Question 1: Predict Annual Returns

Tanmoy Baidya, B.Tech Part-IV , IIT (BHU) Varanasi

Methodology:

1.Missing Values Imputing

1.1. Libor_Rate and Indicator Code is imputed with their mean values on training data.

1.2. Indicator Code is imputed with -999 value.

1.3 Hedge_Value and Status is imputed with boolean False.

2.Label Encoding for Categorical Values

Used mannual Encodig for categorical variables (pf_category, country_code, hedge_value, status, indicator_code).

3.Taking Care of Date Time Objects

Columns like start_date,creation_date and sell_date are converted to numpy datetime objects to generate new features.

4. Feature Engineering

4.1. Currency Conversion: All the currency are converted into same scale(in US Dollar). Here, created a function called Currency Converter which converted all the different currency into USD (Logic Used : GBP=1.3*USD , EUR=1.2*USD, CHF=USD, JPY=0.008*USD). Then this function is applied to the columns: sold, bought, euribor_rate,libor_rate.

4.2 For Currency column, only rows having USD value is considered as 1 and rest of them assigned as 0.

4.3 Outliers present in sold and bought columns are clipped by 10% usuing winsorize method from scipy library

4.4 3 New features are generated from our datetime objects.

4.4.1 Start Day: This is basically difference between Creation Date and Start date. Here I thought this difference may have effect over annual return.

4.4.2 Sell Day : Same way I have calculated the difference between Sell Date and Start Date.

4.4.3 Creation Day: Like wise it is difference between Sell Date and Creation Date.

Formula Used:

Diff_Day=(Difference in years)*365 + (Difference in months)*30+(Difference in Days)

5. Modelling:

I have used Gradient Boosted Tree based models , Gradient Boosting Regressor and XGBoost Regressor. Further tuning to those algorithms and taking ensemble with 60% weight on XGBoost output and 40% weight on Gradient Boosted Model gave me decent result with 0.97106 R-Squared value on test data.

6. Model Hyperparameters :

model_GBM= Max Depth: 10,Min Sample Leaf: 50, Min Samples Split : 10, N Estimators: 100
model_XGB= Max Depth:10 , N Estimators : 100

Tools Used:

Python 3.6 (Anaconda Distribution), Jupyter Notebook 5.2 , Pandas 0.21 , Numpy 1.13, scikit -learn 0.19, seaborn 0.8, matplotlib 2.1, scipy 1.0, xgboost 0.6a