Machine Learning Engineer Nanodegree

Capstone Project Report -

Akshat Bhardwaj

5th Dec 2018

I. Definition

S&P 50 Index Price prediction through Neural networks and statistical modelling.

Project Overview

To predict S&P 500 Index prices using historical data since 1871 and statistical models like Linear Regression, LSTM(Long Short-term Memory) and fbProphet.

Domain Background

Domain background for this Capstone project is investment banking. The Standard & Poor's 500, often abbreviated as the S&P 500, or just the S&P,is an American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ. The S&P 500 index components and their weightings are determined by S&P Dow Jones Indices. It differs from other U.S. stock market indices, such as the Dow Jones Industrial Average or the NASDAQ Composite index, because of its diverse constituency and weighting methodology.

This project and topic is relevant for predicting stock and index prices using AI and machine learning algorithms because this is a field where lots of data and experience are funneled and fed to different algorithm based prediction and trading. This project is another try to verify future pricing based on historical data and prediction analysis.

Personal motive for this project is to predict S&P 500 index price and to compare how various machine learning models perform in predicting price for time series data based on historical variables like sales, price, PE ratio etc with supervised learning models. Motivation is also to implement supervised learning and Neural Networks on real time dataset to check validity against market value.

I could not find any study directly applicable to the features and dataset I have selected. But below given studies are similar to what I am trying to achieve.

Reference- http://www.diva-portal.org/smash/get/diva2:1213449/FULLTEXT01.pdf

https://www.kaggle.com/amarpreetsingh/stock-prediction-lstm-using-keras/notebook

S&P 500 Index- https://en.wikipedia.org/wiki/S%26P_500_Index

Problem Statement

Problem statement is to predict S&P 500 index price for next month and find short term trend in price movement. This is a **regression problem** based on historical data provided in time-series.

Some surveys show 30% of all stock trading done on NYSE is driven by machine learning and algorithmic models. Problem statement for this project is to verify how accurate price predictions can machine learning make based on historical monthly, quarterly and yearly data available about S&P500 index vs what has been the closing price of index. See the dataset features available in next section.

If machine learning stock trading are to be believed, then which algorithm I have selected below suits best in this given context and dataset.

Implementation strategy-

1. Data capture and data loading into master Pandas dataframe.
2. Handling missing values- Imputation and Interpolation.
3. Data Visualization and data analysis.
4. Feature Engineering.
5. Data preprocessing- Scaling.
6. Modelling and deep learning.
7. Model Evaluation Metrics

Metrics

1. Mean Square Error for Linear Regression, LSTM and fbProphet.
2. R2 Score for Linear Regression, LSTM and fbProphet.
3. LSTM Validation loss
4. LSTM Training loss

II. Analysis

Data Exploration

Dataset-

I have selected a dataset from Quandl.com where MULTPL provides S&P500 index related attributes for free. This data is split into monthly, quarterly and yearly basis and available through API upto latest date. Dataset is a time-series since the inception of S&P500 index in 1871. I could not find any dataset or project on Kaggle or similar websites with same dataset or with same objective. There are some stock price datasets and regression projects for predicting stock price or S&P 500 index price prediction but not on similar dataset.

Dataset info- Original data as received from Quandl API has 15.2% missing or NaN values in the time series. The reason is monthly, quarterly and year values on the time series like S&P 500 Dividend Growth by Year,S&P 500 Dividend Growth by Quarter,S&P 500 Price to Book Value by Quarter,Shiller PE Ratio by Year.

Dataset info:

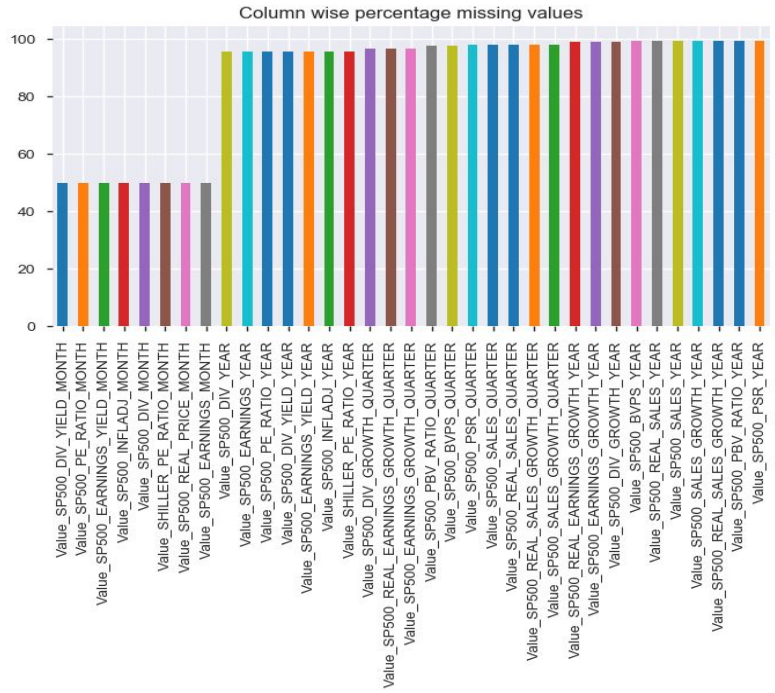
Number of variables	36
Number of observations	3558
Total Missing (%)	15.2%
Total size in memory	1000.8 KiB
Average record size in memory	288.0 B

Variables types -

Numeric	36
Categorical	0
Boolean	0
Date	1
Text (Unique)	0

NaN or missing values-

Challenge is to handle NaN values for dates where yield and ratio are not available. As depicted in the plot below, all yearly data values have more than 90% missing values are converted to np.NaN values in pandas dataframe. These NaN values would be handled later in data preprocessing step. All NaN values have you either imputed or interpolated using scikit.



Data API connection-

Quandl API gives data in pandas dataframe format per dataset. I have merged all 36 columns on ‘Date’ timeseries. I would be using my personal key from my free account on Quandl data website where MULTPL provides API the following attributes for S&P500 index in time-series. All of the data combined together affects the price of S&P 500 index together with underlying constituent equity data. Underlying constituent equity data is out-of-scope for this project but would be interesting to include for further studies.

Snippet of API to fetch data from Quandl.

```
In [198]: #Get data from Quandl APIs into dataframes
SP500_DIV_YIELD_MONTH = quandl.get('MULTPL/SP500_DIV_YIELD_MONTH') #MULTPLkeys[0]
SP500_PE_RATIO_MONTH = quandl.get('MULTPL/SP500_PE_RATIO_MONTH')
SHILLER_PE_RATIO_MONTH = quandl.get('MULTPL/SHILLER_PE_RATIO_MONTH')
SP500_EARNINGS_YIELD_MONTH = quandl.get('MULTPL/SP500_EARNINGS_YIELD_MONTH')
SP500_INFLADJ_MONTH = quandl.get('MULTPL/SP500_INFLADJ_MONTH')#MULTPLkeys[4]

SP500_PSR_QUARTER = quandl.get('MULTPL/SP500_PSR_QUARTER')
SP500_DIV_MONTH = quandl.get('MULTPL/SP500_DIV_MONTH')
SP500_DIV_YEAR = quandl.get('MULTPL/SP500_DIV_YEAR')
SP500_DIV_GROWTH_YEAR = quandl.get('MULTPL/SP500_DIV_GROWTH_YEAR')
SP500_DIV_GROWTH_QUARTER = quandl.get('MULTPL/SP500_DIV_GROWTH_QUARTER')
SP500_PBV_RATIO_QUARTER = quandl.get('MULTPL/SP500_PBV_RATIO_QUARTER') #MULTPLkeys[10]
```

Sample dataframe from API-

```
In [207]: SP500_DIV_YIELD_MONTH.head()
```

Out[207]:

Date	Value
1871-01-31	5.86
1871-02-28	5.78
1871-03-31	5.64
1871-04-30	5.49
1871-05-31	5.35

Output data - “S&P 500 Real Price by Month” (Target output) prediction.

Input data- 35 features available in the dataset which shows sales, P/E, growth, dividend ratios etc per month, quarter and Annually.

Index name- S&P 500 Index

Asset class- Equities

Characteristics of the dataset-

1. There are 36 data points.
2. Time-series data is split per month, per quarter and per year.
3. “S&P 500 Real Price by Month” is the target variable (y).
4. Earning yield, price to sales ratios and yeilds are available in time-series.
5. Data available is linear and ratios for many columns.
6. Columns like S&P 500 Sales by Year, S&P 500 Real Sales by Year have values much higher than other scaled down ratios which should be scaled down all together to avoid any biases.

S&P 500 Dividend Yield by Month	MULTPL/SP500_DIV_YIELD_MONTH

S&P 500 PE Ratio by Month	MULTPL/SP500_PE_RATIO_MONTH
Shiller PE Ratio by Month	MULTPL/SHILLER_PE_RATIO_MONTH
S&P 500 Earnings Yield by Month	MULTPL/SP500_EARNINGS_YIELD_MONTH
S&P 500 Inflation Adjusted by Month	MULTPL/SP500_INFLADJ_MONTH
S&P 500 Price to Sales Ratio by Quarter	MULTPL/SP500_PSR_QUARTER
S&P 500 Dividend by Month	MULTPL/SP500_DIV_MONTH
S&P 500 Dividend by Year	MULTPL/SP500_DIV_YEAR
S&P 500 Dividend Growth by Year	MULTPL/SP500_DIV_GROWTH_YEAR
S&P 500 Dividend Growth by Quarter	MULTPL/SP500_DIV_GROWTH_QUARTER
S&P 500 Price to Book Value by Quarter	MULTPL/SP500_PBV_RATIO_QUARTER
Shiller PE Ratio by Year	MULTPL/SHILLER_PE_RATIO_YEAR
S&P 500 PE Ratio by Year	MULTPL/SP500_PE_RATIO_YEAR
S&P 500 Dividend Yield by Year	MULTPL/SP500_DIV_YIELD_YEAR
S&P 500 Price to Sales Ratio by Year	MULTPL/SP500_PSR_YEAR
S&P 500 Earnings Yield by Year	MULTPL/SP500_EARNINGS_YIELD_YEAR
S&P 500 Price to Book Value by Year	MULTPL/SP500_PBV_RATIO_YEAR
S&P 500 Inflation Adjusted by Year	MULTPL/SP500_INFLADJ_YEAR
S&P 500 Real Price by Month (Target output)	MULTPL/SP500_REAL_PRICE_MONT
S&P 500 Sales by Year	MULTPL/SP500_SALES_YEAR
S&P 500 Sales Growth Rate by Year	MULTPL/SP500_SALES_GROWTH_YEAR
S&P 500 Sales by Quarter	MULTPL/SP500_SALES_QUARTER
S&P 500 Real Sales Growth by Quarter	MULTPL/SP500_REAL_SALES_GROWTH_QUARTER
S&P 500 Sales Growth Rate by Quarter	MULTPL/SP500_SALES_GROWTH_QUARTER
S&P 500 Real Sales Growth by Year	MULTPL/SP500_REAL_SALES_GROWTH_YEAR
S&P 500 Real Earnings Growth by Year	MULTPL/SP500_REAL_EARNINGS_GROWTH_YEAR
S&P 500 Real Sales by Year	MULTPL/SP500_REAL_SALES_YEAR
S&P 500 Real Earnings Growth by Quarter	MULTPL/SP500_REAL_EARNINGS_GROWTH_QUARTER
S&P 500 Earnings Growth Rate by Quarter	MULTPL/SP500_EARNINGS_GROWTH_QUARTER
S&P 500 Real Sales by Quarter	MULTPL/SP500_REAL_SALES_QUARTER
S&P 500 Earnings by Month	MULTPL/SP500_EARNINGS_MONTH

S&P 500 Book Value Per Share by Year	MULTPL/SP500_BVPS_YEAR
S&P 500 Earnings by Year	MULTPL/SP500_EARNINGS_YEAR
S&P 500 Earnings Growth Rate by Year	MULTPL/SP500_EARNINGS_GROWTH_YEAR
S&P 500 Book Value Per Share by Quarter	MULTPL/SP500_BVPS_QUARTER
S&P 500 Real Price by Year	MULTPL/SP500_REAL_PRICE_YEAR
Date	Timeseries since 1871

All column names of master df after joining/merging 36 dataframes from API-

```
Index(['Value_SP500_REAL_PRICE_MONTH', 'Value_SP500_DIV_YIELD_MONTH',
       'Value_SP500_PE_RATIO_MONTH', 'Value_SHILLER_PE_RATIO_MONTH',
       'Value_SP500_EARNINGS_YIELD_MONTH', 'Value_SP500_INFLADJ_MONTH',
       'Value_SP500_PSR_QUARTER', 'Value_SP500_DIV_MONTH',
       'Value_SP500_DIV_YEAR', 'Value_SP500_DIV_GROWTH_YEAR',
       'Value_SP500_DIV_GROWTH_QUARTER', 'Value_SP500_PBV_RATIO_QUARTER',
       'Value_SHILLER_PE_RATIO_YEAR', 'Value_SP500_PE_RATIO_YEAR',
       'Value_SP500_DIV_YIELD_YEAR', 'Value_SP500_PSR_YEAR',
       'Value_SP500_EARNINGS_YIELD_YEAR', 'Value_SP500_PBV_RATIO_YEAR',
       'Value_SP500_INFLADJ_YEAR', 'Value_SP500_SALES_YEAR',
       'Value_SP500_SALES_GROWTH_YEAR', 'Value_SP500_SALES_QUARTER',
       'Value_SP500_REAL_SALES_GROWTH_QUARTER',
       'Value_SP500_SALES_GROWTH_QUARTER',
       'Value_SP500_REAL_SALES_GROWTH_YEAR',
       'Value_SP500_REAL_EARNINGS_GROWTH_YEAR', 'Value_SP500_REAL_SALES_YEAR',
       'Value_SP500_REAL_EARNINGS_GROWTH_QUARTER',
       'Value_SP500_EARNINGS_GROWTH_QUARTER', 'Value_SP500_REAL_SALES_QUARTER',
       'Value_SP500_EARNINGS_MONTH', 'Value_SP500_BVPS_YEAR',
       'Value_SP500_EARNINGS_YEAR', 'Value_SP500_EARNINGS_GROWTH_YEAR',
       'Value_SP500_BVPS_QUARTER'],
      dtype='object')
```

Joining and merging the data points together-

Join and/or merge all datasets on ‘Date’ column using left or outer join to create a master dataframe.

Ex-
```
df = df.merge(

      SHILLER_PE_RATIO_MONTH,on='Date',how='left').merge(
      SP500_EARNINGS_YIELD_MONTH,on='Date',how='left').merge(
      SP500_INFLADJ_MONTH,on='Date',how='left').merge(
      SP500_PSR_QUARTER,on='Date',how='left').merge(
      SP500_DIV_MONTH,on='Date',how='outer').merge(
      SP500_DIV_YEAR,on='Date',how='left')
```

Master df preview snippet-

```
In [260]: df.head()
```

Out[260]:

Date	Value_SP500_REAL_PRICE_MONTH	Value_SP500_DIV_YIELD_MONTH	Value_SP500_PE_RATIO_MONTH	Value_SHILLER_PE_RATIO_MONTH	Value_SP500_EA
1871-01-01	4.44	NaN	11.10	NaN	
1871-01-31	NaN	5.86	NaN	NaN	
1871-02-01	4.50	NaN	11.25	10.92	
1871-02-28	NaN	5.78	NaN	NaN	
1871-03-01	4.61	NaN	11.52	11.19	

5 rows × 35 columns

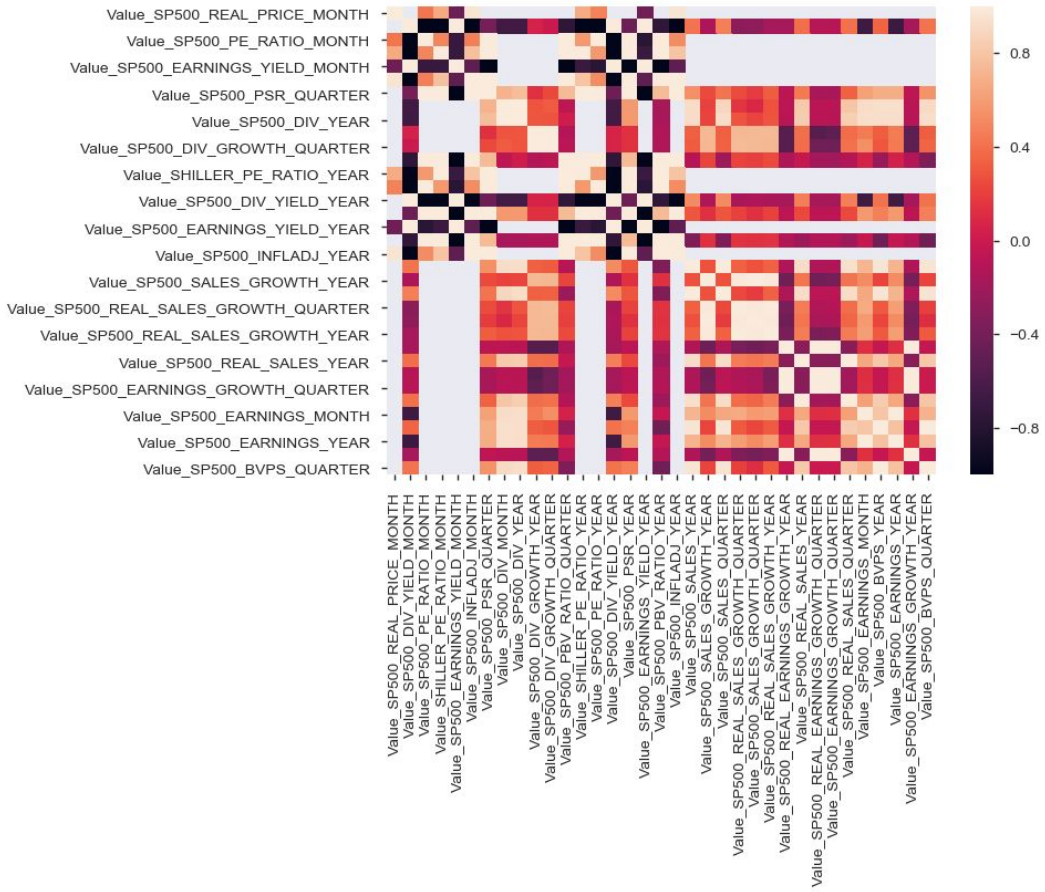
Times series available in ‘Date’ index is available for 1st of every month and last date of every month after joining all data variables. This introduced many missing values which needs to be handled.

Exploratory Visualization

After creating a master dataframe from API data variables, its time to impute and interpolate values for variables and then find out correlation between variables before and after imputation/interpolation.

Master df correlation heatmap-

```
#Correlation plot for all variable data values of API data in df.
sns.heatmap(df.corr(),annot=None,fmt='.2f',square=False)
```




```
#Pandas profiling of df
pandas_profiling.ProfileReport(df)
```

There are lots of missing data in the master dataframe hence correlation between variables is 0 at many places.

Apply imputation on master dataframe-

Dataset info

Number of variables	36
Number of observations	3547
Total Missing (%)	0.0%
Total size in memory	997.7 KiB

Average record size in memory	288.0 B

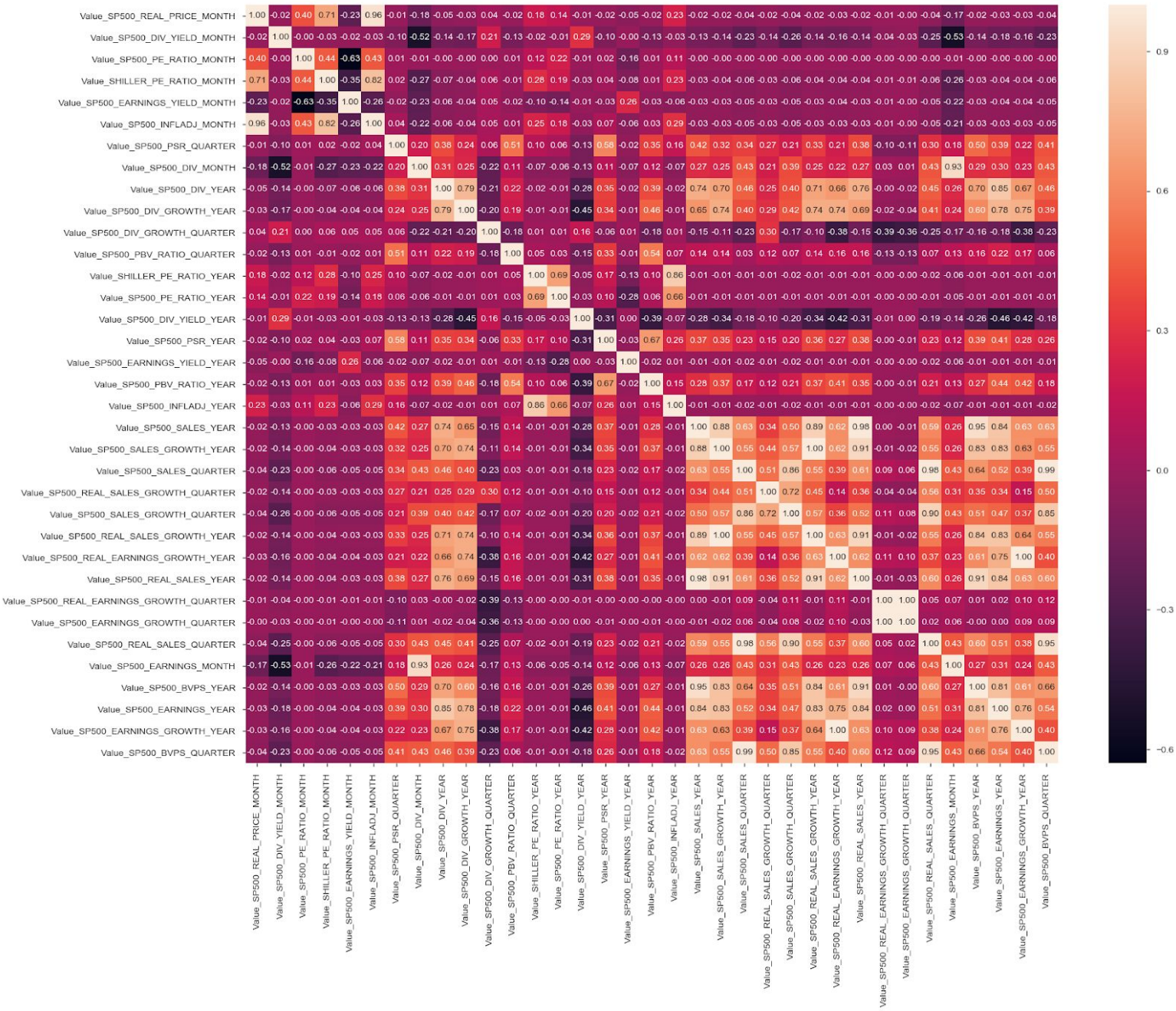
I have used SimpleImputer method from Scikit-learn library for replacing NaN values with ‘most frequent’ values used and this way I have LOCF’ed (last observation carried forward) the missing values in timeseries.

```
from sklearn.impute import SimpleImputer

#Create an imputation object

imputer_most_frequent= SimpleImputer(missing_values=np.nan,strategy
='most_frequent')

#Inject imputed values in the dataset.

df_imputed = pd.DataFrame(imputer_most_frequent.fit_transform(df))

df_imputed.columns = df.columns

df_imputed.index = df.index


pandas_profiling.ProfileReport(df_imputed)
```

Imputed dataframe profiling-

- Value_SP500_BVPS_QUARTER is highly correlated with Value_SP500_REAL_SALES_QUARTER (ρ = 0.95101)
- Value_SP500_BVPS_YEAR is highly correlated with Value_SP500_REAL_SALES_YEAR (ρ = 0.91377)
- Value_SP500_EARNINGS_GROWTH_QUARTER is highly correlated with Value_SP500_REAL_EARNINGS_GROWTH_QUARTER (ρ = 0.99834)
- Value_SP500_EARNINGS_GROWTH_YEAR is highly correlated with Value_SP500_REAL_EARNINGS_GROWTH_YEAR (ρ = 0.9994)
- Value_SP500_EARNINGS_MONTH is highly correlated with Value_SP500_DIV_MONTH (ρ = 0.93017)
- Value_SP500_INFLADJ_MONTH is highly correlated with Value_SP500_REAL_PRICE_MONTH (ρ = 0.95881)
- Value_SP500_PBV_RATIO_YEAR is highly skewed (γ1 = 23.541) Skewed
- Value_SP500_PSR_YEAR is highly skewed (γ1 = 20.073) Skewed
- Value_SP500_REAL_EARNINGS_GROWTH_QUARTER is highly skewed (γ1 = 32.837) Skewed
- Value_SP500_REAL_SALES_GROWTH_YEAR is highly correlated with Value_SP500_SALES_GROWTH_YEAR (ρ = 0.99905)
- Value_SP500_REAL_SALES_QUARTER is highly correlated with Value_SP500_SALES_QUARTER (ρ = 0.98094)
- Value_SP500_REAL_SALES_YEAR is highly correlated with Value_SP500_REAL_SALES_GROWTH_YEAR (ρ = 0.91449)

```
#Correlation plot after imputation
plt.figure(figsize=(20,15))
sns.heatmap(df_imputed.corr(),annot=True,fmt='.2f',square=False)
```



Replacing missing values using Linear Interpolation-

```
#Apply linear interpolation on the dataset.
df_interpolate =
df.interpolate(method='linear',axis=0,inplace=False,limit_direction='both')
```

Correlation of variables after applying interpolation is much better than imputation.
Interpolated data profiling-

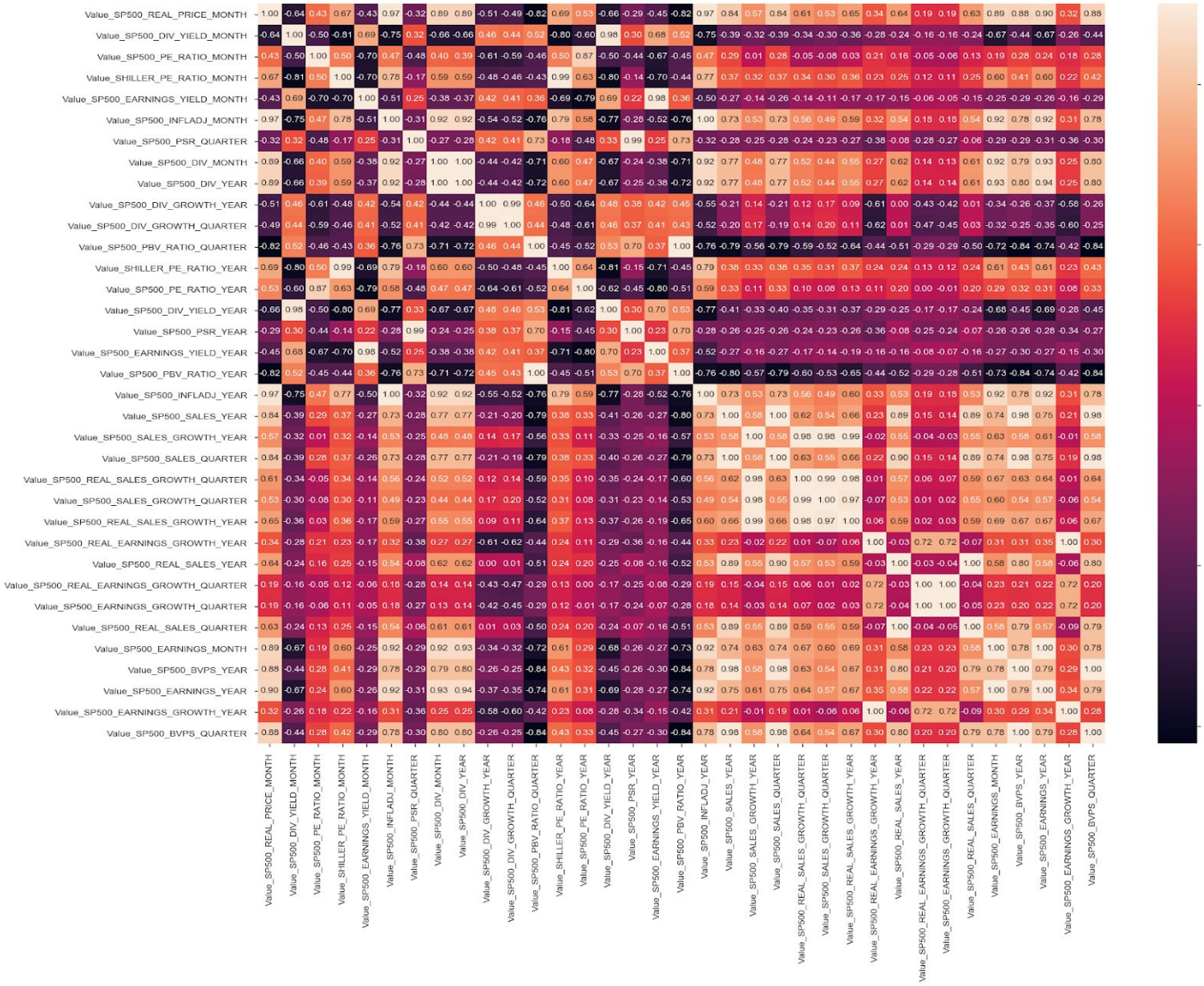
`Value_SHILLER_PE_RATIO_YEAR` is highly correlated with `Value_SHILLER_PE_RATIO_MONTH` (ρ = 0.98988)

- `Value_SP500_BVPS_QUARTER` is highly correlated with `Value_SP500_BVPS_YEAR` (ρ = 0.99959)
- `Value_SP500_BVPS_YEAR` is highly correlated with `Value_SP500_SALES_QUARTER` (ρ = 0.97758)
- `Value_SP500_DIV_GROWTH_QUARTER` is highly correlated with `Value_SP500_DIV_GROWTH_YEAR` (ρ = 0.99279)
- `Value_SP500_DIV_MONTH` is highly correlated with `Value_SP500_INFLADJ_MONTH` (ρ = 0.91953)
- `Value_SP500_DIV_YEAR` is highly correlated with `Value_SP500_DIV_MONTH` (ρ = 0.99955)
- `Value_SP500_DIV_YIELD_YEAR` is highly correlated with `Value_SP500_DIV_YIELD_MONTH` (ρ = 0.97675)
- `Value_SP500_EARNINGS_GROWTH_QUARTER` is highly correlated with `Value_SP500_REAL_EARNINGS_GROWTH_QUARTER` (ρ = 0.99925)
- `Value_SP500_EARNINGS_GROWTH_YEAR` is highly correlated with `Value_SP500_REAL_EARNINGS_GROWTH_YEAR` (ρ = 0.99856)
- `Value_SP500_EARNINGS_MONTH` is highly correlated with `Value_SP500_INFLADJ_YEAR` (ρ = 0.91628)
- `Value_SP500_EARNINGS_YEAR` is highly correlated with `Value_SP500_EARNINGS_MONTH` (ρ = 0.99515)
- `Value_SP500_EARNINGS_YIELD_YEAR` is highly correlated with `Value_SP500_EARNINGS_YIELD_MONTH` (ρ = 0.98185)
- `Value_SP500_INFLADJ_MONTH` is highly correlated with `Value_SP500_REAL_PRICE_MONTH` (ρ = 0.96798)
- `Value_SP500_INFLADJ_YEAR` is highly correlated with `Value_SP500_DIV_YEAR` (ρ = 0.91992)
- `Value_SP500_PBV_RATIO_YEAR` is highly correlated with `Value_SP500_PBV_RATIO_QUARTER` (ρ = 0.99857)
- `Value_SP500_PSR_YEAR` is highly correlated with `Value_SP500_PSR_QUARTER` (ρ = 0.9885)
- `Value_SP500_REAL_SALES_GROWTH_QUARTER` is highly correlated with `Value_SP500_SALES_GROWTH_YEAR` (ρ = 0.97961)
- `Value_SP500_REAL_SALES_GROWTH_YEAR` is highly correlated with `Value_SP500_SALES_GROWTH_QUARTER` (ρ = 0.96655)
- `Value_SP500_REAL_SALES_QUARTER` is highly correlated with `Value_SP500_REAL_SALES_YEAR` (ρ = 0.99503)
- `Value_SP500_SALES_GROWTH_QUARTER` is highly correlated with `Value_SP500_REAL_SALES_GROWTH_QUARTER` (ρ = 0.99013)
- `Value_SP500_SALES_QUARTER` is highly correlated with `Value_SP500_SALES_YEAR` (ρ = 0.99942)

Correlation between variables/features after interpolation of missing values-

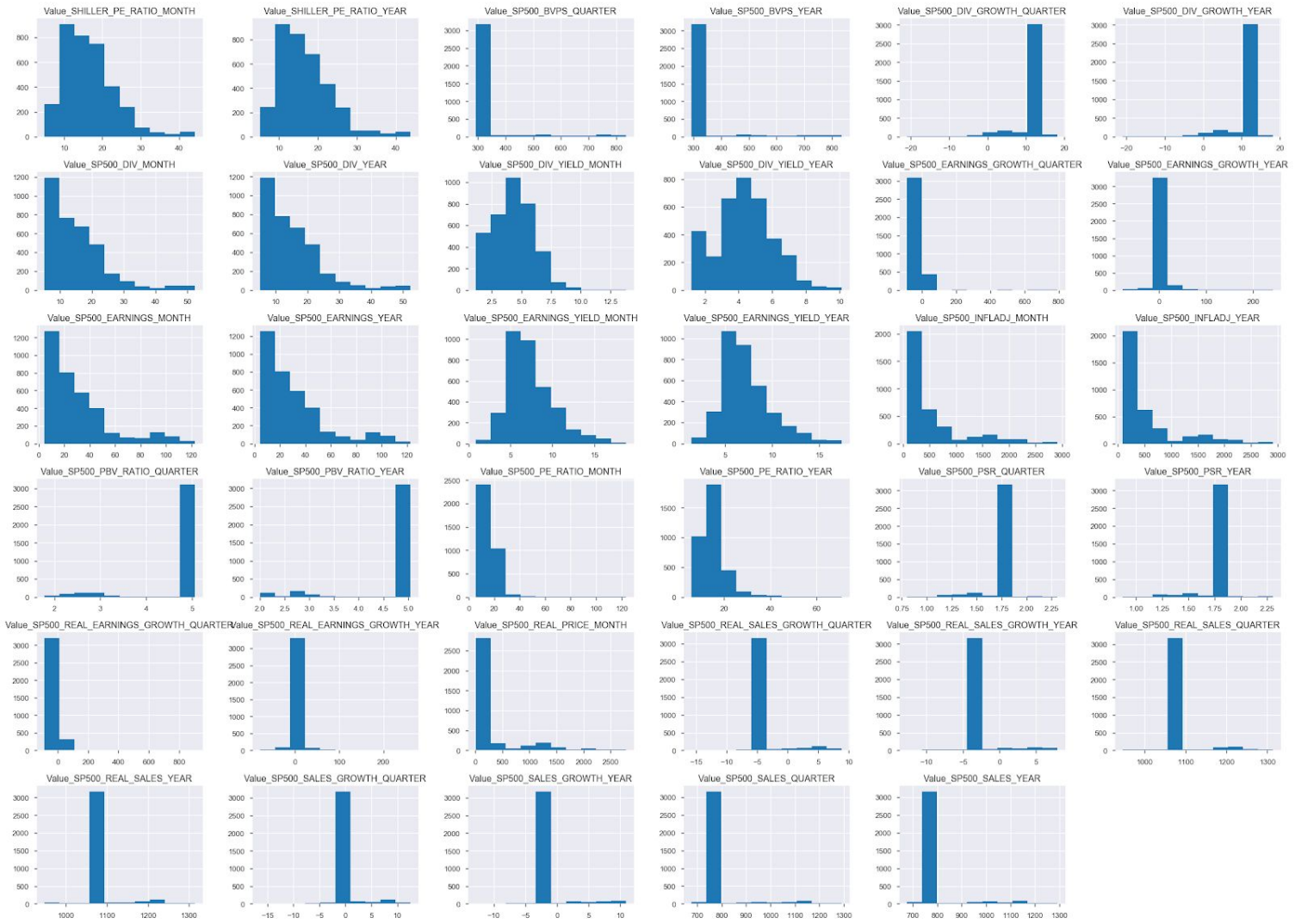
```
#Correlation heatmap plot after interpolation
plt.figure(figsize=(20,15))
sns.heatmap(df_interpolate.corr(),annot=True,fmt='.2f',square=False)
```

Correlation heatmap of variables is better than imputed correlation in interpolated data. There is strong relationship between many variables in the dataset which would help in feature engineering further on. Spearman correlation shows better results for interpolated data too.



Distribution of interpolated values for all variables-

Histogram shows distribution of values for interpolated values. For many variables, data distribution is skewed right except of a few where data is unimodal. This would affect fitting the regression predicted line during implementation of model. Features have to carefully selected.



Final analysis on interpolation and imputation datasets-

Linear Interpolation on missing values looks promising and there is better correlation between variables of the dataset. Imputation has not shown better correlation between variables. I would use interpolation dataframe for further analysis.

Algorithms and Techniques

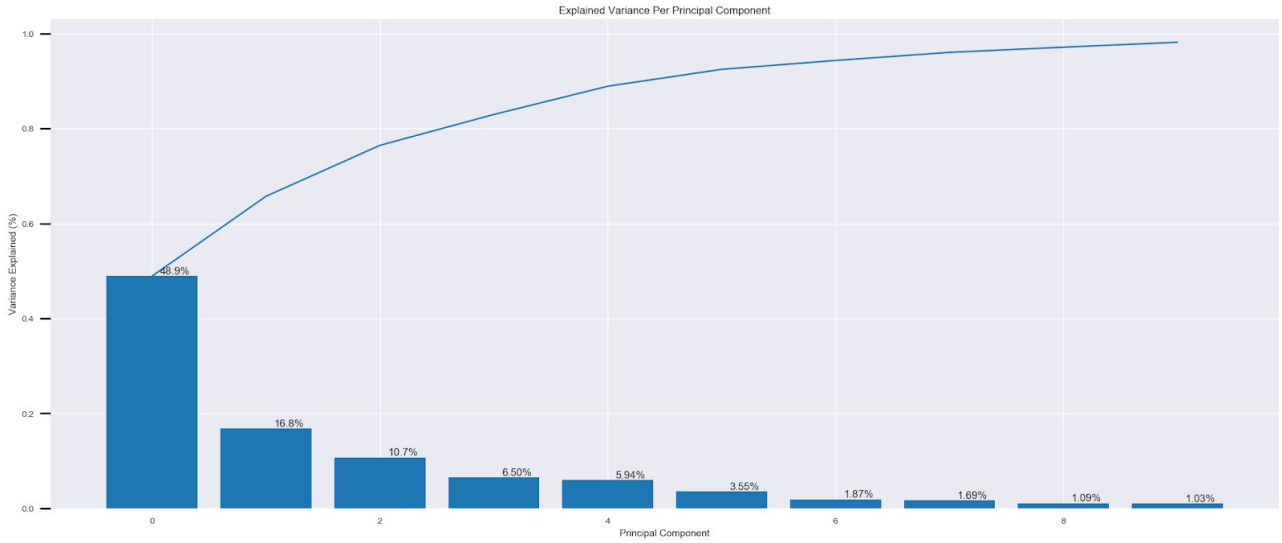
PCA- Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of

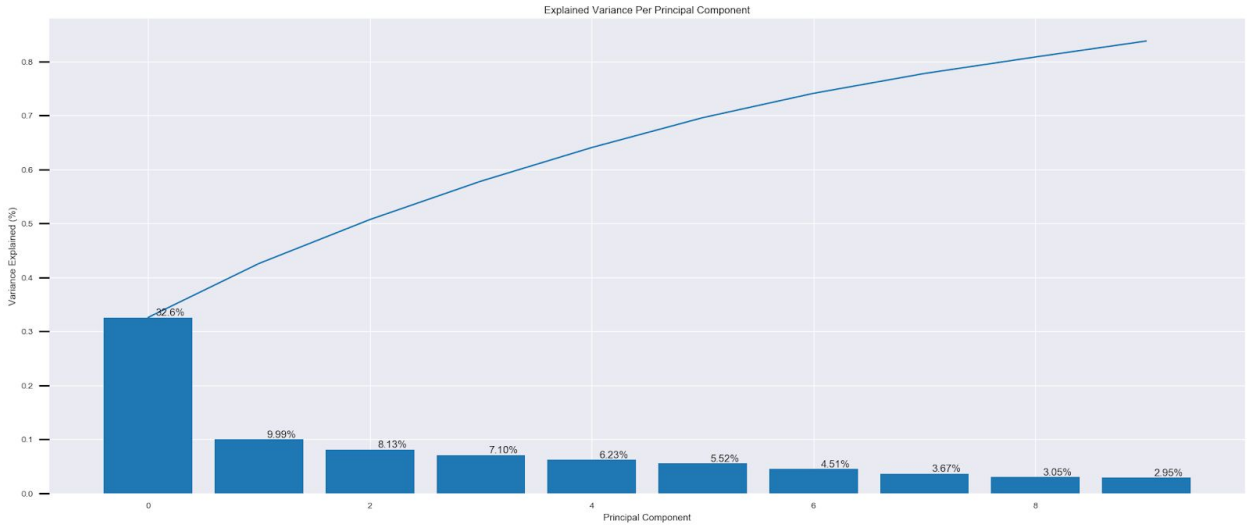
which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.

I have scaled usig StandardScalar() all variables values from interpolated dataset before I used PCA to get an idea about variance of all 35 variables except for target variable `'Value_SP500_REAL_PRICE_MONTH'` on interpolated dataset. I can use the variables from first cluster for modelling in Linear Regression, LSTM and fbProphet models.

PCA cluster variance using interpolated dataset with 10 clusters shows strong variance after 1st and 2nd clusters. I would use it for modelling further.



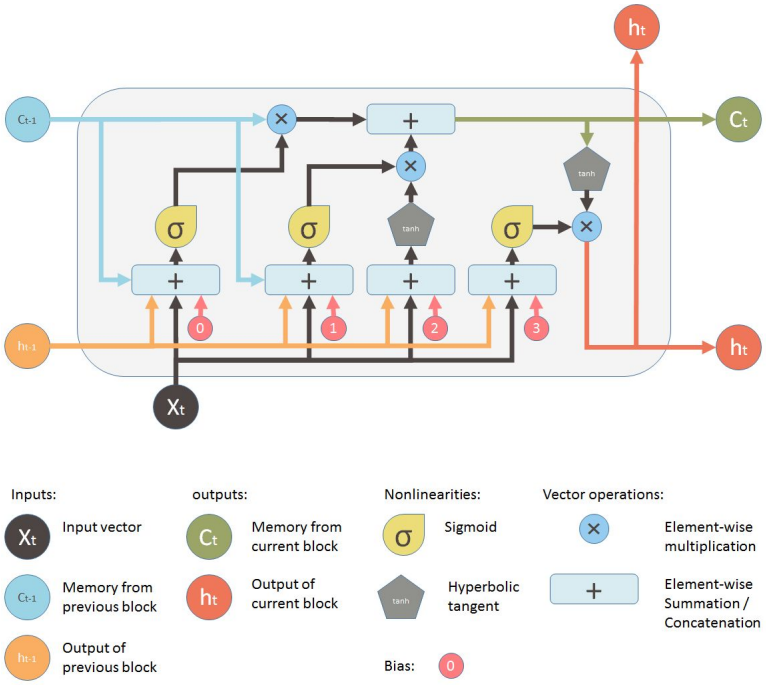
PCA cluster variance using imputed dataset - there is not clear variance between clusters using imputed dataset. I would not use imputed dataset for modelling further on.



Algorithms-

LSTM-

Neural Network model, with the type of model being a Recurring Neural Network (RNN). Under the RNN scope there are the LSTM algorithms that have recently been incorporated in various stock predicting algorithms and strategies because of their effectiveness compared to other neural algorithms. Long-term dependency is a problem that occurs in RNN when the network is in the need of making a prediction that requires context. In a regular RNN the need of understanding the context can be handled but this is dependent on how far back the memory needs to save the information for the context. LSTM's have a chain structure and on the inside they operate using gates and layers of neural networks like other RNN approaches. The structure of the LSTM is constructed in a manner of a cell state that runs through the entire LSTM, the value is changed by the gates that have function by either allowing or disallowing data to be added to the cell state. There are also components by the name of gated cells that allow the information from previous LSTM outputs or layer outputs to be stored in them, this is where the memory aspect of LSTM's kick in (Hochreiter and Schmidhuber, 1997).



LSTM references-

https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

https://en.wikipedia.org/wiki/Long_short-term_memory

https://medium.com/@huangkh19951228/predicting-cryptocurrency-price-with-tensorflow-and-keras-e1674b0dc58a

https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877

fbProphet-

How Prophet works-

At its core, the Prophet procedure is an additive regression model with four main components:

- A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
- A yearly seasonal component modeled using Fourier series.
- A weekly seasonal component using dummy variables.
- A user-provided list of important holidays.

References-

https://research.fb.com/prophet-forecasting-at-scale/

https://towardsdatascience.com/analysis-of-stock-market-cycles-with-fbprophet-package-in-python-7c36db32ecd0

Benchmark

Linear regression model

I have considered simple linear regression model as benchmark model. It is easy and simple to understand a regression problem with this model for prediction, forecasting purposes. I would train the model with 90% of data and test on 10% because the real price has increased in the past few years logarithmically. As mentioned by me, I couldnt find any available benchmark model for similar dataset for S&P 500 index price.

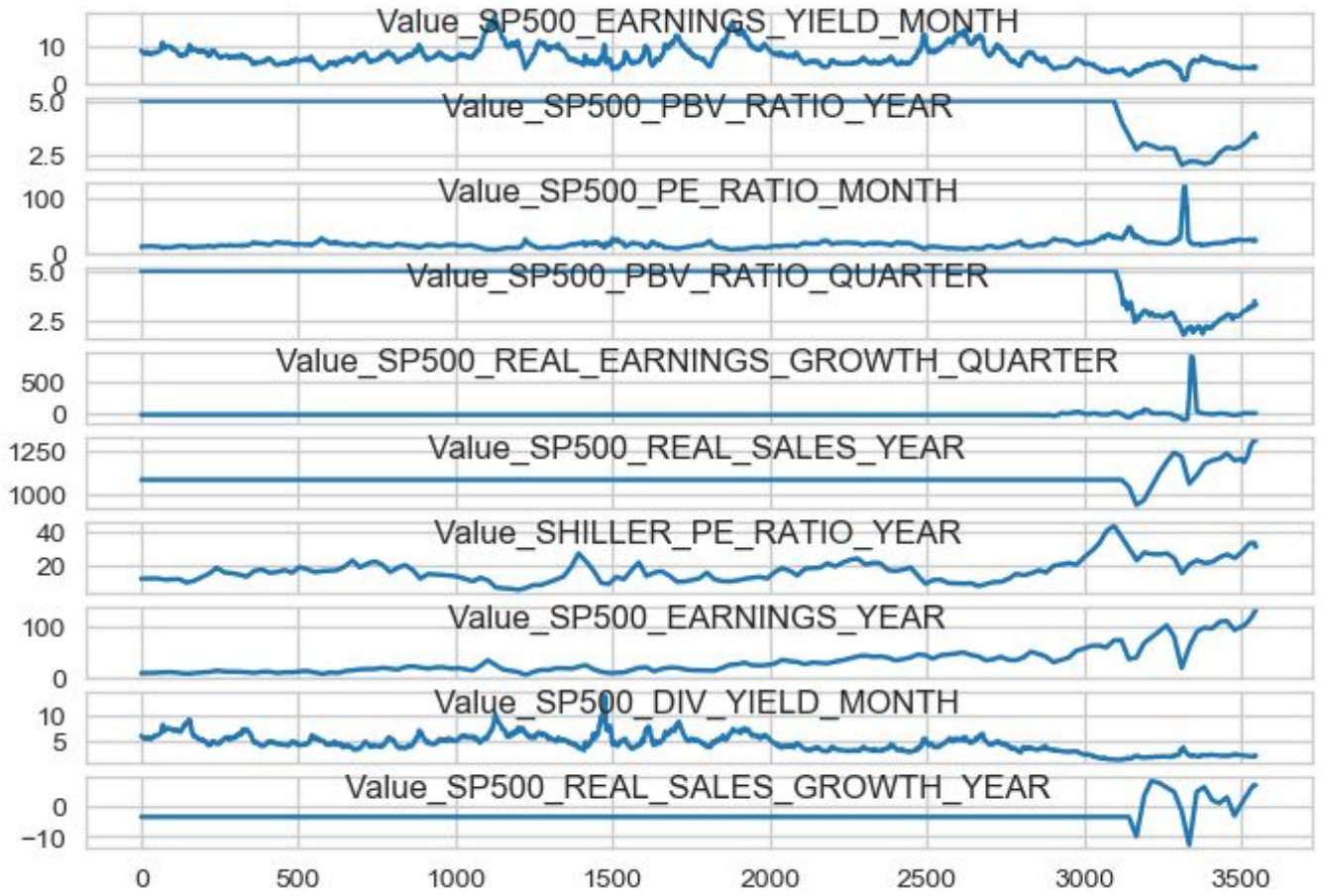
Benchmark results from the selected monthly variables dataset shows high mean square error. Even after working with features through PCA and selecting monthly variables, results from Linear Regression are not good. MSE is too high.

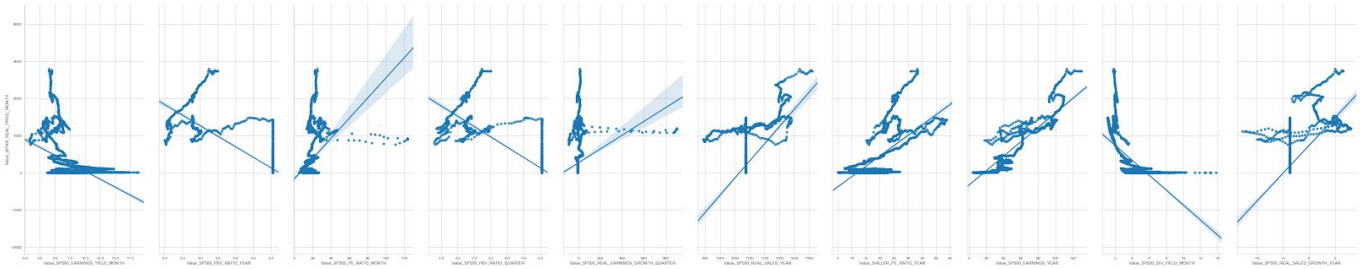
Plotting the Least Squares Line

Best observed high variance features Vs target label 'Value_SP500_REAL_PRICE_MONTH' to understand linear corelation.This is to find colinearity between features and the prediction line for selected set of features. Predicted line would try to fit with linear regression model through

the feature data points.The dataset here is very imbalanced and non linear which makes it different to predict a linear line with optimum coefficients and intercept.

PCA components timeseries plot of interpolated dataset.





From the above pairplot it is clear that the data is unbalanced and skewed and not linear in nature. Predicted linear regression line has different slopes with respect to PCA components.

Reference-

https://en.wikipedia.org/wiki/Linear_regression

III. Methodology

Data Preprocessing

split and slicing, PCA components.

Scaling-

I have used Standard Scaling function from Scikit-learn library.

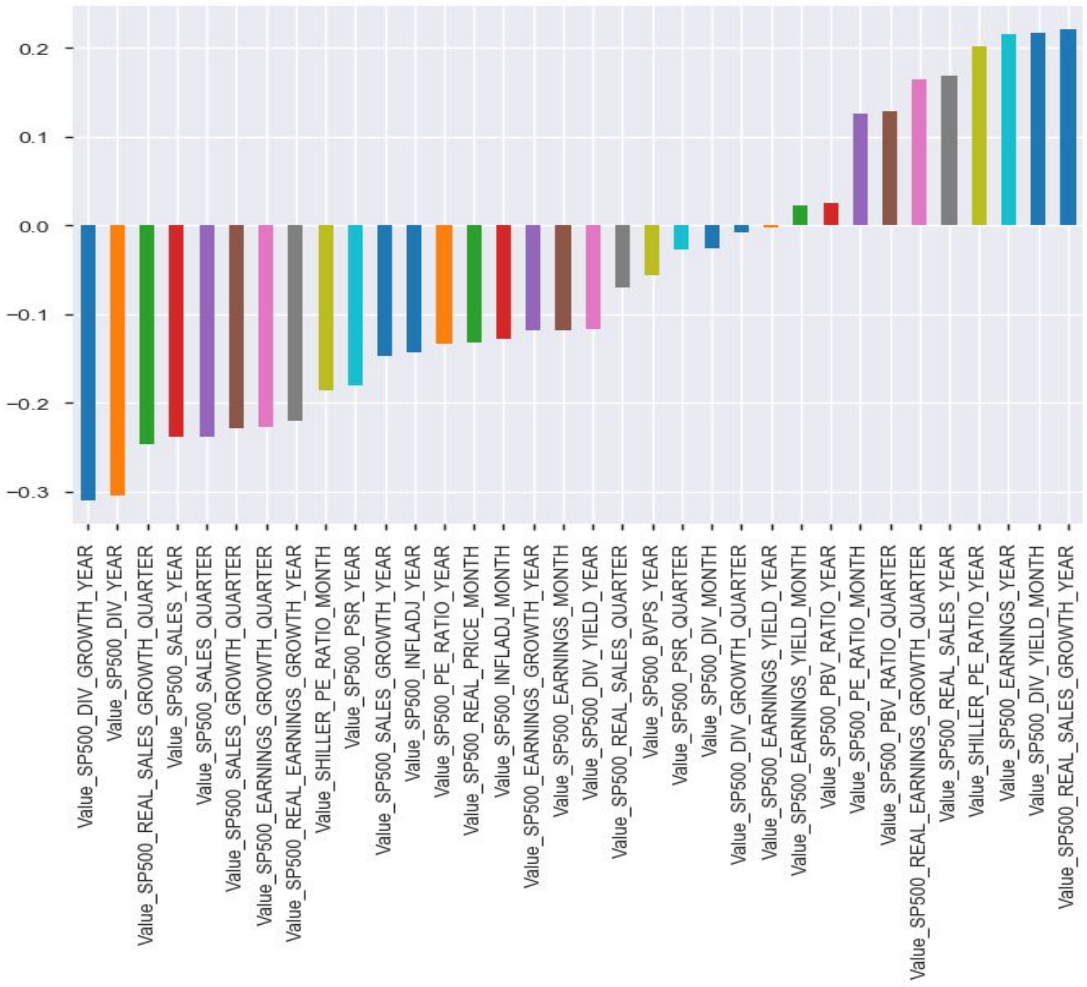
```
from sklearn.preprocessing import StandardScaler
```

PCA components-

I have calculated and shown sorted weights of components in PCA cluster with highest variance. Principal compoments are -

```
['Value_SP500_REAL_EARNINGS_GROWTH_QUARTER',
 'Value_SP500_REAL_SALES_YEAR',
 'Value_SHILLER_PE_RATIO_YEAR',
 'Value_SP500_EARNINGS_YEAR',
 'Value_SP500_DIV_YIELD_MONTH',
 'Value_SP500_REAL_SALES_GROWTH_YEAR']
```

Timeseries of PCA features for interpolated dataset.



Plot of all principal components for interpolated dataset

Split and slicing-

I have split the interpolated dataset into a subset dataset with selected variable labels for LSTM and Linear regression purpose but I did not get expected results even if I use PCA components or monthly variables from the dataset. I split the 98% for training and 2% for prediction.

```
df_PCA_features = df_interpolate.loc[:,['Value_SP500_REAL_PRICE_MONTH',
                                        'Value_SP500_DIV_YIELD_MONTH',
                                        'Value_SP500_PE_RATIO_MONTH',
                                        'Value_SHILLER_PE_RATIO_MONTH',
                                        'Value_SP500_EARNINGS_YIELD_MONTH',
                                        'Value_SP500_INFLADJ_MONTH',

'Value_SP500_EARNINGS_MONTH','Value_SP500_PSR_QUARTER','Value_SP500_SALES_QUARTE
R',
```

```
'Value_SP500_REAL_SALES_GROWTH_QUARTER','Value_SP500_REAL_EARNINGS_GROWTH_QUARTE
R']]
df_PCA_features.shape
(3548, 11)
```

Training and Test data size-

```
Training and Test dataset is of size 3477 & 71
Features size of X_train and training target Y_train shape is (3477, 10) &
(3477, 1)
Features size of X_test and Test target Y_test shape is (71, 10) & (71, 1)
Training dataset is converted to np.array with size (3477, 10) & (3477, 1)
Test dataset is converted to np.array with size (71, 10) & (71, 1)
```

I split the interpolated dataset chronologically to avoid look-ahead bias in the timeseries. Index price values for last 71 months (about 6 years looking back) shows a steep increase in real price curves. I changed the dataframe values to np.array in preprocessing step before feeding it to LSTM network even though it is not required because LSTM accepts pandas dataframe as input to the network.

Implementation

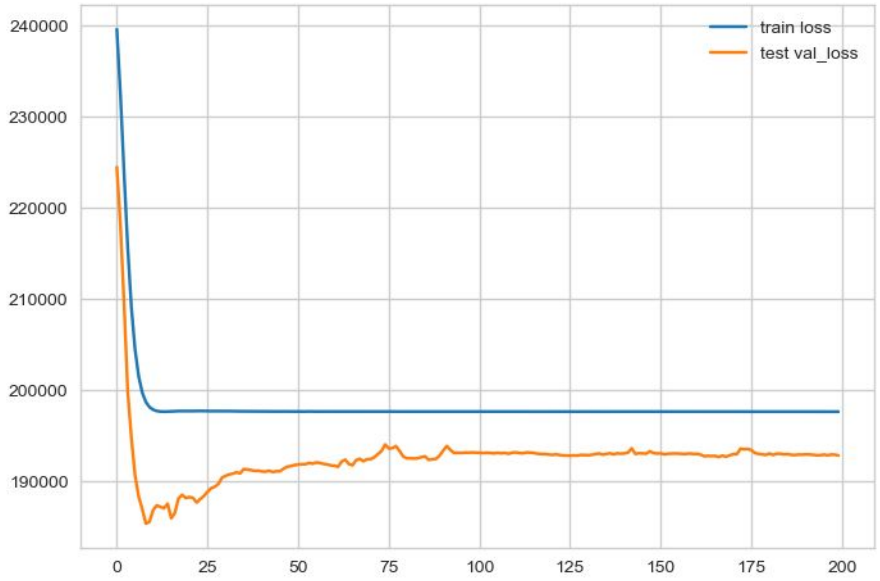
LSTM(Long Short-term Memory)

I have designed the network with Sequential model, 512 as input to hidden LSTM layers. Then I have added dropout of 20% with BatchNormalization in hidden layer which would normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

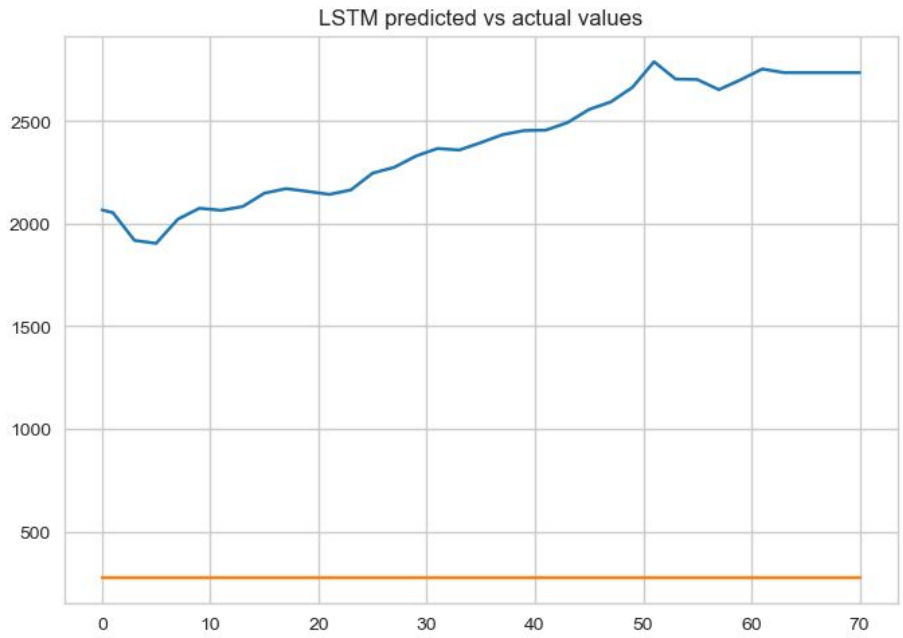
I used ‘relu’ activation function before Dense model and ‘Softmax’ activation function in output as standard configuration for LSTM models. Checkpoint would save the model for best results.

```
#Design network for mean squared error regression problem
model = Sequential()
#The first dimension is supposed to be each sample.input should be (n_samples,
timesteps, n_features)
model.add(LSTM(512, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(BatchNormalization())
# model.add(LSTM(512,
input_shape=(X_train.shape[0],X_train.shape[2]),return_sequences=True))
# model.add(Dropout(0.1))
# model.add(BatchNormalization())
```

```
Activation('relu')
model.add(Dense(1))
Activation('softmax')
# Compiling the model using mean square error loss, and Adam optimizer.
model.compile(loss='mse', optimizer='adam',metrics=['accuracy'])
filepath = "RNN_final-{epoch:02d}-{val_acc:.3f}"#unique file name which will
include epochs and validation accuracy score.
checkpoint = ModelCheckpoint("models/{}.model".format(filepath,monitor =
'val_acc',verbose = 1))

# fit network with epochs
history = model.fit(X_train, Y_train, epochs=200, batch_size=100,
validation_data=(X_train, Y_train), verbose=2,
                callbacks = [checkpoint],
                shuffle=False)
model.save('LSTM_model',overwrite=True,include_optimizer=True)
# plot history
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='test val_loss')
plt.legend()
plt.show()
```



Train data vs test validation loss plot



Predicted price of S&P 500 index was way below the actual prices for last 71 months using PCA variables with highest variance or with selected features with monthly ratios.

Difficulties with LSTM-

LSTM network is difficult to create because in my case Dense layer does not take more than 1 estimator and I could understand why. I had difficulties with defining input size and defining more hidden layers with LSTM network. The network I could build is very basic and minimum functional neural network.

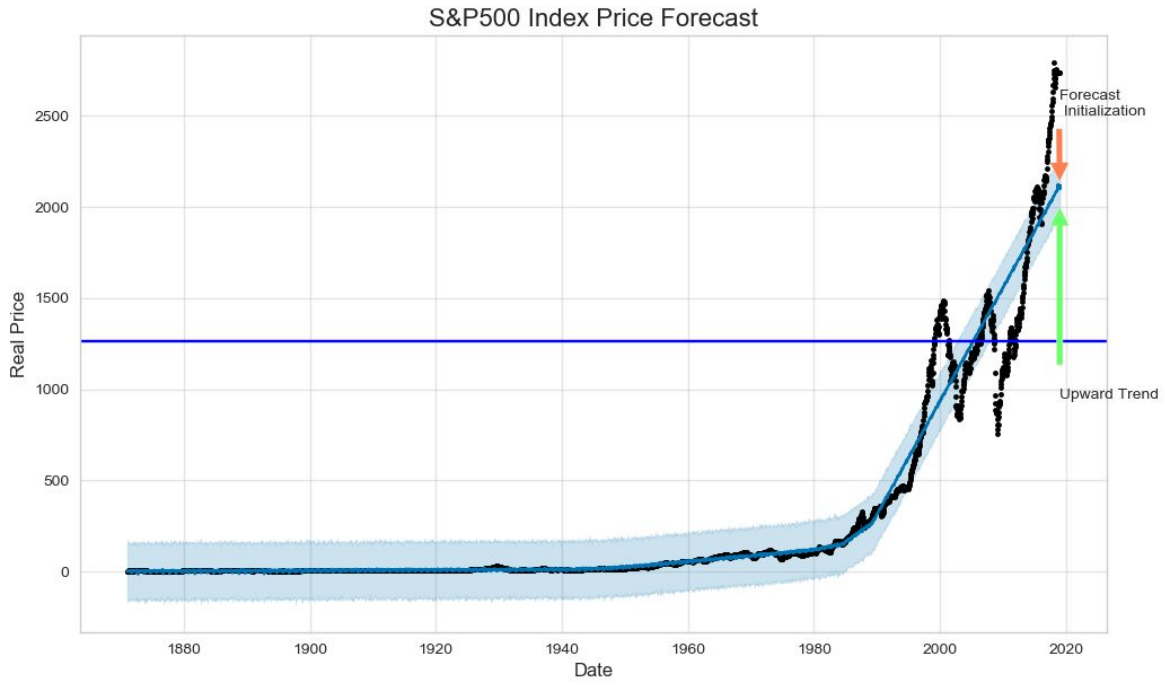
Refinement

I tried to refine LSTM model with different combinations of features but didn't get expected results. LSTM model could be refined with multiple hidden layers and different activation functions. There is clearly huge scope of improvements with better business knowledge on features and LSTM expertise to make the neural network train as much better.

For refinement I tried fbProphet model and the results are not better either. fbProphet model takes 2 labels in a dataframe- time-series and forecast label.

```
from fbprophet import Prophet

#Create future dates
```

```
future_dates = fb.make_future_dataframe(periods=30)

#Predict prices for future dates

future_price = fb.predict(future_dates)
```



Forecasted price using fbProphet-

```
future_price[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
3573	2019-01-26	2104.546587	1937.262720	2262.138696
3574	2019-01-27	2109.010824	1954.873221	2261.450394
3575	2019-01-28	2109.505319	1951.637642	2271.802746
3576	2019-01-29	2111.732478	1958.432705	2263.845634
3577	2019-01-30	2115.552914	1957.799005	2267.874511

IV. Results

Model Evaluation and Validation

LSTM model evaluation-

```
Means Square Error between Y_test adn prediction values is : 1586328991228.7246
R2 score between Y_test adn prediction value is : -73.4024904609074

Results from training-
- 2s - loss: 195472.0549 - acc: 0.0000e+00 - val_loss: 182259.5592 - val_acc:
0.0000e+00
Epoch 172/200
 - 2s - loss: 195474.6800 - acc: 0.0000e+00 - val_loss: 182338.4478 - val_acc:
2.8760e-04
Epoch 173/200
 - 2s - loss: 195510.6922 - acc: 0.0000e+00 - val_loss: 182226.1746 - val_acc:
0.0000e+00
Epoch 174/200
 - 2s - loss: 195486.7277 - acc: 0.0000e+00 - val_loss: 182274.0971 - val_acc:
2.8760e-04
```

Linear Regression evaluation-

```
Mean squared error: 1076172.6990871658
Root Mean squared error: 1037.3874392372243
Variance R2 score: -1.9382920775190278
```

fbProphet evaluation-

```
Mean squared error: 1279401.5658830237
```

Evaluations of both models are very bad quality and results should not be trusted by any means. This dataset and project requires in depth evaluation and may be additional features for expected results.

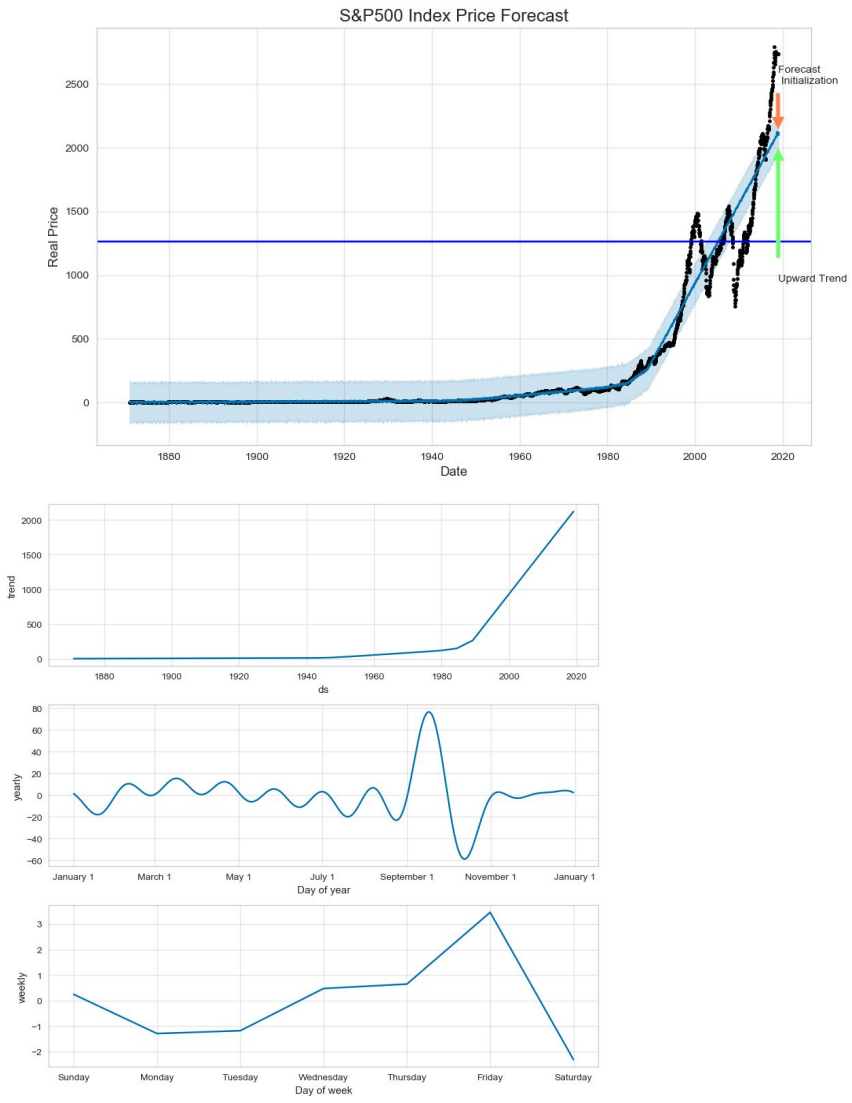
Justification

This dataset and project required in depth review of dataset and greater tuning of LSTM network. I could not find any justification or clear answer to why predicted values are way less than actual leading to huge error, 0 accuracy and high training/validation loss.

Both benchmark results from Linear regression model and LSTM are bad.

V. Conclusion

Free-Form Visualization



S&P 500 index price could not be predicted as expected partly due to dataset (possibly) and also because I could not train LSTM correctly. This project requires better features and more data to forecast the price correctly.

Reflection

My reflections on this project is that dataset is very unbalanced and skewed and features are not clearly related with each other. It is difficult to find any patterns in the dataset which links with the real price of index. More work on correcting the dataset is required and feature engineering requires more time and business knowledge. Business knowledge is clearly missing in this project.

Feature engineering has been challenging with this dataset specially with skewed variables and missing values in time-series.

Improvement

Handling missing values might make the difference in predicting the index price in future. Also, constituent data of underlying stocks would help in selecting correct features too. Features like, trading volume, high price, low price, open price, closing price of underlying constituent stocks would definitely help in feature engineering.
