Predicting Returns from Stock Portfolio Using Machine Learning Algorithms.

Project Objectives :

In this project, We first identify the optimal allocation of capital for a portfolio of stock returns based on value versus risk using Sharpe Ratio. We are considering stocks from Bombay Stock Exchange with data over the past decade. We simulate random allocation of stock and use Sharpe ratio to choose the optimal portfolio with maximum return and minimum risk or volatility. Once we get the allocation ratio for each stock in the portfolio, we compute the historic returns for that allocation and build a model on that historical data to predict further price movement of the portfolio using Linear Regression and Long and Short term Neural Networks.

We plan to use various computational techniques to identify the ideal allocation of capital in a portfolio of stocks using efficient frontier technique. Then we employ machine learning model to predict the value of portfolio. We eventually would want to calibrate the different allocation ratio and the machine learning model to predict the movement of portfolio.

Summary of Contributions:

- Fetched Data from API. Performed Visualization and Scaled Data. Implemented efficient frontier for computing optimal stock allocation.
- Trained Linear Regression and LSTM models on data and developed metrics on these models.
- Worked out on optimal parameters to fine tune the model and achieve high accuracy.

Techniques and Tools:

- Yahoo Finance Library: To fetch Stock data of different companies listed in Bombay Stock Exchange, fetched data of past decade with daily Open, Close, High, Low and Volume.
- Pandas and Numpy: For Data handling and manipulation.
- Matplotlib and Seaborn: For Visualization.
- Sharpe Ratio: The Sharpe ratio reveals the average investment return, minus the risk-free rate of return, divided by the standard deviation of returns for the investment.
- Efficient Frontier: The efficient frontier is the set of optimal portfolios that offer the highest expected return for a defined level of risk or the lowest risk for a given level of expected return. Portfolios that lie below the efficient frontier are sub-optimal because they do not provide enough return for the level of risk.
- Linear Regression: Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).
- Ordinary Least Squares: This procedure seeks to minimise the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimise. This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.

- Recurrent Neural Networks:
  RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer. We introduce recurrent neural network as a normal feed forward network cannot handle sequential data and cannot memorise previous inputs. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorise previous inputs due to their internal memory.

- Long and Short Term Neural Network:
  Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory.

- Sklearn:
  Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

- Keras:
  Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function.

Results and Conclusions:

Linear Regression:

Results of Linear Regression forecasting next 100 trading days.
Prediction Using LR
Values Forecasted : 100
Model Name : LR
Train RMSE : 0.0040086347488074115
Train R2 Score : 0.9986504871480923
Test RMSE : 0.012948160848910394
Test R2 Score : 0.9950279876239176
####################

####################
Results of Linear Regression forecasting next 200 trading days.
Prediction Using LR
Values Forecasted : 200
Model Name : LR
Train RMSE : 0.003979731795015481
Train R2 Score : 0.9985471381002259
Test RMSE : 0.014009988551964943
Test R2 Score : 0.9939035902067581
####################

####################
Results of Linear Regression forecasting next 300 trading days.
Prediction Using LR
Values Forecasted : 300
Model Name : LR
Train RMSE : 0.003914788479597741
Train R2 Score : 0.9984520407192581
Test RMSE : 0.013985608797798716
Test R2 Score : 0.9885610692392469
####################

####################
Results of Linear Regression forecasting next 400 trading days.
Prediction Using LR
Values Forecasted : 400
Model Name : LR
Train RMSE : 0.003715769599636717
Train R2 Score : 0.9984202928808295
Test RMSE : 0.015337175785080951
Test R2 Score : 0.9721351377701888
####################

####################
Results of Linear Regression forecasting next 500 trading days.
Prediction Using LR
Values Forecasted : 500
Model Name : LR
Train RMSE : 0.003603825035967074
Train R2 Score : 0.9982365057041702
Test RMSE : 0.017195000469283885
Test R2 Score : 0.9110671312117862
####################

####################
Results of Linear Regression forecasting next 600 trading days.
Prediction Using LR
Values Forecasted : 600
Model Name : LR
Train RMSE : 0.0032871086917017644
Train R2 Score : 0.9983526045346967
Test RMSE : 0.022054269727648443
Test R2 Score : 0.6521293413304413
####################

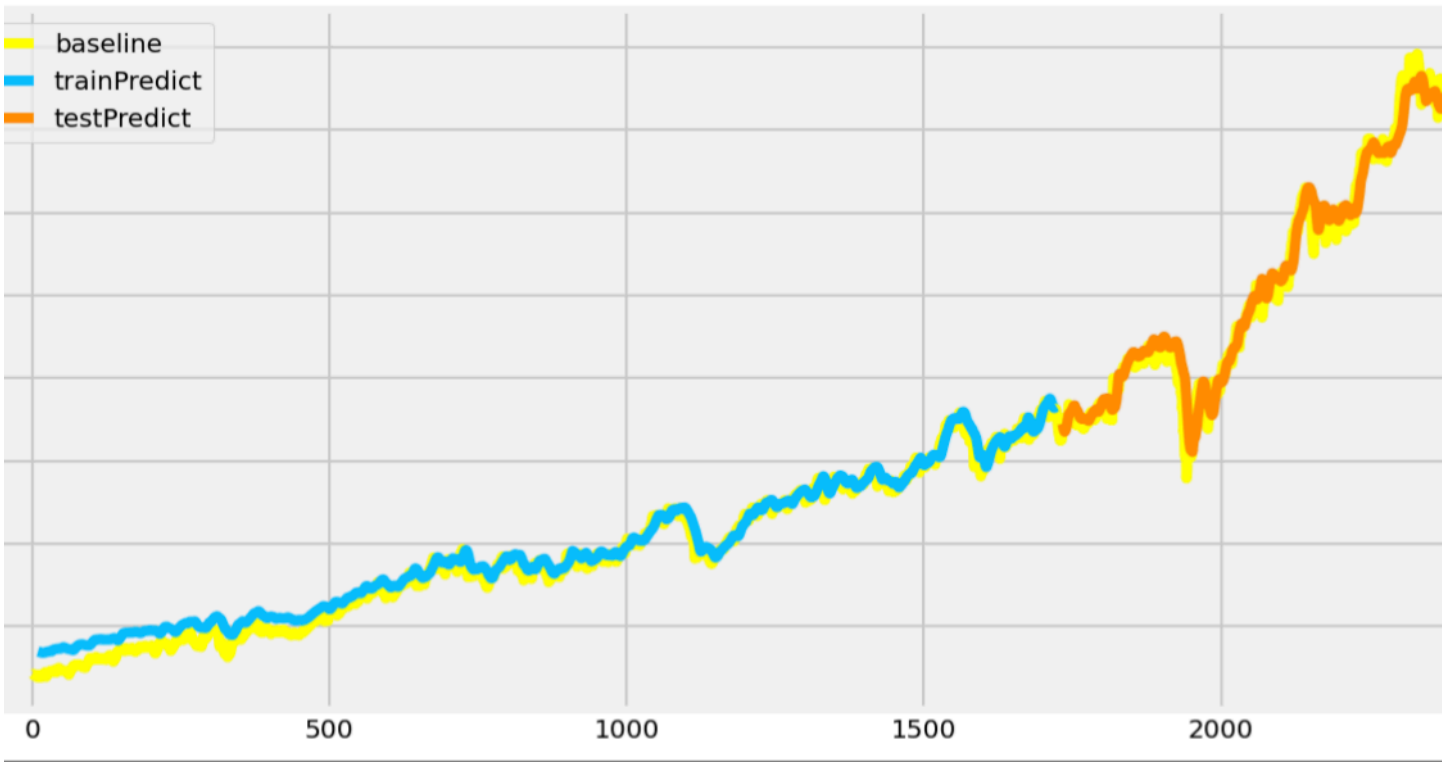
####################
Results of Linear Regression forecasting next 700 trading days.
Prediction Using LR
Values Forecasted : 700
Model Name : LR
Train RMSE : 0.00292799338755662
Train R2 Score : 0.9986032839539296
Test RMSE : 0.026990435683838222
Test R2 Score : 0.6105832216915619
####################

####################

Long and Short Term Neural Networks:

Metrics for train and test datasets:

train_mse = 0.004419493016489919
test_mse = 0.012377929999732707



Long and Short Term Neural Network Model Prediction

train_rmse = 0.06647926756884374
test_rmse = 0.11125614589645243

train_mae = 0.05310097557756166
test_mae = 0.08351598745942392

train_r2 = 0.9777689308163915
test_r2 = 0.978556546018192

Conclusion:

While Linear Regression is performing on par with LSTM, LSTM is better at predicting both outliers and central values simultaneously. It's able to perform predictions accurately for longer extend to trading days. Hence this project accomplished to find optimal allocation of capital for stocks and tried to predict the portfolio's closing value in the future trading days.

References:

https://www.machinelearningplus.com/machine-learning/portfolio-optimization-python-example/