Imports & Settings

In [1]:

```
# Core tools
import joblib
import pandas as pd

# Company info
import yfinance as yf

# Dash
from dash.dependencies import Input, Output

# Explainer Dashboard
from explainerdashboard.custom import *
from explainerdashboard import RegressionExplainer, ExplainerDashboard
```

Setting Ticker

Until functionality is built directly into the dashboard for switching between companies, the ticker to be analyzed will need to be manually specified here.

In [66]:

```
ticker = 'AAPL'
```

Loading Data

S&P 500

In [67]:

```
snp = pd.read_csv('data/sp500.csv')
snp.head()
```

Out[67]:

	Symbol	Security	SEC filings	GICS Sector	GICS Sub-Industry	Headquarters Location	Date first added	CIK	Founded
0	MMM	3M	reports	Industrials	Industrial Conglomerates	Saint Paul, Minnesota	1976-08-09	66740	1902
1	ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	1800	1888
2	ABBV	AbbVie	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1551152	2013 (1888)
3	ABMD	Abiomed	reports	Health Care	Health Care Equipment	Danvers, Massachusetts	2018-05-31	815094	1981
4	ACN	Accenture	reports	Information Technology	IT Consulting & Other Services	Dublin, Ireland	2011-07-06	1467373	1989

Company Info

In [68]:

```
info = yf.Ticker(ticker).info
info.keys()
```

Out[68]:

```
dict_keys(['zip', 'sector', 'fullTimeEmployees', 'longBusinessSummary', 'city', 'phone', 'state', 'country', 'companyOfficers', 'webs
ite', 'maxAge', 'address1', 'industry', 'ebitdaMargins', 'profitMargins', 'grossMargins', 'operatingCashflow', 'revenueGrowth', 'oper
atingMargins', 'ebitda', 'targetLowPrice', 'recommendationKey', 'grossProfits', 'freeCashflow', 'targetMedianPrice', 'currentPrice',
'earningsGrowth', 'currentRatio', 'returnOnAssets', 'numberOfAnalystOpinions', 'targetMeanPrice', 'debtToEquity', 'returnOnEquity',
'targetHighPrice', 'totalCash', 'totalDebt', 'totalRevenue', 'totalCashPerShare', 'financialCurrency', 'revenuePerShare', 'quickRati
o', 'recommendationMean', 'exchange', 'shortName', 'longName', 'exchangeTimezoneName', 'exchangeTimezoneShortName', 'isEsgPopulated',
'gmtOffSetMilliseconds', 'quoteType', 'symbol', 'messageBoardId', 'market', 'annualHoldingsTurnover', 'enterpriseToRevenue', 'beta3Ye
ar', 'enterpriseToEbitda', '52WeekChange', 'morningStarRiskRating', 'forwardEps', 'revenueQuarterlyGrowth', 'sharesOutstanding', 'fun
dInceptionDate', 'annualReportExpenseRatio', 'totalAssets', 'bookValue', 'sharesShort', 'sharesPercentSharesOut', 'fundFamily', 'last
FiscalYearEnd', 'heldPercentInstitutions', 'netIncomeToCommon', 'trailingEps', 'lastDividendValue', 'SandP52WeekChange', 'priceToBoo
k', 'heldPercentInsiders', 'nextFiscalYearEnd', 'yield', 'mostRecentQuarter', 'shortRatio', 'sharesShortPreviousMonthDate', 'floatSha
res', 'beta', 'enterpriseValue', 'priceHint', 'threeYearAverageReturn', 'lastSplitDate', 'lastSplitFactor', 'legalType', 'lastDividen
dDate', 'morningStarOverallRating', 'earningsQuarterlyGrowth', 'priceToSalesTrailing12Months', 'dateShortInterest', 'pegRatio', 'ytdR
eturn', 'forwardPE', 'lastCapGain', 'shortPercentOfFloat', 'sharesShortPriorMonth', 'impliedSharesOutstanding', 'category', 'fiveYear
AverageReturn', 'previousClose', 'regularMarketOpen', 'twoHundredDayAverage', 'trailingAnnualDividendYield', 'payoutRatio', 'volume24
Hr', 'regularMarketDayHigh', 'navPrice', 'averageDailyVolume10Day', 'regularMarketPreviousClose', 'fiftyDayAverage', 'trailingAnnualD
ividendRate', 'open', 'toCurrency', 'averageVolume10days', 'expireDate', 'algorithm', 'dividendRate', 'exDividendDate', 'circulatingS
upply', 'startDate', 'regularMarketDayLow', 'currency', 'trailingPE', 'regularMarketVolume', 'lastMarket', 'maxSupply', 'openInteres
t', 'marketCap', 'volumeAllCurrencies', 'strikePrice', 'averageVolume', 'dayLow', 'ask', 'askSize', 'volume', 'fiftyTwoWeekHigh', 'fr
omCurrency', 'fiveYearAvgDividendYield', 'fiftyTwoWeekLow', 'bid', 'tradeable', 'dividendYield', 'bidSize', 'dayHigh', 'regularMarket
Price', 'logo_url'])
```

In [69]:

```
company = snp[snp.Symbol == ticker].Security.values[0]
sector = snp[snp.Symbol == ticker]['GICS Sector'].values[0]
industry = snp[snp.Symbol == ticker]['GICS Sub-Industry'].values[0]
hq = snp[snp.Symbol == ticker]['Headquarters Location'].values[0]
founded = snp[snp.Symbol == ticker].Founded.values[0]
marketcap = f"${info['marketCap']:,}"
website = info['website']
description = info['longBusinessSummary']
logo = info['logo_url']
```

Price History

In [70]:

```
prices = pd.read_csv(f'data/price_histories/{ticker}_history.csv', index_col='Date')
prices.head()
```

Out[70]:

	Open	High	Low	Close	Adj Close	Volume
Date						
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100600	469033600.0
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.095352	175884800.0
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.088353	105728000.0
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090540	86441600.0
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.093165	73449600.0

Prebuilt Model

In [71]:

```
data_full = pd.read_csv(f'data/preprocessed_data/{ticker}_preprocessed.csv')
data = data_full.drop(columns=['Report Date', 'Price Date', 'Open', 'High', 'Low', 'Adj Close', 'Volume'])
data.columns = [col.replace('.', '') for col in data.columns]
X = data.drop(columns=['Close'])
y = data.Close
```

In [72]:

```
model = joblib.load(f'models/{ticker}_model.joblib')
type(model)
```

Out[72]:

```
xgboost.sklearn.XGBRFRegressor
```

In [73]:

```
model
```

Out[73]:

```
XGBRFRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=0.2,
               colsample_bytree=0.2, gamma=25, gpu_id=-1,
               importance_type='gain', interaction_constraints='',
               learning_rate=0.5, max_delta_step=0, max_depth=4,
               min_child_weight=1, missing=nan, monotone_constraints='()',
               n_estimators=100, n_jobs=0, num_parallel_tree=100,
               objective='reg:squarederror', random_state=42, reg_alpha=0,
               scale_pos_weight=1, tree_method='exact', validate_parameters=1,
               verbosity=None)
```

Setting Explainer

In [74]:

```
explainer = RegressionExplainer(model, X, y, shap='tree')
```

```
Generating self.shap_explainer = shap.TreeExplainer(model)
```

Customizing Layout

Overview Tab

In [75]:

```
class OverviewTab(ExplainerComponent):
    def __init__(self, explainer, name=None, **kwargs):
        super().__init__(explainer, title='Overview')

    def layout(self):
        return dbc.Container(
            children=[
                dbc.Row([
                    dbc.Col([
                        html.Div([
                            html.Img(
                                src=logo,
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem'}
                            ),
                            html.H1(
                                children=html.B(f'{company} ({ticker})'),
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem',
                                       'marginLeft': '20px'}
                            )
                        ]),
                        html.Hr(),
                        html.Table(
                            children=[
                                html.Tr([
                                    html.Td(html.B('Sector:')),
                                    html.Td(sector),
                                    html.Td(html.B('Headquarters:')),
                                    html.Td(hq),
                                    html.Td(html.B('Market Cap:')),
                                    html.Td(marketcap)
                                ]),
                                html.Tr([
                                    html.Td(html.B('Industry:')),
                                    html.Td(industry),
                                    html.Td(html.B('Founded:')),
                                    html.Td(founded),
                                    html.Td(html.B('Website:')),
                                    html.Td(html.A(website, href=website, target='_blank'))
                                ])
                            ],
                            style={'border-collapse': 'separate',
                                   'border-spacing': '20px 5px',
                                   'margin-left': '-20px'}
                        ),
                        html.Br()
                    ])
                ]),
                dbc.Row([
                    dbc.Col([
                        html.H4(html.B('Description')),
                        html.P(description),
                        html.Br()
                    ])
                ]),
                dbc.Row([
                    dbc.Col([
                        html.H4(html.B('Price History')),
                        dcc.Graph(
                            id='price_history_graph',
                            figure={
                                'data': [{
                                    'x': prices.index,
                                    'y': prices.Close,
                                    'type': 'line'
                                }],
                                'layout': {
                                    'margin': dict(t=10),
                                    'xaxis': dict(title='Year'),
                                    'yaxis': dict(title='Price per Share',
                                                  tickprefix='$')
                                }
                            },
                            config={'displayModeBar': False}
                        )
                    ])
                ])
            ],
            style={'marginTop': '25px'}
        )
```

Features Tab

In [76]:

```python
class FeaturesTab(ExplainerComponent):
    def __init__(self, explainer, name=None, **kwargs):
        super().__init__(explainer, title='Features')

    def component_callbacks(self, app, **kwargs):
        @app.callback(
            Output('feature_graph', 'figure'),
            Input('feature_explorer_dropdown', 'value')
        )
        def update_graph(value):
            return {
                'data': [{
                    'x': data_full['Report Date'],
                    'y': data_full[value],
                    'type': 'bar'
                }],
                'layout': {
                    'margin': dict(t=25),
                    'xaxis': dict(title='Year'),
                    'yaxis': dict(title='Value (USD)',
                                  tickprefix='$')
                }
            }

    def layout(self):
        return dbc.Container(
            children=[
                dbc.Row([
                    dbc.Col([
                        html.Div([
                            html.Img(
                                src=logo,
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem'}
                            ),
                            html.H1(
                                children=html.B(f'{company} ({ticker})'),
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem',
                                       'marginLeft': '20px'}
                            )
                        ]),
                        html.Hr(),
                    ])
                ]),
                dbc.Row([
                    dbc.Col([html.H4(html.B('Feature Explorer'))]),
                    dbc.Col([
                        dcc.Dropdown(id='feature_explorer_dropdown',
                                     options=[{'label': col, 'value': col} for col in X.columns],
                                     value='Total Assets',
                                     multi=False)
                    ])
                ]),
                dbc.Row([
                    dbc.Col([
                        dcc.Graph(id='feature_graph',
                                  config={'displayModeBar': False})
                    ])
                ])
            ],
            style={'marginTop': '25px'}
        )
```

SHAP Tab

In [77]:

```
class SHAPTab(ExplainerComponent):
    def __init__(self, explainer, name=None, **kwargs):
        super().__init__(explainer, title='SHAP Analysis')

        self.feat_imps = ImportancesComponent(
                            explainer,
                            depth=15,
                            no_permutations=True,
                            hide_popout=True
                        )

        self.depend = ShapDependenceComponent(
                            explainer,
                            hide_popout=True,
                            hide_outliers=True,
                            hide_footer=True,
                            hide_index=True
                        )

        self.interaction = InteractionSummaryComponent(
                            explainer,
                            hide_popout=True,
                            hide_type=True,
                            depth=10
                        )

        self.inter_plot = InteractionDependenceComponent(
                            explainer,
                            hide_popout=True,
                            hide_index=True,
                        )

    def layout(self):
        return dbc.Container(
            children=[
                dbc.Row([
                    dbc.Col([
                        html.Div([
                            html.Img(
                                src=logo,
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem'}
                            ),
                            html.H1(
                                children=html.B(f'{company} ({ticker})'),
                                style={'display': 'inline',
                                       'vertical-align': 'middle',
                                       'height': '3rem',
                                       'width': '3rem',
                                       'marginLeft': '20px'}
                            )
                        ]),
                        html.Hr(),
                    ])
                ]),
                dbc.Row([dbc.Col([self.feat_imps.layout()])]),
                html.Br(),
                dbc.Row([dbc.Col([self.depend.layout()])]),
                html.Br(),
                dbc.Row([dbc.Col([self.interaction.layout()])]),
                html.Br(),
                dbc.Row([dbc.Col([self.inter_plot.layout()])]),
                html.Br()
            ],
            style={'marginTop': '25px'}
        )
```

Running the Dashboard

In [78]:

```
db = ExplainerDashboard(explainer,
                        tabs=[OverviewTab, FeaturesTab, SHAPTab],
                        title='Stock KPIs',
                        description='',
                        bootstrap=dbc.themes.SANDSTONE,
                        fluid=False,
                        header_hide_download=True,
                        hide_poweredby=True)

db.run(port=8050)
```

```
Building ExplainerDashboard..
Detected notebook environment, consider setting mode='external', mode='inline' or mode='jupyterlab' to keep the notebook interactive
while the dashboard is running...
Generating layout...
Calculating shap values...
Calculating dependencies...
Calculating shap interaction values...
Reminder: TreeShap computational complexity is O(TLD^2), where T is the number of trees, L is the maximum number of leaves in any tre
e and D the maximal depth of any tree. So reducing these will speed up the calculation.
Reminder: you can store the explainer (including calculated dependencies) with explainer.dump('explainer.joblib') and reload with e.
g. ClassifierExplainer.from_file('explainer.joblib')
Registering callbacks...
Starting ExplainerDashboard on http://74.129.178.98:8050
Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

Dash is running on http://0.0.0.0:8050/

 * Serving Flask app 'explainerdashboard.dashboards' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://74.129.178.98:8050/ (Press CTRL+C to quit)
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET / HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET /_dash-dependencies HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET /assets/favicon.ico?m=1630208771.8202608 HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET /_dash-layout HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET /_dash-component-suites/dash_core_components/async-graph.js HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 204 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 204 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "GET /_dash-component-suites/dash_core_components/async-plotlyjs.js HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:13] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:14] "GET /_dash-component-suites/dash_core_components/async-dropdown.js HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:17] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:20] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:23] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:27] "POST /_dash-update-component HTTP/1.1" 200 -
74.129.178.98 - - [07/Sep/2021 21:19:29] "POST /_dash-update-component HTTP/1.1" 200 -
```