modeling_roadmap

June 3, 2020

1 Modeling Primer

1.1 Setup

In this notebook we will try to create some Ordinary Least Squares (OLS) models to predict stock price movement. To prepare, there are several csv files in this directory that are downloaded from the FRED webiste - an excellent source for macroeconomic data. We will learn: * How to load data into Python from a csv or excel file using Pandas * Also how to use the yfinance package to pull in stock price data from Yahoo! finance * Explore the data with some visualizations using matplotlib, which is nicely integrated with pandas * Create a model using the statsmodels api * Various diagnostic techniques for timeseries modeling available to us via statsmodels and scikit-learn

1.1.1 Part 1: Pandas

Pandas has become the standard for dataframes in the Python world due to the easy power that it provides the user. We will go over some useful features as we use Pandas to expore our data and use it for regression, but I would encourage you to explore the site linked in the title of this section as well as this cheat sheet for other very useful methods.

[2]:

	DATE	GDPC1
0	1947-01-01	2033.061
1	1947-04-01	2027.639
2	1947-07-01	2023.452
3	1947-10-01	2055.103
4	1948-01-01	2086.017

[3]:

```
['DATE',
 'GDPC1',
 'T',
 '_AXIS_ALIASES',
 '_AXIS_IALIASES',
 '_AXIS_LEN',
 '_AXIS_NAMES',
 '_AXIS_NUMBERS',
 '_AXIS_ORDERS',
 '_AXIS_REVERSED',
```

```
'__abs__',
'__add__',
'__and__',
'__array__',
'__array_priority__',
'__array_wrap__',
'__bool__',
'__class__',
'__contains__',
'__copy__',
'__deepcopy__',
'__delattr__',
'__delitem__',
'__dict__',
'__dir__',
'__div__',
'__doc__',
'__eq__',
'__finalize__',
'__floordiv__',
'__format__',
'__ge__',
'__getattr__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__invert__',
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__matmul__',
'__mod__',
```

```
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdiv__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
'_accessors',
'_add_numeric_operations',
'_add_series_only_operations',
'_add_series_or_dataframe_operations',
'_agg_by_level',
'_agg_examples_doc',
'_agg_summary_and_see_also_doc',
'_aggregate',
'_aggregate_multiple_funcs',
'_align_frame',
'_align_series',
'_box_col_values',
```

```
'_box_item_values',
'_builtin_table',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_percentile',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_combine_const',
'_combine_frame',
'_combine_match_columns',
'_combine_match_index',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_axes_dict_from',
'_construct_axes_from_arguments',
'_constructor',
'_constructor_expanddim',
'_constructor_sliced',
'_convert',
'_count_level',
'_create_indexer',
'_cython_table',
'_data',
'_deprecations',
'_dir_additions',
'_dir_deletions',
'_drop_axis',
'_drop_labels_or_levels',
'_ensure_valid_index',
'_find_valid_index',
'_from_arrays',
'_from_axes',
'_get_agg_axis',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cacher',
'_get_index_resolvers',
'_get_item_cache',
'_get_label_or_level_values',
```

```
'_get_numeric_data',
'_get_space_character_free_column_resolvers',
'_get_value',
'_get_values',
'_getitem_bool_array',
'_getitem_frame',
'_getitem_multilevel',
'_gotitem',
'_iget_item_cache',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_info_repr',
'_init_mgr',
'_internal_get_values',
'_internal_names',
'_internal_names_set',
'_is_builtin_func',
'_is_cached',
'_is_copy',
'_is_cython_func',
'_is_datelike_mixed_type',
'_is_homogeneous_type',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_numeric_mixed_type',
'_is_view',
'_ix',
'_ixs',
'_join_compat',
'_maybe_cache_changed',
'_maybe_update_cacher',
'_metadata',
'_needs_reindex_multi',
'_obj_with_exclusions',
'_protect_consolidate',
'_reduce',
'_reindex_axes',
'_reindex_columns',
'_reindex_index',
'_reindex_multi',
'_reindex_with_indexers',
'_repr_data_resource_',
'_repr_fits_horizontal_',
```

```
'_repr_fits_vertical_',
'_repr_html_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_sanitize_column',
'_selected_obj',
'_selection',
'_selection_list',
'_selection_name',
'_series',
'_set_as_cached',
'_set_axis',
'_set_axis_name',
'_set_is_copy',
'_set_item',
'_set_value',
'_setitem_array',
'_setitem_frame',
'_setitem_slice',
'_setup_axes',
'_shallow_copy',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_to_dict_of_blocks',
'_try_aggregate_string_function',
'_typ',
'_unpickle_frame_compat',
'_unpickle_matrix_compat',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'_xs',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'append',
'apply',
```

```
'applymap',
'as_matrix',
'asfreq',
'asof',
'assign',
'astype',
'at',
'at_time',
'axes',
'between_time',
'bfill',
'bool',
'boxplot',
'clip',
'clip_lower',
'clip_upper',
'columns',
'combine',
'combine_first',
'compound',
'copy',
'corr',
'corrwith',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'eval',
'ewm',
'expanding',
'explode',
```

```
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'floordiv',
'from_dict',
'from_records',
'ge',
'get',
'get_dtype_counts',
'get_ftype_counts',
'get_values',
'groupby',
'gt',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
'index',
'infer_objects',
'info',
'insert',
'interpolate',
'isin',
'isna',
'isnull',
'items',
'iteritems',
'iterrows',
'itertuples',
'join',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lookup',
'lt',
'mad',
'mask',
'max',
'mean',
```

```
'median',
'melt',
'memory_usage',
'merge',
'min',
'mod',
'mode',
'mul',
'multiply',
'ndim',
'ne',
'nlargest',
'notna',
'notnull',
'nsmallest',
'nunique',
'pct_change',
'pipe',
'pivot',
'pivot_table',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'query',
'radd',
'rank',
'rdiv',
'reindex',
'reindex_like',
'rename',
'rename_axis',
'reorder_levels',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
```

```
'select_dtypes',
'sem',
'set_axis',
'set_index',
'shape',
'shift',
'size',
'skew',
'slice_shift',
'sort_index',
'sort_values',
'sparse',
'squeeze',
'stack',
'std',
'style',
'sub',
'subtract',
'sum',
'swapaxes',
'swaplevel',
'tail',
'take',
'to_clipboard',
'to_csv',
'to_dense',
'to_dict',
'to_excel',
'to_feather',
'to_gbq',
'to_hdf',
'to_html',
'to_json',
'to_latex',
'to_msgpack',
'to_numpy',
'to_parquet',
'to_period',
'to_pickle',
'to_records',
'to_sparse',
'to_sql',
'to_stata',
'to_string',
'to_timestamp',
'to_xarray',
'transform',
```

```
 'transpose',
 'truediv',
 'truncate',
 'tshift',
 'tz_convert',
 'tz_localize',
 'unstack',
 'update',
 'values',
 'var',
 'where',
 'xs']
```

We note a few things about our GDP data frame as it currently stands that need fixed: * Bold indicates index(row names)/columns - it has a generic index (which starts at 0, as all python indexing does) but we prefer this to be the date column * We haven't seen yet - but our date column is actually strings of the date - we want to change this to a datetime format to avoid issues later on (none specifically, just best practice)

[4]:

```
               GDPC1
DATE
1947-01-01  2033.061
1947-04-01  2027.639
1947-07-01  2023.452
1947-10-01  2055.103
1948-01-01  2086.017
```

We're going to be loading a few data frames from csv and we will need to do these same steps for each of them. Whenever there's a repeated process in our data pipeline such as this, it is advisable to make it into a function so we can simplify our main script.

[6]:

```
            PCEC96
DATE
2002-01-01  8981.7
2002-02-01  9022.0
2002-03-01  9020.6
2002-04-01  9066.3
2002-05-01  9031.8
```

Note the above data for PCE was monthly while GDP was quarterly - we'll see one wawy to account for this later.

1.1.2 Part 2: Yfinance

Now let's look into loading stock prices, Microsoft is a fun case study! Note you can change the ticker inside the first line of the below cell to change our analysis to any ticker that you can find info for on Yahoo! Finance.

[7]:

```
{'zip': '98052',
 'sector': 'Technology',
 'fullTimeEmployees': 144000,
 'longBusinessSummary': 'Microsoft Corporation develops, licenses, and supports
software, services, devices, and solutions worldwide. Its Productivity and
Business Processes segment offers Office, Exchange, SharePoint, Microsoft Teams,
Office 365 Security and Compliance, and Skype for Business, as well as related
Client Access Licenses (CAL); Skype, Outlook.com, and OneDrive; LinkedIn that
includes Talent and marketing solutions, and subscriptions; and Dynamics 365, a
set of cloud-based and on-premises business solutions for small and medium
businesses, large organizations, and divisions of enterprises. Its Intelligent
Cloud segment licenses SQL and Windows Servers, Visual Studio, System Center,
and related CALs; GitHub that provides a collaboration platform and code hosting
service for developers; and Azure, a cloud platform. It also provides support
services and Microsoft consulting services to assist customers in developing,
deploying, and managing Microsoft server and desktop solutions; and training and
certification to developers and IT professionals on various Microsoft products.
Its More Personal Computing segment offers Windows OEM licensing and other non-
volume licensing of the Windows operating system; Windows Commercial, such as
volume licensing of the Windows operating system, Windows cloud services, and
other Windows commercial offerings; patent licensing; Windows Internet of
Things; and MSN advertising. It also provides Microsoft Surface, PC accessories,
and other intelligent devices; Gaming, including Xbox hardware, and Xbox
software and services; video games and third-party video game royalties; and
Search, including Bing and Microsoft advertising. It sells its products through
distributors and resellers; and directly through digital marketplaces, online
stores, and retail stores. It has strategic partnerships with Humana Inc.,
Nokia, Telkomsel, Swiss Re, Kubota Corporation, and FedEx Corp. The company was
founded in 1975 and is headquartered in Redmond, Washington.',
 'city': 'Redmond',
 'phone': '425-882-8080',
 'state': 'WA',
 'country': 'United States',
 'companyOfficers': [],
 'website': 'http://www.microsoft.com',
 'maxAge': 1,
 'address1': 'One Microsoft Way',
 'fax': '425-706-7329',
 'industry': 'Software-Infrastructure',
 'previousClose': 184.91,
 'regularMarketOpen': 184.815,
 'twoHundredDayAverage': 165.58823,
 'trailingAnnualDividendYield': 0.010221188,
 'payoutRatio': 0.3233,
 'volume24Hr': None,
 'regularMarketDayHigh': 185.93,
 'navPrice': None,
```

```
'averageDailyVolume10Day': 34153866,
'totalAssets': None,
'regularMarketPreviousClose': 184.91,
'fiftyDayAverage': 179.39085,
'trailingAnnualDividendRate': 1.89,
'open': 184.815,
'toCurrency': None,
'averageVolume10days': 34153866,
'expireDate': None,
'yield': None,
'algorithm': None,
'dividendRate': 2.04,
'exDividendDate': 1589932800,
'beta': 1.229326,
'circulatingSupply': None,
'startDate': None,
'regularMarketDayLow': 183.58,
'priceHint': 2,
'currency': 'USD',
'trailingPE': 34.973583,
'regularMarketVolume': 25838439,
'lastMarket': None,
'maxSupply': None,
'openInterest': None,
'marketCap': 1405666459648,
'volumeAllCurrencies': None,
'strikePrice': None,
'averageVolume': 51070178,
'priceToSalesTrailing12Months': 10.828311,
'dayLow': 183.58,
'ask': 184.6,
'ytdReturn': None,
'askSize': 900,
'volume': 25838439,
'fiftyTwoWeekHigh': 190.7,
'forwardPE': 29.84863,
'fromCurrency': None,
'fiveYearAvgDividendYield': 1.92,
'fiftyTwoWeekLow': 124.21,
'bid': 184.42,
'tradeable': False,
'dividendYield': 0.011,
'bidSize': 1100,
'dayHigh': 185.93,
'exchange': 'NMS',
'shortName': 'Microsoft Corporation',
'longName': 'Microsoft Corporation',
```

'exchangeTimezoneName': 'America/New_York',
'exchangeTimezoneShortName': 'EDT',
'isEsgPopulated': False,
'gmtOffSetMilliseconds': '-14400000',
'quoteType': 'EQUITY',
'symbol': 'MSFT',
'messageBoardId': 'finmb_21835',
'market': 'us_market',
'annualHoldingsTurnover': None,
'enterpriseToRevenue': 10.46,
'beta3Year': None,
'profitMargins': 0.31656,
'enterpriseToEbitda': 23.646,
'52WeekChange': 0.46952236,
'morningStarRiskRating': None,
'forwardEps': 6.21,
'revenueQuarterlyGrowth': None,
'sharesOutstanding': 7583439872,
'fundInceptionDate': None,
'annualReportExpenseRatio': None,
'bookValue': 13.893,
'sharesShort': 44236712,
'sharesPercentSharesOut': 0.0058,
'fundFamily': None,
'lastFiscalYearEnd': 1561852800,
'heldPercentInstitutions': 0.74093,
'netIncomeToCommon': 41094000640,
'trailingEps': 5.3,
'lastDividendValue': None,
'SandP52WeekChange': 0.09011209,
'priceToBook': 13.341971,
'heldPercentInsiders': 0.014249999,
'nextFiscalYearEnd': 1625011200,
'mostRecentQuarter': 1569801600,
'shortRatio': 1.12,
'sharesShortPreviousMonthDate': 1586908800,
'floatShares': 7517653920,
'enterpriseValue': 1357848772608,
'threeYearAverageReturn': None,
'lastSplitDate': 1045526400,
'lastSplitFactor': '2:1',
'legalType': None,
'morningStarOverallRating': None,
'earningsQuarterlyGrowth': 0.21,
'dateShortInterest': 1589500800,
'pegRatio': 2.14,
'lastCapGain': None,

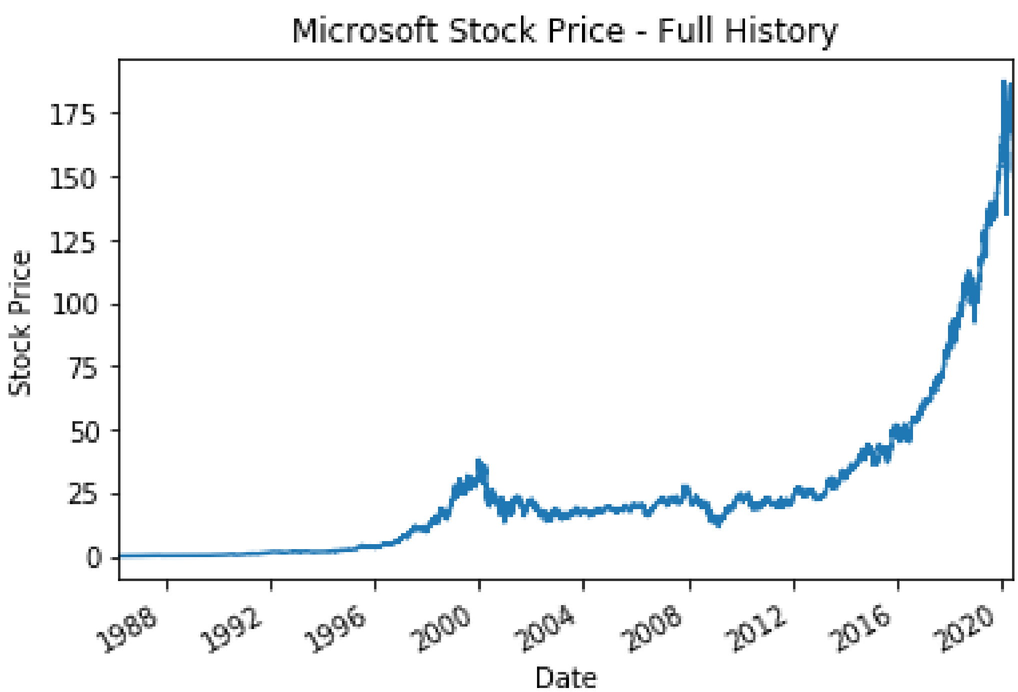
```
 'shortPercentOfFloat': 0.0058999998,
 'sharesShortPriorMonth': 53310482,
 'category': None,
 'fiveYearAverageReturn': None,
 'regularMarketPrice': 184.815,
 'logo_url': 'https://logo.clearbit.com/microsoft.com'}
```

[9]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1986-03-13	0.06	0.06	0.06	0.06	1031788800	0.0	0.0
1986-03-14	0.06	0.07	0.06	0.06	308160000	0.0	0.0
1986-03-17	0.06	0.07	0.06	0.07	133171200	0.0	0.0
1986-03-18	0.07	0.07	0.06	0.06	67766400	0.0	0.0
1986-03-19	0.06	0.06	0.06	0.06	47894400	0.0	0.0

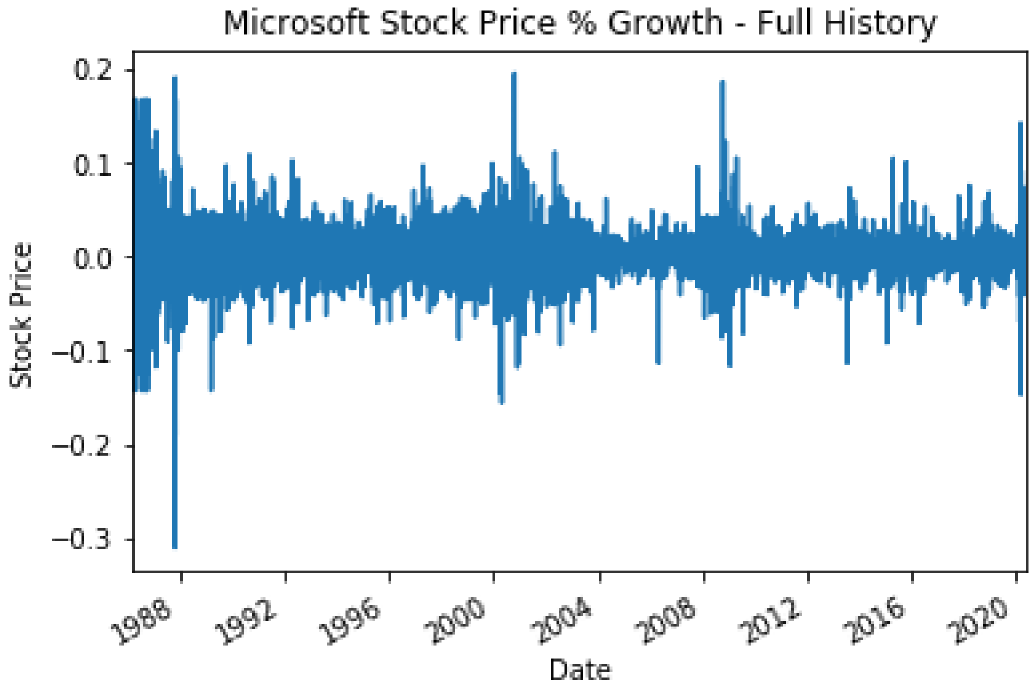
1.1.3 Part 3: Visualization with Matplotlib

Here we introduce matplotlib, which has been the dominant plotting engine in Python for some time. Their website, linked above, has a great catalog of sample graphs and is a great place to start when using a certain type of graph for the first time.

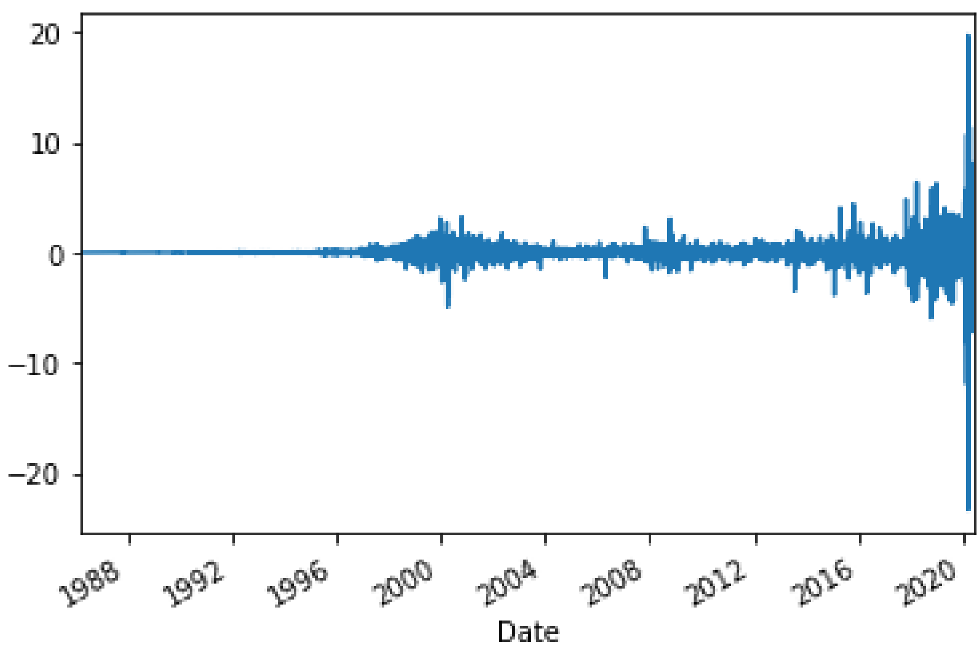


Clearly the above data series is not stationary - a critical assumption of ols, so lets calculate a percentage growth rate for stock price and perhaps this will be a more useful variable to model on

[11]: Text(0, 0.5, 'Stock Price')



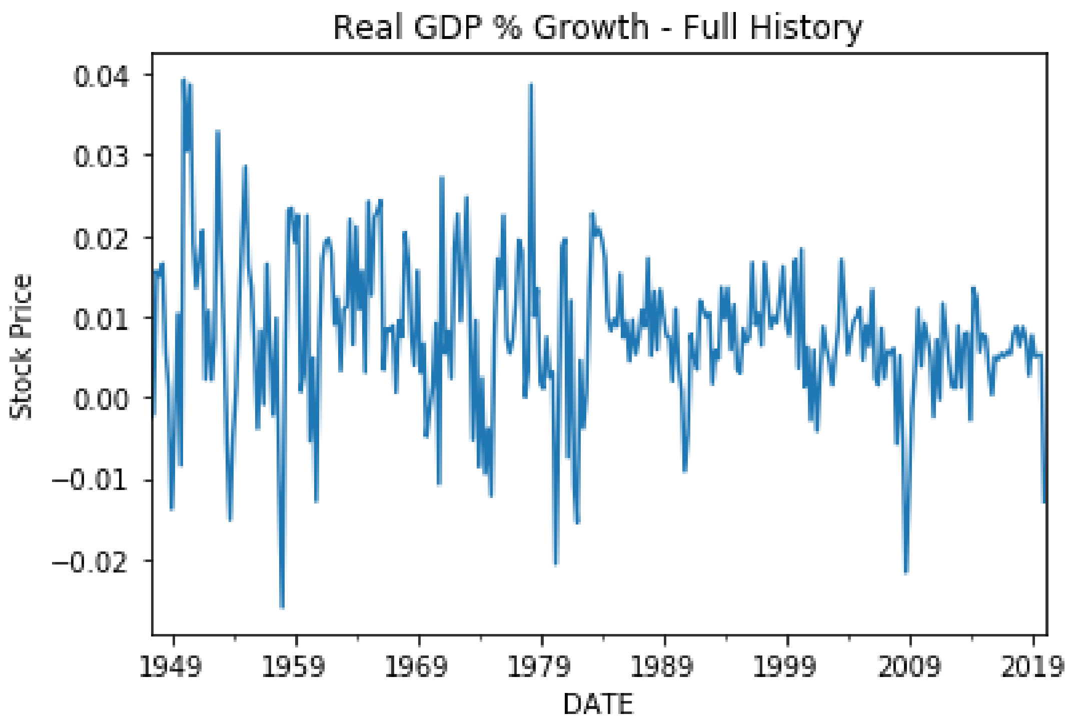
[12]: <matplotlib.axes._subplots.AxesSubplot at 0x12ac19a90>



Between the two graphs, we can see that the percentage difference is much more random-walk like. As changes in price get greater as the price itself gets greater, causing some heteroskedasticity concerns. Thus % difference will be our preferred dependent variable for the model.

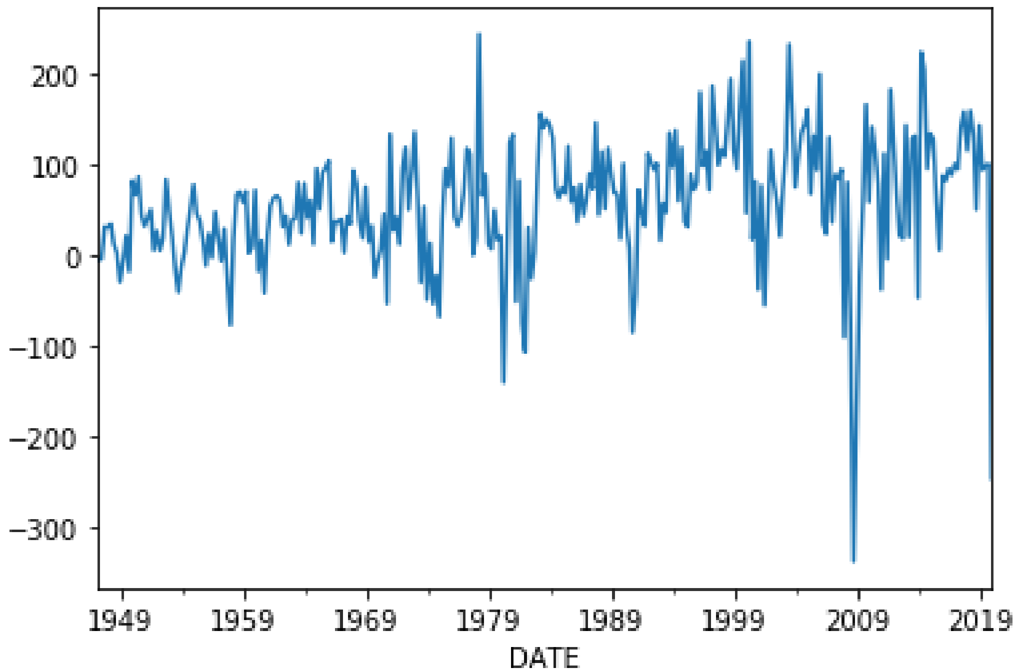
We'll start by building a model on GDP, since it is generally a great economic indicator. Let's explore this variable in a similar way to how we explored our stock price.

```
[13]: Text(0, 0.5, 'Stock Price')
```



We see that this is good, however we may have some heteroskedasticity (non-constant variance) concerns as the earlier part of the data series shows higher peaks and valleys than the later part. While heteroskedasticity concerns the residuals, we note that stock price is not an adjusted weight, whereas our macroeconomics variables are represented as inflation adjusted, calculated based off a base year (2012 in our case). Thus we should check if differencing would be a more promising approach.

```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0x12d898350>
```



This looks a bit more like a random walk so we will proceed to model based off this.

1.1.4 Part 4: Modeling with statsmodels

We will try to build a simple model with real gdp as our independent variable predicting the stock price. We will add an intercept as well. Our first step is to prepare the data to pass to OLS. We will see this is generally the bulk of the work, as producing the model is very simple using statsmodels. First let's take a look at our two dataframes.

```
Date
1986-03-14    0.000000
1986-03-17    0.166667
1986-03-18   -0.142857
1986-03-19    0.000000
1986-03-20    0.000000
                ...
2020-05-28   -0.002255
2020-05-29    0.010198
2020-06-01   -0.002292
2020-06-02    0.011377
2020-06-03    0.002434
Name: Close, Length: 8626, dtype: float64
```

```
DATE
```

```
1947-04-01     -5.422
1947-07-01     -4.187
1947-10-01     31.651
1948-01-01     30.914
1948-04-01     34.433
                ...
2019-01-01    143.733
2019-04-01     94.579
2019-07-01     99.252
2019-10-01    100.858
2020-01-01   -247.268
Name: GDPC1, Length: 292, dtype: float64
```

There's some problems for us to solve here. We note that the stock price dataframe has closing prices for all business days since 1986, while the GDP data frame has quarterly measures from 1947. Lets begin by finding the closing price for each quarter for the stock so that it will match up similar to the representation of GDP. We also note that GDP is indexed by the beginning of each quarter's date, so we will have to align this as well so that we can clearly represent our dataframes and confirm we have successfully aligned the data.

[16]:
```
[Timestamp('1986-03-31 00:00:00'),
 Timestamp('1986-06-30 00:00:00'),
 Timestamp('1986-09-30 00:00:00'),
 Timestamp('1986-12-31 00:00:00'),
 Timestamp('1987-03-31 00:00:00'),
 Timestamp('1987-06-30 00:00:00'),
 Timestamp('1987-09-30 00:00:00'),
 Timestamp('1987-12-31 00:00:00'),
 Timestamp('1988-03-31 00:00:00'),
 Timestamp('1988-06-30 00:00:00'),
 Timestamp('1988-09-30 00:00:00'),
 Timestamp('1988-12-30 00:00:00'),
 Timestamp('1989-03-31 00:00:00'),
 Timestamp('1989-06-30 00:00:00'),
 Timestamp('1989-09-29 00:00:00'),
 Timestamp('1989-12-29 00:00:00'),
 Timestamp('1990-03-30 00:00:00'),
 Timestamp('1990-06-29 00:00:00'),
 Timestamp('1990-09-28 00:00:00'),
 Timestamp('1990-12-31 00:00:00'),
 Timestamp('1991-03-28 00:00:00'),
 Timestamp('1991-06-28 00:00:00'),
 Timestamp('1991-09-30 00:00:00'),
 Timestamp('1991-12-31 00:00:00'),
 Timestamp('1992-03-31 00:00:00'),
 Timestamp('1992-06-30 00:00:00'),
 Timestamp('1992-09-30 00:00:00'),
```

```
Timestamp('1992-12-31 00:00:00'),
Timestamp('1993-03-31 00:00:00'),
Timestamp('1993-06-30 00:00:00'),
Timestamp('1993-09-30 00:00:00'),
Timestamp('1993-12-31 00:00:00'),
Timestamp('1994-03-31 00:00:00'),
Timestamp('1994-06-30 00:00:00'),
Timestamp('1994-09-30 00:00:00'),
Timestamp('1994-12-30 00:00:00'),
Timestamp('1995-03-31 00:00:00'),
Timestamp('1995-06-30 00:00:00'),
Timestamp('1995-09-29 00:00:00'),
Timestamp('1995-12-29 00:00:00'),
Timestamp('1996-03-29 00:00:00'),
Timestamp('1996-06-28 00:00:00'),
Timestamp('1996-09-30 00:00:00'),
Timestamp('1996-12-31 00:00:00'),
Timestamp('1997-03-31 00:00:00'),
Timestamp('1997-06-30 00:00:00'),
Timestamp('1997-09-30 00:00:00'),
Timestamp('1997-12-31 00:00:00'),
Timestamp('1998-03-31 00:00:00'),
Timestamp('1998-06-30 00:00:00'),
Timestamp('1998-09-30 00:00:00'),
Timestamp('1998-12-31 00:00:00'),
Timestamp('1999-03-31 00:00:00'),
Timestamp('1999-06-30 00:00:00'),
Timestamp('1999-09-30 00:00:00'),
Timestamp('1999-12-31 00:00:00'),
Timestamp('2000-03-31 00:00:00'),
Timestamp('2000-06-30 00:00:00'),
Timestamp('2000-09-29 00:00:00'),
Timestamp('2000-12-29 00:00:00'),
Timestamp('2001-03-30 00:00:00'),
Timestamp('2001-06-29 00:00:00'),
Timestamp('2001-09-28 00:00:00'),
Timestamp('2001-12-31 00:00:00'),
Timestamp('2002-03-28 00:00:00'),
Timestamp('2002-06-28 00:00:00'),
Timestamp('2002-09-30 00:00:00'),
Timestamp('2002-12-31 00:00:00'),
Timestamp('2003-03-31 00:00:00'),
Timestamp('2003-06-30 00:00:00'),
Timestamp('2003-09-30 00:00:00'),
Timestamp('2003-12-31 00:00:00'),
Timestamp('2004-03-31 00:00:00'),
Timestamp('2004-06-30 00:00:00'),
```

```
Timestamp('2004-09-30 00:00:00'),
Timestamp('2004-12-31 00:00:00'),
Timestamp('2005-03-31 00:00:00'),
Timestamp('2005-06-30 00:00:00'),
Timestamp('2005-09-30 00:00:00'),
Timestamp('2005-12-30 00:00:00'),
Timestamp('2006-03-31 00:00:00'),
Timestamp('2006-06-30 00:00:00'),
Timestamp('2006-09-29 00:00:00'),
Timestamp('2006-12-29 00:00:00'),
Timestamp('2007-03-30 00:00:00'),
Timestamp('2007-06-29 00:00:00'),
Timestamp('2007-09-28 00:00:00'),
Timestamp('2007-12-31 00:00:00'),
Timestamp('2008-03-31 00:00:00'),
Timestamp('2008-06-30 00:00:00'),
Timestamp('2008-09-30 00:00:00'),
Timestamp('2008-12-31 00:00:00'),
Timestamp('2009-03-31 00:00:00'),
Timestamp('2009-06-30 00:00:00'),
Timestamp('2009-09-30 00:00:00'),
Timestamp('2009-12-31 00:00:00'),
Timestamp('2010-03-31 00:00:00'),
Timestamp('2010-06-30 00:00:00'),
Timestamp('2010-09-30 00:00:00'),
Timestamp('2010-12-31 00:00:00'),
Timestamp('2011-03-31 00:00:00'),
Timestamp('2011-06-30 00:00:00'),
Timestamp('2011-09-30 00:00:00'),
Timestamp('2011-12-30 00:00:00'),
Timestamp('2012-03-30 00:00:00'),
Timestamp('2012-06-29 00:00:00'),
Timestamp('2012-09-28 00:00:00'),
Timestamp('2012-12-31 00:00:00'),
Timestamp('2013-03-28 00:00:00'),
Timestamp('2013-06-28 00:00:00'),
Timestamp('2013-09-30 00:00:00'),
Timestamp('2013-12-31 00:00:00'),
Timestamp('2014-03-31 00:00:00'),
Timestamp('2014-06-30 00:00:00'),
Timestamp('2014-09-30 00:00:00'),
Timestamp('2014-12-31 00:00:00'),
Timestamp('2015-03-31 00:00:00'),
Timestamp('2015-06-30 00:00:00'),
Timestamp('2015-09-30 00:00:00'),
Timestamp('2015-12-31 00:00:00'),
Timestamp('2016-03-31 00:00:00'),
```

```
 Timestamp('2016-06-30 00:00:00'),
 Timestamp('2016-09-30 00:00:00'),
 Timestamp('2016-12-30 00:00:00'),
 Timestamp('2017-03-31 00:00:00'),
 Timestamp('2017-06-30 00:00:00'),
 Timestamp('2017-09-29 00:00:00'),
 Timestamp('2017-12-29 00:00:00'),
 Timestamp('2018-03-29 00:00:00'),
 Timestamp('2018-06-29 00:00:00'),
 Timestamp('2018-09-28 00:00:00'),
 Timestamp('2018-12-31 00:00:00'),
 Timestamp('2019-03-29 00:00:00'),
 Timestamp('2019-06-28 00:00:00'),
 Timestamp('2019-09-30 00:00:00'),
 Timestamp('2019-12-31 00:00:00'),
 Timestamp('2020-03-31 00:00:00')]
```

This was successful. We can see that we have isolated the business day for each quarter from the stock price data frame. Now we need to use this list of timestamps to filter our dataframe to these stock prices. Then we will want to change the index, so that it shows the first day of each quarter, instead of the last business day.

[17]:
```
1986-01-01      0.06
1986-04-01      0.07
1986-07-01      0.06
1986-10-01      0.11
1987-01-01      0.21
                 …
2019-01-01    116.08
2019-04-01    132.33
2019-07-01    137.80
2019-10-01    156.83
2020-01-01    157.27
Name: Close, Length: 137, dtype: float64
```

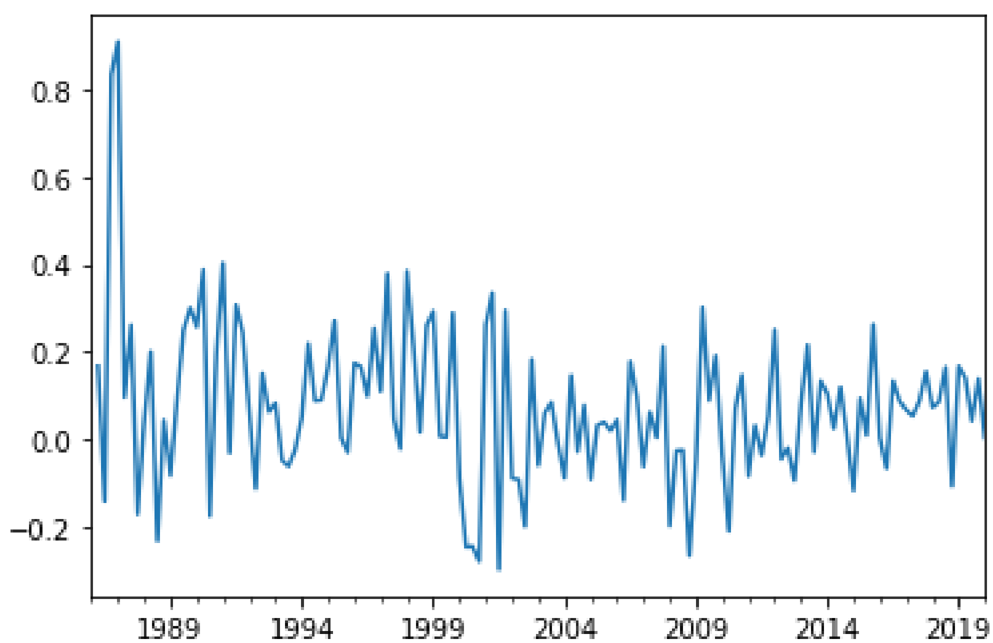
Now we are ready to merge in the dataframes and only keep the data that is relevant for both variables (ie remove 1947-1986 from our real gdp variable). First let's make a function out of what we have just done, so that we can reuse in the future.

[18]:
```
1986-01-01      0.06
1986-04-01      0.07
1986-07-01      0.06
1986-10-01      0.11
1987-01-01      0.21
                 …
2019-01-01    116.08
2019-04-01    132.33
2019-07-01    137.80
```

```
2019-10-01    156.83
2020-01-01    157.27
Name: Close, Length: 137, dtype: float64
```

This was successful, so let's put the two variables together to make sure we don't run into any dimension issues when we use OLS. Recall we wanted to use the price growth (in %) with the gdp difference

```
            Stock Price Quarterly % Growth  GDP difference
1947-01-01                             NaN             NaN
1947-04-01                             NaN          -5.422
1947-07-01                             NaN          -4.187
1947-10-01                             NaN          31.651
1948-01-01                             NaN          30.914
...                                    ...             ...
2019-01-01                        0.166164         143.733
2019-04-01                        0.139990          94.579
2019-07-01                        0.041336          99.252
2019-10-01                        0.138099         100.858
2020-01-01                        0.002806        -247.268

[293 rows x 2 columns]
```



We know our data has no missing values, that's why this next step is okay. Be cautious to deal with missing values prior to doing this step as it will remove the row for any missing value, which is dangerous if that value is part of our intended development period.

[20]:
```
                Stock Price Quarterly % Growth  GDP difference
1986-04-01                            0.166667          36.700
1986-07-01                           -0.142857          78.336
1986-10-01                            0.833333          44.382
1987-01-01                            0.909091          61.909
1987-04-01                            0.095238          90.303
…                                          …             …
2019-01-01                            0.166164         143.733
2019-04-01                            0.139990          94.579
2019-07-01                            0.041336          99.252
2019-10-01                            0.138099         100.858
2020-01-01                            0.002806        -247.268

[136 rows x 2 columns]
```

We're pretty close now. To use statsmodels ols function we need to pass it y - our dependent variable and X - our design matrix. Thus we have to split this up again, then we will add our constant to the design matrix

```
1986-04-01    0.166667
1986-07-01   -0.142857
1986-10-01    0.833333
1987-01-01    0.909091
1987-04-01    0.095238
                ...
2019-01-01    0.166164
2019-04-01    0.139990
2019-07-01    0.041336
2019-10-01    0.138099
2020-01-01    0.002806
Name: Stock Price Quarterly % Growth, Length: 136, dtype: float64


/usr/local/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2495:
FutureWarning: Method .ptp is deprecated and will be removed in a future
version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)

            const  GDP difference
1986-04-01    1.0          36.700
1986-07-01    1.0          78.336
1986-10-01    1.0          44.382
1987-01-01    1.0          61.909
1987-04-01    1.0          90.303
...           ...             ...
2019-01-01    1.0         143.733
2019-04-01    1.0          94.579
2019-07-01    1.0          99.252
2019-10-01    1.0         100.858
```

```
2020-01-01    1.0        -247.268

[136 rows x 2 columns]
```

Now we are ready to pass this to OLS and look at a summary of the model

[22]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                OLS Regression Results
================================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.011
Model:                                        OLS   Adj. R-squared:
0.003
Method:                             Least Squares   F-statistic:
1.434
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
0.233
Time:                                    19:09:07   Log-Likelihood:
41.576
No. Observations:                             136   AIC:
-79.15
Df Residuals:                                 134   BIC:
-73.33
Df Model:                                       1
Covariance Type:                        nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--
const              0.0554      0.022      2.561      0.012       0.013
0.098
GDP difference     0.0002      0.000      1.197      0.233      -0.000
0.001
==============================================================================
Omnibus:                       46.544   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              150.486
Skew:                           1.250   Prob(JB):                     2.10e-33
Kurtosis:                       7.506   Cond. No.                         159.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
"""
```

We see that we have an insignificant variable, as evidenced by the low p-value on the GDP Difference parameter. Also, our R-squared is quite weak, meaning we might be wise to consider a different explanatory variable. For fast prototyping, let's define a function that will take in our reg data frame - with y as the first column and the rest of the columns as regressor variables, and create the model.

[23]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                  OLS Regression Results
================================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.011
Model:                                        OLS   Adj. R-squared:
0.003
Method:                             Least Squares   F-statistic:
1.434
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
0.233
Time:                                    19:09:07   Log-Likelihood:
41.576
No. Observations:                             136   AIC:
-79.15
Df Residuals:                                 134   BIC:
-73.33
Df Model:                                       1
Covariance Type:                        nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--
const              0.0554      0.022      2.561      0.012       0.013
0.098
GDP difference     0.0002      0.000      1.197      0.233      -0.000
0.001
==============================================================================
Omnibus:                       46.544   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              150.486
Skew:                           1.250   Prob(JB):                     2.10e-33
Kurtosis:                       7.506   Cond. No.                         159.
==============================================================================

Warnings:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

We see that this worked so let's explore some alternative variables. First we will make our data handling a bit easier by making each macroeconomic variable it's own data series, whereas it had previously been a dataframe. This way we don't have to keep remembering the weird column headers in our dataframes.

GDP Difference Lag 2

[25]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.015
Model:                                        OLS   Adj. R-squared:
0.007
Method:                             Least Squares   F-statistic:
1.985
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
0.161
Time:                                    19:09:07   Log-Likelihood:
41.853
No. Observations:                             136   AIC:
-79.71
Df Residuals:                                 134   BIC:
-73.88
Df Model:                                       1
Covariance Type:                        nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0971      0.023      4.288      0.000       0.052       0.142
GDPC1         -0.0003      0.000     -1.409      0.161      -0.001       0.000
==============================================================================
Omnibus:                       41.866   Durbin-Watson:                   1.774
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              130.626
Skew:                           1.124   Prob(JB):                     4.31e-29
Kurtosis:                       7.243   Cond. No.                         164.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

"""

PCE Growth

```
[26]: <class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.001
Model:                                        OLS   Adj. R-squared:
-0.013
Method:                             Least Squares   F-statistic:
0.08744
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
0.768
Time:                                    19:09:07   Log-Likelihood:
53.743
No. Observations:                              72   AIC:
-103.5
Df Residuals:                                  70   BIC:
-98.93
Df Model:                                       1
Covariance Type:                        nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0325      0.018      1.758      0.083      -0.004       0.069
PCEC96         1.6399      5.546      0.296      0.768      -9.421      12.700
==============================================================================
Omnibus:                        0.489   Durbin-Watson:                   2.220
Prob(Omnibus):                  0.783   Jarque-Bera (JB):                0.373
Skew:                          -0.174   Prob(JB):                        0.830
Kurtosis:                       2.944   Cond. No.                         404.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Disposable Income per capita growth

```
[27]: <class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
```

```
==============================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.003
Model:                                        OLS   Adj. R-squared:
-0.004
Method:                             Least Squares   F-statistic:
0.4368
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
0.510
Time:                                    19:09:07   Log-Likelihood:
41.074
No. Observations:                             136   AIC:
-78.15
Df Residuals:                                 134   BIC:
-72.32
Df Model:                                       1
Covariance Type:                        nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0737      0.015      4.769      0.000       0.043       0.104
A229RX0       -1.1735      1.776     -0.661      0.510      -4.685       2.338
==============================================================================
Omnibus:                       43.509   Durbin-Watson:                   1.766
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              135.674
Skew:                           1.174   Prob(JB):                     3.46e-30
Kurtosis:                       7.293   Cond. No.                         115.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Still nothing great. Lets get the dow jones industrial average from Yahoo! finance and see if maybe that is more promising

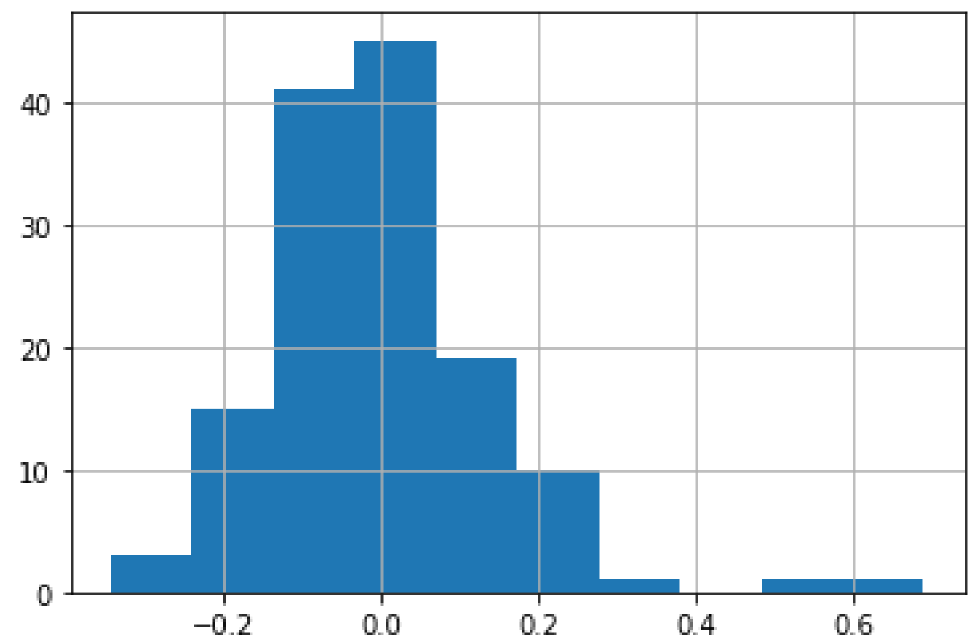
Dow Jones Industrial Average

[29]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                  OLS Regression Results
==============================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.352
```

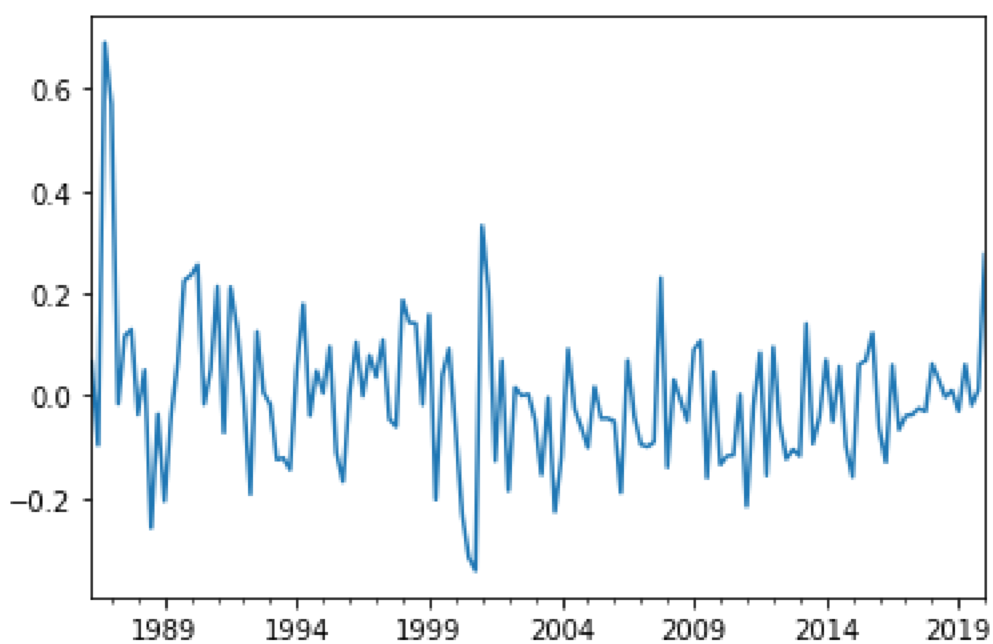
```
Model:                            OLS   Adj. R-squared:
0.347
Method:                 Least Squares   F-statistic:
72.82
Date:                Wed, 03 Jun 2020   Prob (F-statistic):
2.69e-14
Time:                        19:09:08   Log-Likelihood:
70.367
No. Observations:                 136   AIC:
-136.7
Df Residuals:                     134   BIC:
-130.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0441      0.013      3.410      0.001       0.019       0.070
Close          1.3674      0.160      8.534      0.000       1.050       1.684
==============================================================================
Omnibus:                       43.437   Durbin-Watson:                   1.596
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.380
Skew:                           1.178   Prob(JB):                     1.09e-29
Kurtosis:                       7.241   Cond. No.                         12.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Hey! 35% R-squared might not sound that good but in practice, this isn't too bad. We have a significant parameter on the dow, this looks like its got some promise so let's explore further.

Let's look at the residuals and see if they look normally distributed and also plot them to see if heteroskedasticity seems to be an issue. We will have formal tests for this in the diagnostics section, but a quick visual inspection is often useful.

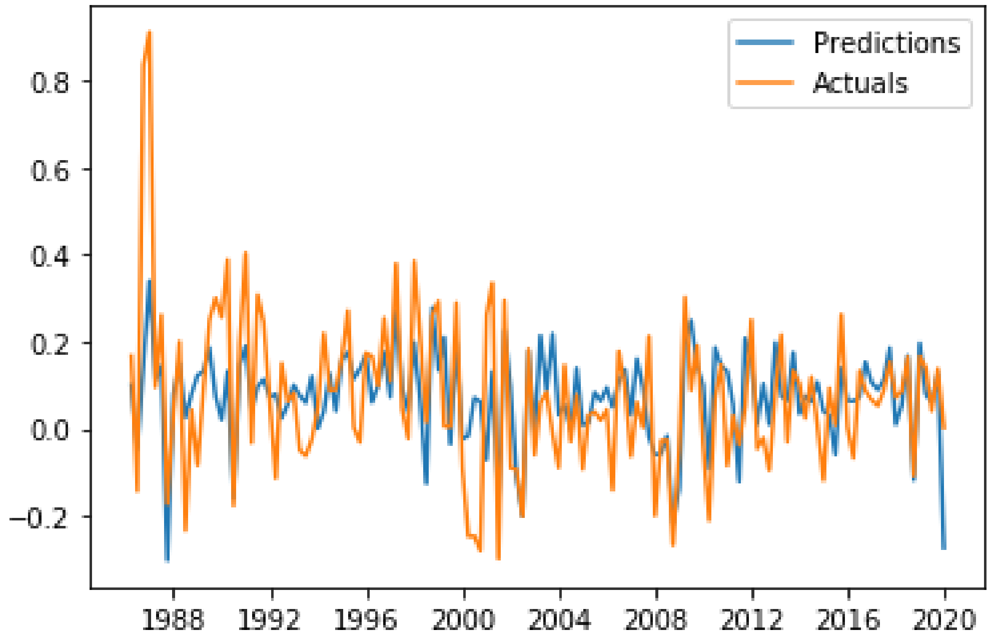


[30]: <matplotlib.axes._subplots.AxesSubplot at 0x12e0be7d0>



Let's backtest this a bit. Statsmodels makes it easy to do a one-quarter ahead backtest. While we care more about dynamic backtesting, this can be informative.

[31]: <matplotlib.legend.Legend at 0x12eb535d0>



1.1.5 Part 5: Diagnostics

As we approach our diagnostics, we recall that there are some fundamental assumptions of linear regression that must hold true with our model. These assumptions will be the focus of our diagnostics tests. * Residuals are normally distributed * All regression variables (independent and dependent) are stationary * Heteroskedasticity is not present in the residuals * Autocorrelation is not present in the residuals

while there are several methods to test each of these assumptions, we will just choose one for each for today.

Residual Normality - Shapiro-Wilk Test H_0: Residuals are normally distributed

H_a: Residuals are not normally distributed

We see below from our low p-value that our residuals show evidence of not being normally distributed, which is an issue.

[32]: (0.9330253601074219, 4.456813712749863e-06)

Stationarity of Variables - Augmented Dickey-Fuller Test H_0: Unit root is present (variable is not stationary)

H_a: Unit root is not present (variable is stationary)

We see below from our low p-values on both our variables that they are stationary.

[33]:
```
(-5.779606998136448,
 5.162594305814863e-07,
 2,
 133,
 {'1%': -3.480500383888377,
  '5%': -2.8835279559405045,
  '10%': -2.578495716547007},
 -112.68408719448911)
```

[34]:
```
(-11.955311198093387,
 4.223335755473661e-22,
 0,
 135,
 {'1%': -3.479742586699182,
  '5%': -2.88319822181578,
  '10%': -2.578319684499314},
 -283.73635073592243)
```

Heteroskedasticity of Residuals - White's Test H_0: Homoskedasticity

H_a: Heteroskedasticity

We see the p-value is 0.01, indicating that heteroskedasticity is an issue with this model.

[35]:
```
(8.857749994344147,
 0.011927901049868917,
 4.6329239461916325,
 0.011348552272043518)
```

We can use heteroskedasticity-consistent standard errors to check if our model specification is still significant in the presence of heteroskedasticity, and indeed it is.

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                  OLS Regression Results
================================================================================
==========
Dep. Variable:     Stock Price Quarterly % Growth   R-squared:
0.352
Model:                                        OLS   Adj. R-squared:
0.347
Method:                             Least Squares   F-statistic:
41.24
Date:                            Wed, 03 Jun 2020   Prob (F-statistic):
2.16e-09
Time:                                    19:09:09   Log-Likelihood:
```

```
70.367
No. Observations:                     136   AIC:
-136.7
Df Residuals:                         134   BIC:
-130.9
Df Model:                               1
Covariance Type:                      HAC
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0441      0.013      3.515      0.001       0.019       0.069
Close          1.3674      0.213      6.422      0.000       0.946       1.789
==============================================================================
Omnibus:                       43.437   Durbin-Watson:                   1.596
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.380
Skew:                           1.178   Prob(JB):                     1.09e-29
Kurtosis:                       7.241   Cond. No.                         12.9
==============================================================================

Warnings:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC)
using 1 lags and without small sample correction
"""
```

1.1.6 Autocorrelation of Residuals - Breusch-Godfrey Test

H_0: No autocorrelation is present

H_a: Autocorrelation is present

We see from our p-value of 0.2 that autocorrelation does not seem to be present in this model.

[37]:
```
(5.958441218424532, 0.2022734799828453, 1.4891342537969865, 0.2091811367216205)
```