Capstone Project

Data Science Nanodegree

Mayukh Sil

April 18, 2020

BUILD A STOCK PRICE INDICATOR

Definition

Project Overview:

This project will train a model that can establish stock price indicators based on stocks and funds historical trends. Models can assist stock market traders to help them make decisions about stocks and funds trading, and more.

This project involves the XGB, CatBoost And LGBM Regressors models suitable for processing and predicting important events with relatively long intervals and delays in time series.

The data set selected by this project comes from Yahoo Finance, there is an API which can be used to access historical stock prices also we can manually download historical prices from Yahoo Finance website, for purpose of this project I have downloaded historical stock price of ITC Limited which is listed in National Stock Exchange And Bombay Stock Exchange, India.

Problem Statement:

In this project we are trying to predict the future price of stock 7 Days in advance using historical data. Predicting future prices are quite complicated. The project expects to train a model and use it to build a stock price indicator with a 7-day average error rate of less than 5%.
This project is divided into 3 steps: First an ETL pipeline which takes historical data cleans it and computes various technical indicators and store them into a database, second is a ML

modelling script which trains three of best regressors available i.e. XGB, LGBM and Catboost. Finally saving those trained models into a pickle file which can be used to predict prices later, third is a web application which can be used to predict the price of stock.

Metrics:

This project uses a regression model to build a stock price indicator. The following are commonly used for regression model indicators:

1. Mean Square Error (MSE)
2. Mean Absolute Error (MAE)
3. Huber Loss
4. Root Mean Squared Error
5. Quantile
6. Root Mean Squared Logarithmic Error

Models are trained using Root Mean Squared Error as the metric to optimize.

Once trained, I computed the R^2 score And Mean Absolute Error on the validation set.

Root Mean Square Error (**RMSE**) is the standard deviation of the residuals (prediction **errors**). Residuals are a measure of how far from the regression line data points are; **RMSE** is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

The **coefficient of determination**, denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

Mean Absolute Error is a commonly used regression loss function. MAE is the sum of absolute distances between our target variable and predicted values.

Analysis:

Data Exploration:

The data set selected by this project comes from Yahoo Finance. This website provides a query of the price history of each stock and fund, including daily open, close, high, low and volume data. This project chooses to use the historical trend data of the stock whose code is **ITC.NS** as the data set for training. The data set contains a total of 3702 working days of historical data.

The data set contains 7 features, respectively, date, opening price (open), highest and lowest price the stock traded at (high and low), how many stocks were traded (volume) and closing price adjusted for stock splits and dividends (close). The following chart shows the daily price trend of the **ITC.NS**.

Methodology:

Data Preprocessing:

Filling missing values:

Yahoo Finance API may return missing data for a certain period of time, because during this time the stock exchange suspends the stock to trade on the stock market. The reason for this may be that the stock is due to a certain message or a certain Activities caused a continuous rise or fall in stock prices. After the situation is clarified or the company returns to normal, the stock will be listed on the exchange. I want to fill in the missing values. If there is price or volume data before, fill it with the previous value. If there is no price or volume data before, it will be filled according to the latter value.

Logarithmic processing:

Because the stock price is roughly logarithmically distributed, so it is better to use a logarithmic processing on stock historical data. After logarithmic processing, the distance between the y-axis coordinate scales is proportional to the price increase rate.

Generating input and output data:

This project first uses 7 working days of data as a training window, and trains a XGB,LGBM And Catboost Regressor to predict the closing price of the 7th working day after the end of the training window. First, I produced the input tensor. The 2D tensor is the original data set, and the dimensions are samples and features. The 2D tensor is the processed input tensor, the dimensions are samples, timesteps and features. For each of the 2D tensors, it contains an entire timesteps of data, which is all data for 7 consecutive working days. Then, the label corresponding to each sample (the closing price after five working days of the last day in timesteps) is combined into a one-dimensional tensor.

Splitting Data

Finally, the sample and its label are then divided into training data and testing data at a scale of 0.9 and 0.1. Then I shuffled their order.

Implementation:

Algorithms:

Time series:

A time series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus, it is a sequence of discrete-time data.

This time-meaning sequence is also called dynamic data. Dynamic data is very common in the natural, economic and social fields. For example, the number of animal and plant populations is increasing month by month or year by year under certain ecological conditions, the daily closing index of a stock exchange, the gross national product per month, the number of unemployed or the price index, and so on.

Catboost Regressor:

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.
"CatBoost" name comes from two words "Category" and "Boosting".

As discussed, the library works well with multiple Categories of data, such as audio, text, image including historical data.

"Boost" comes from gradient boosting machine learning algorithm as this library is based on gradient boosting library. Gradient boosting is a powerful machine learning algorithm that is widely applied to multiple types of business challenges like fraud detection, recommendation

items, forecasting and it performs well also. It can also return very good result with relatively less data, unlike DL models that need to learn from a massive amount of data.

LGBM Regressor:

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

XGBoost Regressor:

XGBoost stands for e**X**treme **G**radient **B**oosting.
The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost.
It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library.

Techniques:

This project uses Xgboost , LGBM and Catboost Libraries to train 3 regressors on the stock dataset and finally after training the prediction of each of the 3 regressors are added and is divided by 3 to get the mean prediction value. Also, the data which is fed into the regressors is transformed log scaled data along with different technical indicators which are computed using the price of the stock. These technical indicators are good features in predicting the future price of a stock.

Training model:

- First I trained a XGB Regressor, on the stock data with n_estimators=10000,min_child_weight= 40,learning_rate=0.01,colsample_bytree = 1,subsample = 0.9 which gave validation_0-rmse:0.03738.

```
reg = xgb.XGBRegressor(n_estimators=10000,min_child_weight= 40,learning_rate=0.01,colsample_bytree = 1,sub
sample = 0.9)
reg.fit(X_train, y_train,eval_set = [(X_validate[:300],y_validate[:300])],early_stopping_rounds = 50) # Ch
ange verbose to True if you want to see it train
```

```
: XGBRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.01, max_delta_step=0, max_depth=6,
              min_child_weight=40, missing=nan, monotone_constraints=None,
              n_estimators=10000, n_jobs=0, num_parallel_tree=1,
              objective='reg:squarederror', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=0.9, tree_method=None,
              validate_parameters=False, verbosity=None)
```

The XGBoost Regressor gave a good RMSE of 0.0378 By tuning the hyper-parameters mentioned above.

Refinement:

- Then I trained a Catboost Regressor with iterations=10000,learning_rate=0.005, loss_function = 'RMSE' , which gave bestTest = 0.04186688717 bestIteration = 1676

```
model = CatBoostRegressor(iterations=10000,
                          learning_rate=0.005,
                          loss_function = 'RMSE')

model.fit(X_train, y_train,
          eval_set = [(X_validate[:300],y_validate[:300])],
         early_stopping_rounds = 50,
        verbose=True)
```

```
Stopped by overfitting detector  (50 iterations wait)

bestTest = 0.04186688717
bestIteration = 1676

Shrink model to first 1677 iterations.
<catboost.core.CatBoostRegressor at 0x7fbb9e3d8c18>
```

- Then finally trained a LGBM Regressor with num_leaves=31,learning_rate=0.001,max_bin= 30, n_estimators=10000 , which gave valid_0's l2: 0.0014098.

```
gbm = lgb.LGBMRegressor(num_leaves=31,
                        learning_rate=0.001,
                        max_bin = 30,
                        n_estimators=10000)

gbm.fit(X_train, y_train,
        eval_set = [(X_validate[:300],y_validate[:300].ravel())],
        early_stopping_rounds = 50,
        verbose = True
       )
```

```
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.001, max_bin=30,
              max_depth=-1, min_child_samples=20, min_child_weight=0.001,
              min_split_gain=0.0, n_estimators=10000, n_jobs=-1, num_leaves=31,
              objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
              silent=True, subsample=1.0, subsample_for_bin=200000,
              subsample_freq=0)
```

Parameters of the CatBoost and LGBM regressors are fine tuned to achieve lowest error. Then by averaging out the predictions of above 3 regressors we get our final prediction.

In addition, I specified the early_stopping_rounds in each of the three regressors to prevent them from overfitting on the training dataset.

The CatBoost regressor again gave a good RMSE of 0.04186688717.

And the LGBM regressor gave a good L2 0.0014098.

After the final training is completed, the R2 of the validating data is 0.88.

```
print(r2_score(y_val_org[:],y_val_pred[:]))
print(r2_score(y_train_org,y_train_pred))
```

```
0.8835554405766171
0.997468127332727
```

And Mean Absolute Error is 8.39.

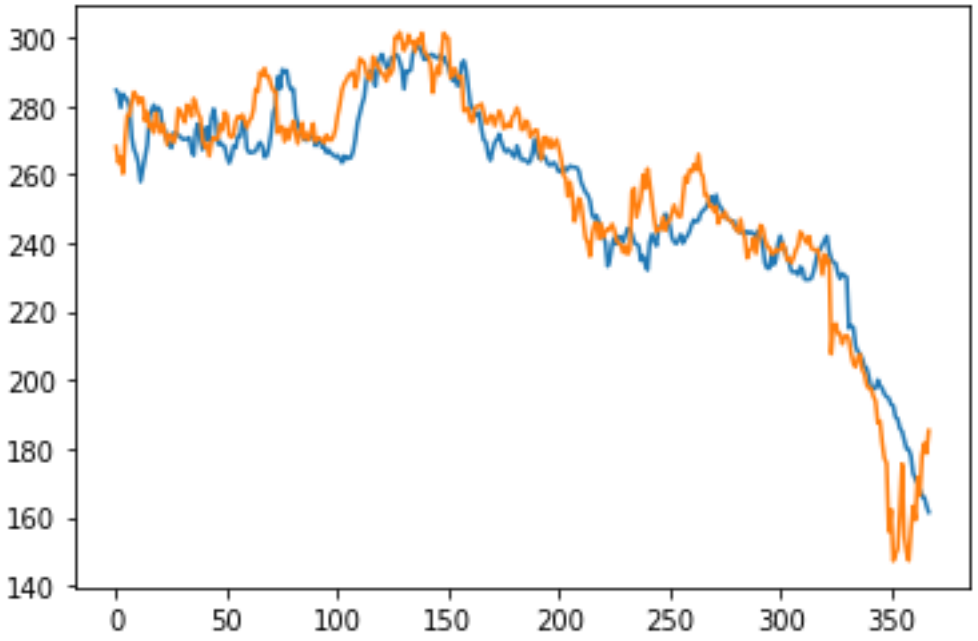
```
print(mean_absolute_error(y_val_org,y_val_pred))
print(mean_absolute_error(y_train_org,y_train_pred))
```

```
8.394047951209096
2.826698987988538
```

This mean absolute error is justified because the data used for validation is past one-year data in which the price of the stock is at the highest levels so a prediction which is around 8 counts away from the actual has a rough error percentage of 2.5%.

Model Evaluation and Validation:

In order to verify the reliability of the model, I use the daily price and volume data of ITC stock until last year, and calculate the error between the predicted price and the real price. The figure below shows predicted price in blue and original price in orange



As we can see from the plot there are nearly 350+ working days and the blue line is very close to the orange line i.e. in 5% range of the actual price.

Justification:

In order for the machine to predict the future price of the stock, I used a regressor model, first using the XGB Regressor second using the CatBoost Regressor and finally using the LGBM Regressor. Then I adjust the hyperparameter to make the model achieve the best effect and record the hyperparameter. Finally, I create a web app that customers can get the close price of the day they expect by inputting the date from which they want to predict the price and selecting the stock using a radio button so that they can get predicted price, actual price and the percentage error.

Conclusion:

Reflection:

During the training, I tried to use different hyperparameter combinations to build the model, and used the grid search method to find the optimal hyperparameters. Finally, I used the optimal hyperparameter to create model for stock. The 7-day close price R2 Score of the model was .88 , which was better than my expected result. Finally, I encapsulate the process of data acquisition, data processing, modeling and predicting into a scripts so that users can easily obtain the predicted price by simply running the script Probably because of hardware problems, this script will run on my machine for a few minutes, which may make the user experience very poor.

Improvement:

This project uses the XGB, CatBoost and LGBM for regression problems. The final model predicted stock value 7 days out is within +/- 5% of actual value, on average.

In the subsequent experiments, I can try LSTM models with more hidden layers and neurons. The more hidden layers and neurons the model has, the more powerful the model can fit data, and the model can process more complex data. However, with the more hidden layers and neurons, the model is prone to overfitting. Therefore, the selection of the appropriate number of neurons and the appropriate number of layers have become the key to improving the generalization ability of the model. In the subsequent experiments, I will try to use more layers of models to improve the approximation of the prediction of the model. Also, a better user Interface for the webapp.

Screenshots: