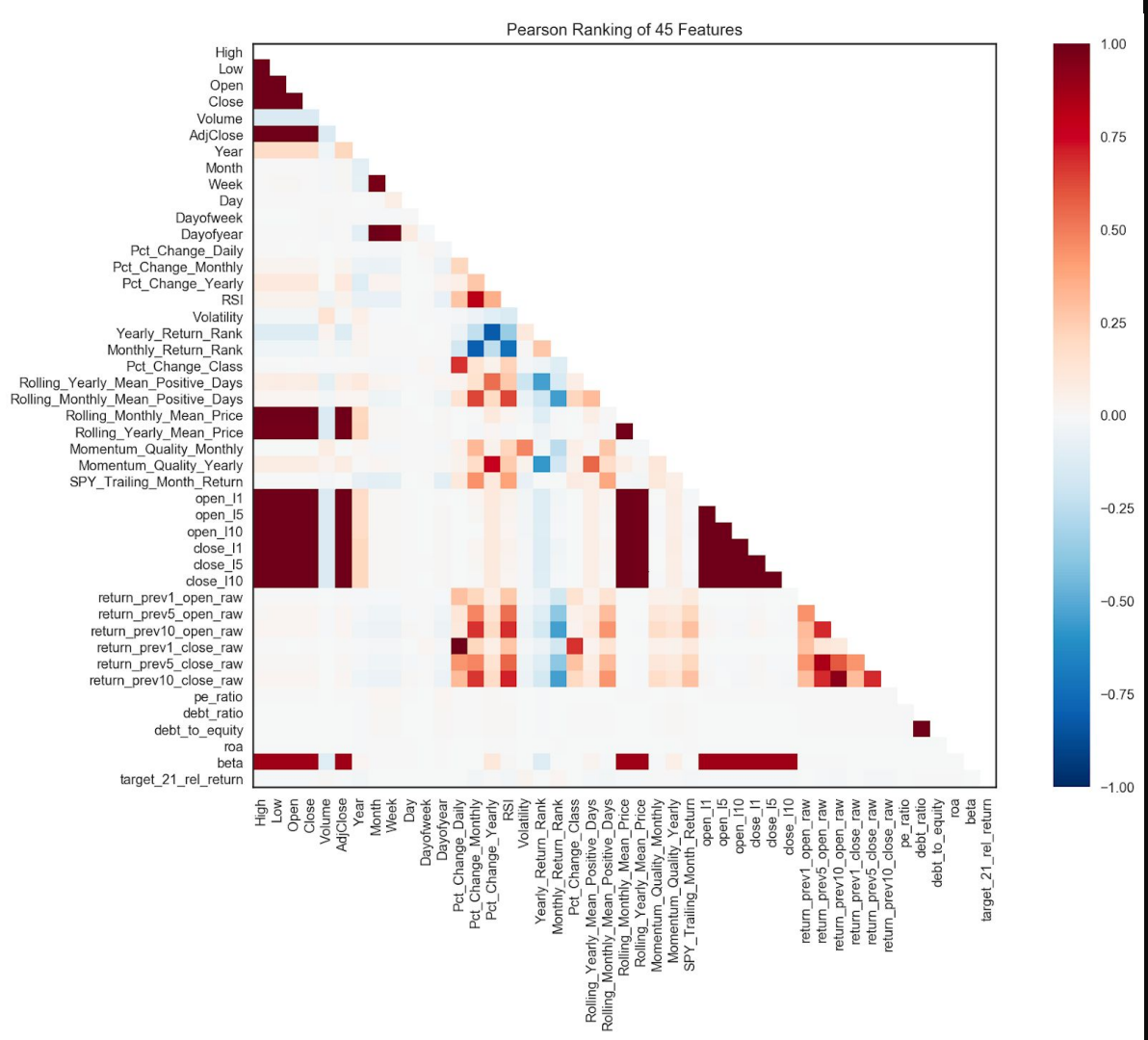
- Momentum Quality
- Rolling Moving Averages
- Volatility—standard deviation of returns
- Relative Strength Index (RSI) — signal potential oversold/overbought conditions
- SPY Return — Signal systematic market conditions
- Earnings Per Share (EPS)
- P/E Ratio — (EPS) / (Stock_Price)
- Return on Assets (ROA)
- Debt-to-Equity
- Beta — Stock sensitivity to S&P 500 changes

EXPLORATORY DATA ANALYSIS
&
FEATURE SELECTION

We perform EDA to :

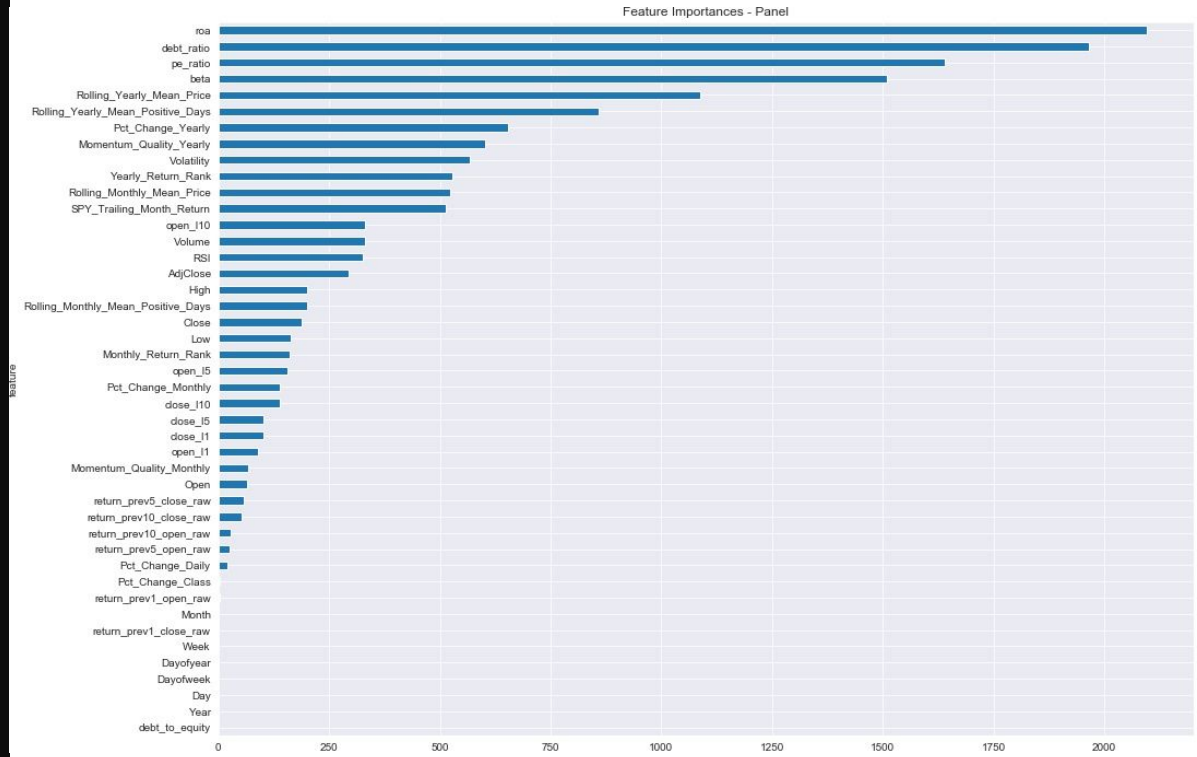
- Explore features' relationships and detect collinearity
- Test how our features are doing in panel and ticker-level hypothesis, the two completely different approaches we are about to take
- Make sound decisions on feature dropping and eventually feature selection

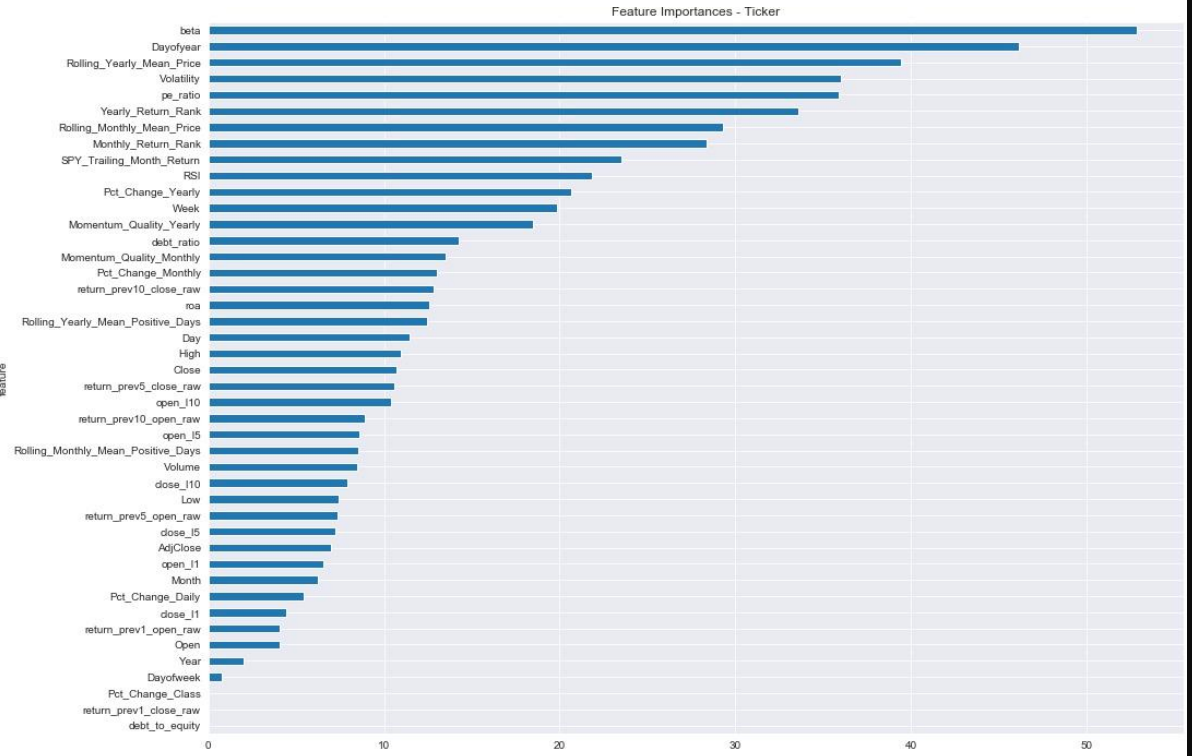
EDA



FEATURE CORRELATION

FEATURE SELECTION - PANEL-LEVEL





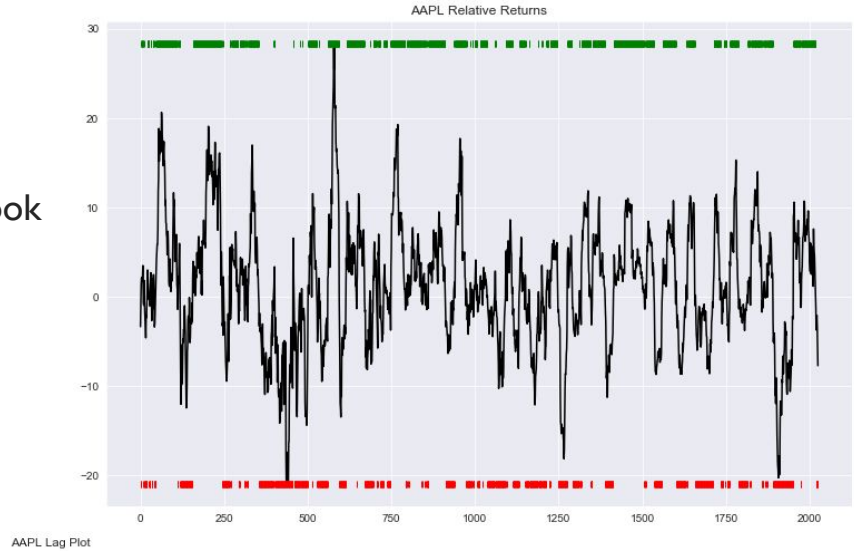
FEATURE SELECTION - TICKER-LEVEL

FINDINGS

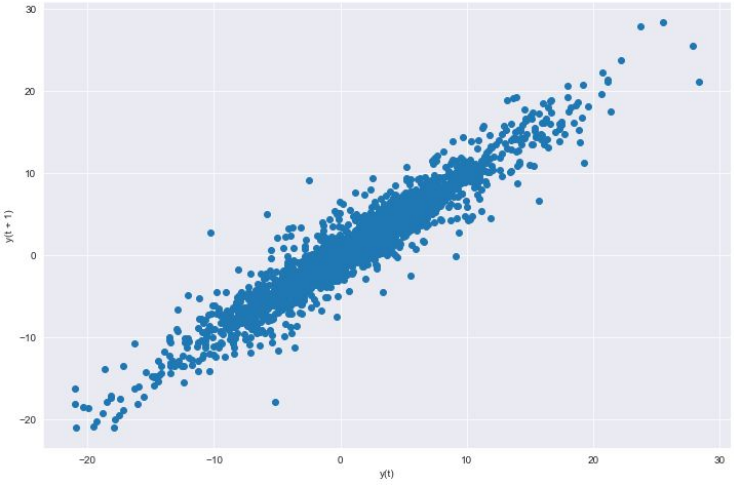
- Engineered features top for both panel and ticker-level approach
- Ticker-level importances are of a mixed bag because the importance of features differs across individual entities → difficulty in modeling the panel features (important features might be pushed out)
- Multicollinearity in engineered features → final selection of 27 features (engineered in their majority)

TARGET PROJECTION

What do returns look like?



AAPL Lag Plot



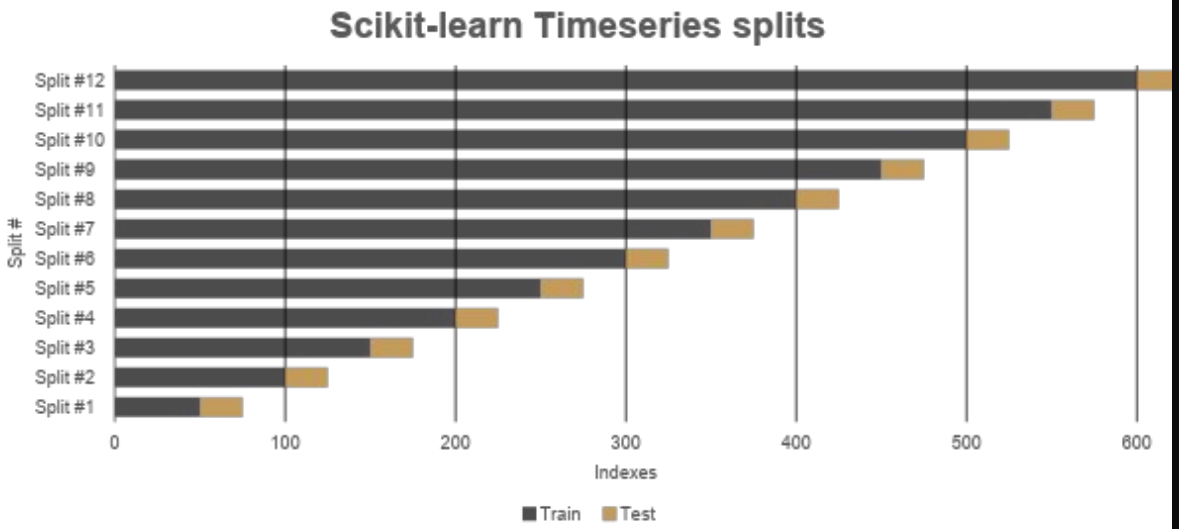
What does the lag structure look like?

CROSS VALIDATION FRAMEWORKS

Scikit-Learn TimeSeries Splits

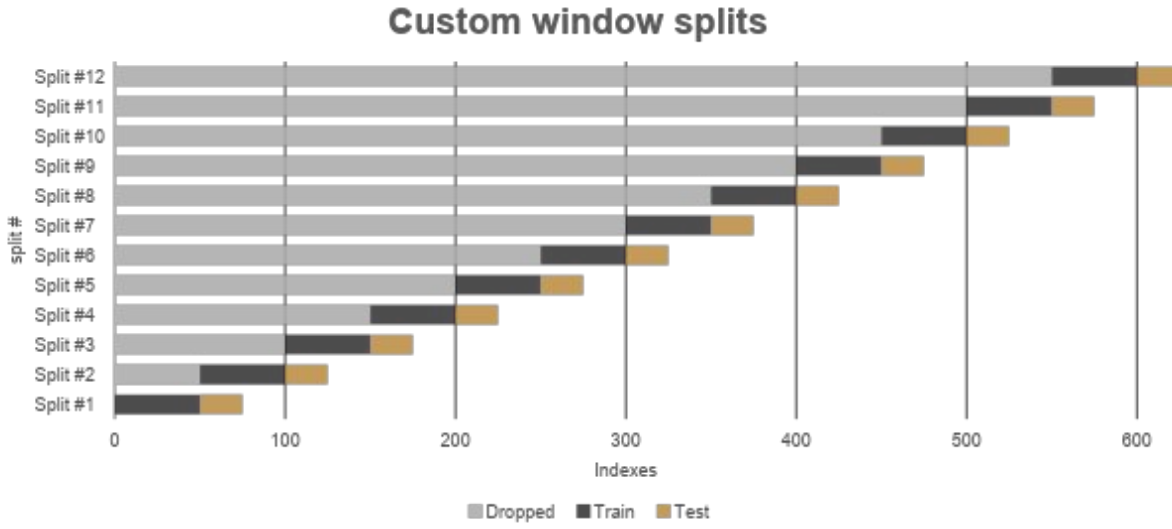
Custom Panel Splits

Custom Window Splits



CROSS VALIDATION

CROSS VALIDATION



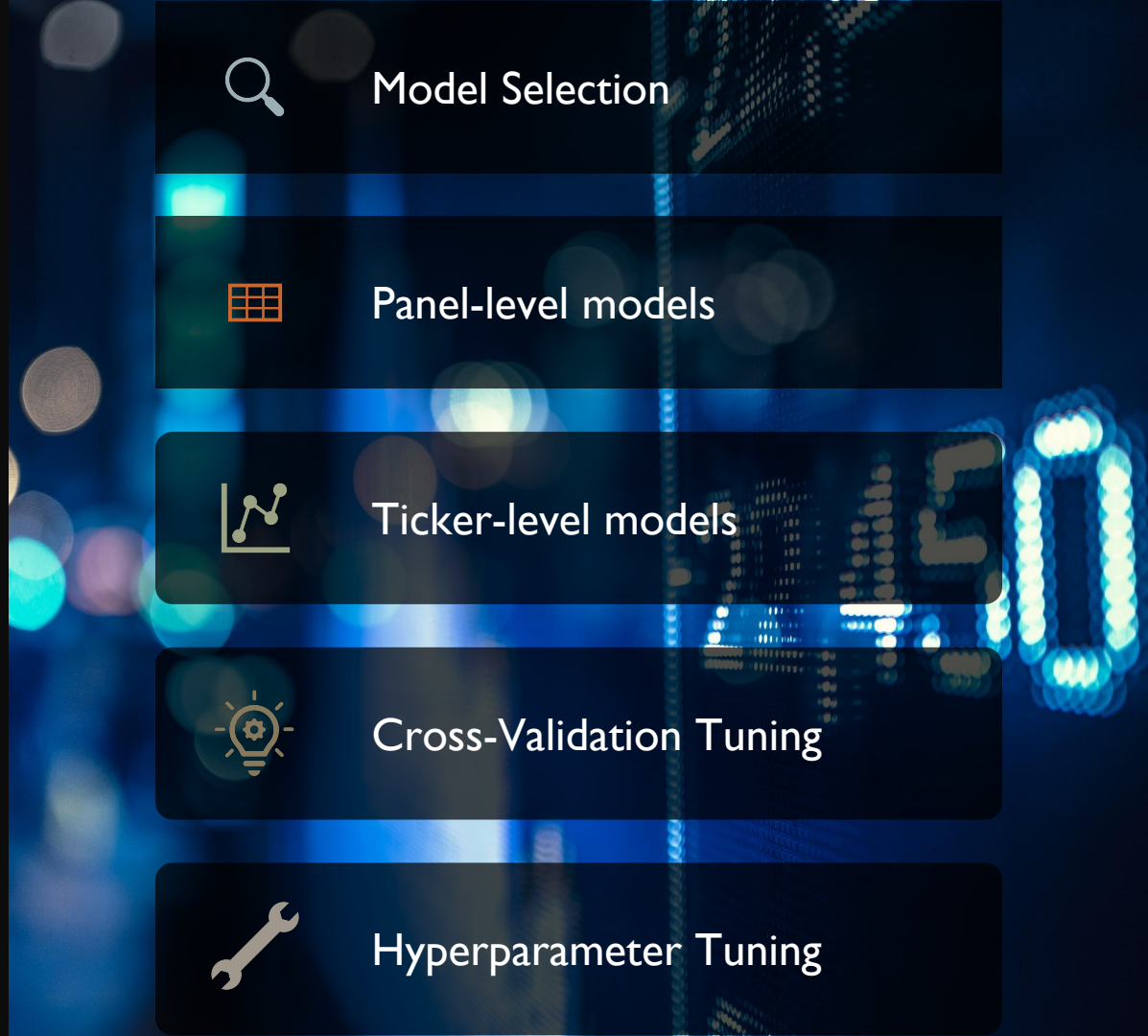

```python
def n_day_ahead_split(indexer, train=252, test=21, window=False):
    """
    Function to generate time series splits of a panel or time series, provided
    a dedicated minimum for the training sample and a dedicated testing window.
    Default is to use a minimum of a year's worth of data with the month ahead
    horizon for testing consistent with most constructed targets.
    Returns a generator object for compliance with sci-kit learn API.
    """
    buffer = indexer % test
    end = train + buffer
    start = 0
    while end < indexer:
        train_indices = np.arange(start, end).tolist()
        test_indices = np.arange(end, (end+test)).tolist()
        end += test
        if window:
            start += test

        yield train_indices, test_indices
```

- Panel splits are calculated similarly with one major exception
  - Scikit-Learn TimeSeries splits are entity-agnostic – the first split contains the first n-observations
  - Panel data reflects different entities at different points in time
  - Solution: program splits such that each entity is present in each time-series split
  - Same time-series folds, but each fold contains each ticker at those points in time

```python
def panel_split(n_folds, groups, grouping_var='date_of_transaction'):
    """
    Function to generate time series splits of a panel, provided
    a number of folds, and an indexable dataframe to create groups.
    Returns a generator object for compliance with sci-kit learn API.
    """
    date_idx = (groups[[grouping_var]]
                .drop_duplicates()
                .sort_values(grouping_var)
                .reset_index()
                .rename({'index':'tsidx'}, axis=1))

    by_ticker_index = groups.reset_index().rename({'index':'panel_index'}, axis=1)
    by_ticker_index = (pd.merge(by_ticker_index, date_idx, on=grouping_var)
                       .sort_values('panel_index')
                       .set_index('panel_index'))

    ticker_range = sorted(by_ticker_index['tsidx'].unique().tolist())

    splits = TimeSeriesSplit(n_splits=n_folds)

    for train_indices, test_indices in splits.split(ticker_range):
        panel_train_indices = (by_ticker_index[by_ticker_index['tsidx'].isin(train_indices)]
                               .index
                               .tolist())
        panel_test_indices = (by_ticker_index[by_ticker_index['tsidx'].isin(test_indices)]
                              .index
                              .tolist())
        yield panel_train_indices, panel_test_indices
```

CROSS VALIDATION

MODELING


Model Selection
Panel-level models
Ticker-level models
Cross-Validation Tuning
Hyperparameter Tuning

MODELING

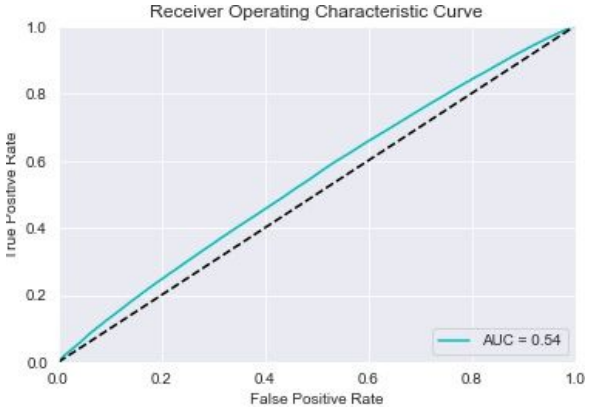
- Performed both panel-level and ticker-level modeling
  - Panel-level accounts for the trajectory of related entities (industry-level effects)
  - Ticker-level focuses explicitly on the features/developments relevant for that entity
- First pass: logistic regression, k-nearest neighbors, random forest, gradient-boosting classifiers, and the LightGBM implementation
- Focused predominantly on LightGBM (accuracy, speed, and tunability)

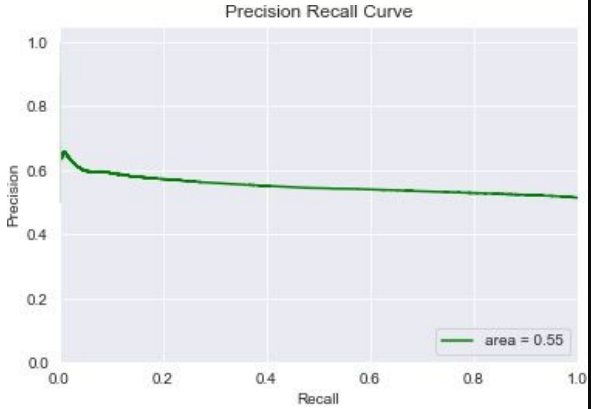
MODELING

- First-pass, panel-level results are disappointing
- Out-of-sample AUC and F1-scores reliably around 0.50 for panel-level, panel validation splits for all models
  - Marginal improvements in out-of-sample AUC scores for panel-level, *time-series* validation splits
  - LightGBM outperform off-the-shelf Scikit-Learn models, marginally, but reliably

MODEL EVALUATION PANEL LEVEL

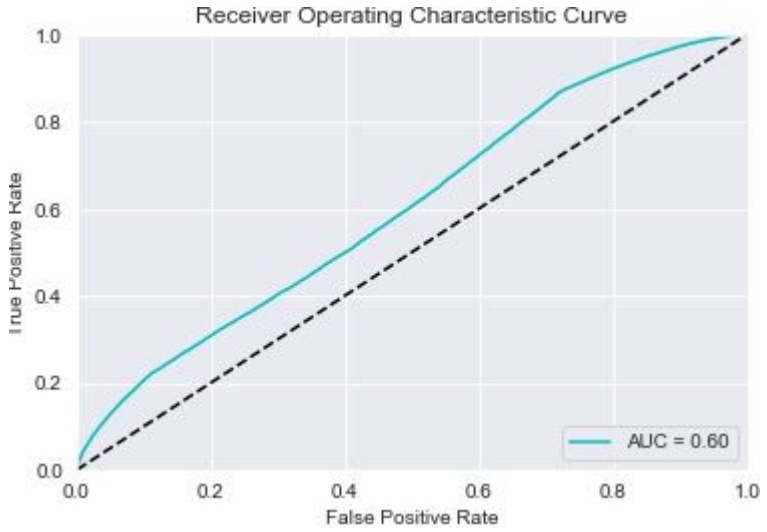
- Panel-level regressions bear little fruit
- Little lift in AUC/PR-curves - classification report a wash of 0.5s
- Time-series splits outperform panel-splits
- A model agnostic about entities is better – little, valuable information about each individual unit
- Dynamic threshold search is generally unhelpful





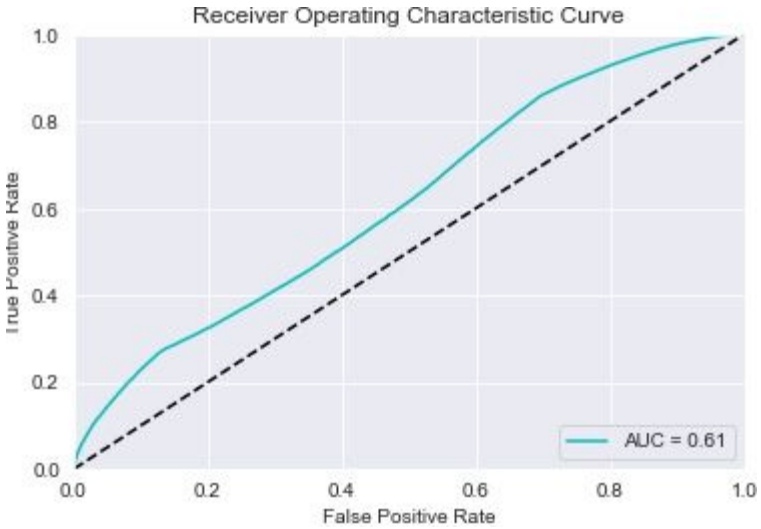
MODEL EVALUATION TICKER LEVEL

- At the ticker-level, non-trivial improvements in out-of-fold performance
  - Off-the-shelf Scikit-Learn TimeSeries splits, 12-folds (~160 obs. out-of-fold)
  - Average AUC across all 486 individual tickers of 0.60
  - Weighted-average F1-scores of 0.57 (0.58 and 0.60 for positive class precision and F1-scores)
  - Robust even in small (10-50) samples of tickers



MODEL EVALUATION TICKER LEVEL

- Using our custom-built 252 day (1-year) rolling-window training set and 121 day (6-months) validation set
  - ~14 folds
  - Modest improvement in evaluation metrics
  - Information too far in the past is unhelpful

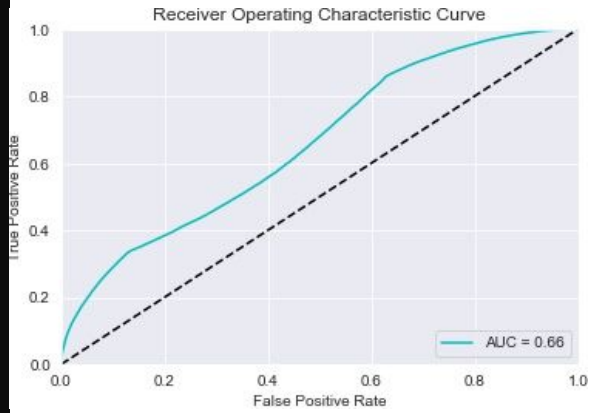


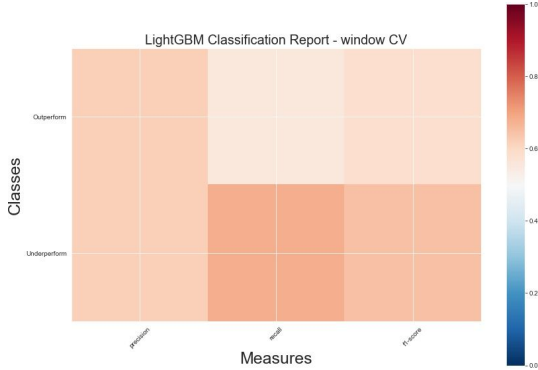
MODEL EVALUATION AND TUNING

- Cross-Validation
  - Data too far in the *past* does not help - small deterioration >2 years out
  - By that logic, predicting a target too far in the *future* may be unrealistic
  - 21-day validation fold reliably performs well, with AUC scores around 0.70
    - Smallest realistic set
    - May lead to overfitting
  - 42-day (2-months), 63-day (3-months), and 84-day (4-month) also explored
- Hyperparameter Tuning
  - More estimators (10,000+), better models
  - Models perform *extremely* well in-fold, symptomatic the LightGBM algorithm
  - Short (depth 2) trees with few leaves and bins (25) to address overfitting

FINAL MODEL

- Quarter-ahead rolling window splits yield reasonable performance
  - Training window of 252 days (one year) and a test window of 63 days (one quarter) - roll through and record performance out-of-fold
  - AUC curve across the basket of 0.66 (well beyond the panel-level coin-flip)
  - Weighted-average F1-scores of 0.58
    - 0.56 and 0.57 precision and F1-scores for the positive class
    - Scores improve to 0.62, 0.62, and 0.65, respectively, using threshold search
    - Final accuracy of about 62%





APPLICATION

APPLICATION

- Applying model to a theoretical portfolio
- Begin with a passive style fund that tracks S&P 500 returns
- Group high and low conviction stocks based on model probability scores at the ticker level
- Reduce portfolio allocation to the low conviction stocks and in turn increase position in high conviction
- Rebalance monthly

NEXT STEPS

- Features that capture the emotion and geopolitical narratives
- More intensive, purpose-built modeling (LSTM, leverage GPUs)
- Address window cross-validation split optimization problem
- Application development: historic returns, predicted performance, host with all tickers, top rankings
- Map to real returns and backtest – what returns does this provide the individual investor

THANK YOU

GLOSSARY

- **Passive Fund Management** — Portfolio of assets intended to track a benchmark index
- **Active Fund Management** — Any portfolio constructed with the intent to outperform a benchmark index
- **S&P 500** — Index of 500 U.S. stocks based on market cap and is commonly used as a representation of the stock market as a whole
- **Volatility**—standard deviation of returns
- **Momentum** — Price Return over a given time period
- **Momentum Quality** — Metric dependent on price return and price path smoothness
- **Momentum Effect** — The tendency of high returning stocks to continue outperforming while weak returning stocks underperform
- **Relative Strength Index** (RSI) — signal potential oversold/overbought conditions
- **SPY Return** — Signal systematic market conditions
- **Earnings Per Share** (EPS)
- **P/E Ratio** — (EPS) / (Stock_Price)
- **Return on Assets** (ROA)
- **Debt-to-Equity** — Measures degree financial leverage
- **Beta** — Stock sensitivity to S&P 500 changes