Report: Coding Assignment 1

Student: Kohinur Akter (20201166)

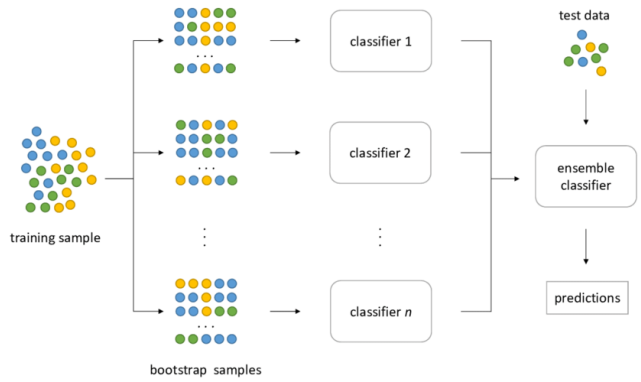
Implementation of deep neural network baseline model

Model selection:

I have used gradient boosting which refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.

Extreme Gradient Boosting (XGBoost) is an open-source library to provide implementation for the gradient boosting algorithm. I have used XGBRegressor for estimating the most important hyperparameters.

Ensemble architecture:



Credit: Bagging in Financial Machine Learning: Sequential Bootstrapping, Author: Proskurin Oleksandr

Ensemble model decrease the overall variance by averaging the performance of multiple estimates. Training different learners using different subsets of data with replacement is the idea of bootstrapping. Bragging is the concept of voting and here I have used averaging for regression.

Credit: Intuition behind the boosting algorithm, Author: Sirakorn Lamyai

In boosting, learners have poor predictive power. So, higher weights are given to subset of data which the learners misclassified earlier. Predictions are than combined with voting (classification), and I have used weighted sum (for regression)

Gradient boosting uses differentiable function losses from weak learners to generalize. At each boosting stage, the learners are used to minimize the loss function given current model.

In extreme gradient boosting process, the boosting is parallelized so that it improves the training time. This way many models can be trained on various subsets of training data then voting is performed for best model.

At first, I have used the default parameters:

max_depth = 6,
number of estimators = 100,
n_jobs = 16,
random state =16
learning rate = 0.3

```
In [78]:
1 model = xgb.XGBRegressor()
2 model.fit(xgb_train, y_train)
```

```
Out[78]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
                      gamma=0, gpu_id=-1, importance_type=None,
                      interaction_constraints='', learning_rate=0.300000012,
                      max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
                      monotone_constraints='()', n_estimators=100, n_jobs=16,
                      num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
                      reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
                      validate_parameters=1, verbosity=None)
```

For implementing ensemble model using training multiple models and averaging the predictions, I have trained following models:

```
models = [KernelRidge(), ElasticNet(), Lasso(), GradientBoostingRegressor(),
          BayesianRidge(), LassoLarsIC(), RandomForestRegressor(), xgb.XGBRegressor()]
```

The training data is splits into 5 subset and 20% of the data were used for testing during training.

```
# RANDOMISING THE CROSS VALIDATION SAMPLES
shuff = ShuffleSplit(n_splits = 5, test_size =.2, random_state=42)
```

The scores of the models on the validation sets are follows.

	Name	Train Accuracy Mean	Test Accuracy
0	KernelRidge	31.612	32.246
1	ElasticNet	37.497	37.560
2	Lasso	37.497	37.561
3	GradientBoostingRegressor	21.594	21.770
4	BayesianRidge	26.901	27.224
5	LassoLarsIC	26.901	27.223
6	RandomForestRegressor	23.296	23.131
7	XGBRegressor	22.516	21.869

Since the Lasso and ElasticNet accuracy were better I have tested the models on validation set. The following are the loss and model prediction scores

model	Score	loss
Lasso	0.4968	0.30706
ElasticNet	0.4969	0.30705
RandomForestRegressor	0.8169	0.17061
XGBRegressor	0.83509	0.16165

None of the predicted values matched exactly to the real price-

```
correct pred: 0
incorrect pred: 4323
correct prediction:
 prediction--true price
 Empty DataFrame
Columns: []
Index: []
incorrect prediction:
 prediction--true price
              0         1
0     332641.0  453500.0
1     567342.0  510000.0
2     366897.0  379900.0
3     335763.0  340000.0
4     277850.0  273500.0
...        ...       ...
4318  260577.0  295000.0
4319  611012.0  625000.0
4320  569412.0  412500.0
4321  486854.0  595000.0
4322  530323.0  395000.0

[4323 rows x 2 columns]
```

Improvement procedure:

1. Used all mostly related 10 features of the dataset rather than using the optimized 5 features.
2. I have used pipeline to train multiple models and used intuitive parameters for gradient boosting models, then used the average score of the models.

```
model_Lasso= make_pipeline(RobustScaler(), Lasso(alpha =0.000327, random_state=18))

model_ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.00052, l1_ratio=0.70654, random_state=18))


model_GBoost = GradientBoostingRegressor(n_estimators=3000, learning_rate=0.05,
                                   max_depth=4, max_features='sqrt',
                                   min_samples_leaf=15, min_samples_split=10,
                                   loss='huber', random_state =18)

model_XGB=xgb.XGBRegressor(n_jobs=-1, n_estimators=849, learning_rate=0.015876,
                           max_depth=58, colsample_bytree=0.599653, colsample_bylevel=0.287441, subsample=0.154134, seed=18)

model_lgb = lgb.LGBMRegressor(objective='regression',num_leaves=5,
                              learning_rate=0.05, n_estimators=720,
                              max_bin = 55, bagging_fraction = 0.8,
                              bagging_freq = 5, feature_fraction = 0.2319,
                              feature_fraction_seed=9, bagging_seed=9,
                              min_data_in_leaf =6, min_sum_hessian_in_leaf = 11)

forest_reg = RandomForestRegressor(bootstrap=False, criterion='mse', max_depth=None,
                                   max_features=60, max_leaf_nodes=None, min_impurity_decrease=0.0,
                                   min_samples_leaf=1,
                                   min_samples_split=2, min_weight_fraction_leaf=0.0,
                                   n_estimators=70, n_jobs=1, oob_score=False, random_state=42,
                                   verbose=0, warm_start=False)
```

Models	scores
Lasso	0.6937
E_Net	0.6937
XGBRegressor	0.8371
GBRegressor	0.8371
LgbRegressor	0.821507

The scores are still same, only a little improvement

The average least square loss of all the model is = 0.1728 which is not an improvement.

3. (n_estimators=3000, learning_rate=0.05, max_depth=4, max_features='sqrt', min_samples_split=5, loss='ls', random_state =18)
   Using these parameters for gradient boosting regression I have achieved score or 0.83427
4. I have used all the features including date also since there is correlation with the price. Using the previous parameters of the gradient boosting regressor score is 0.91593 which is an improvement.

I have tried removal of outliers, but the score did not improve, score is 0. 911805.
Further I have tried tuning the parameters the scores are mostly same.

5. I have used Keras Regression model with dense layers and relu activation function. After training 400 epochs the result has not improved.

```
fig = plt.figure(figsize=(10,5))
plt.scatter(val_y, y_pred)
# Perfect predictions
plt.plot(val_y, val_y,'r')
```

```
<matplotlib.lines.Line2D at 0x1a22ad68370>]
```

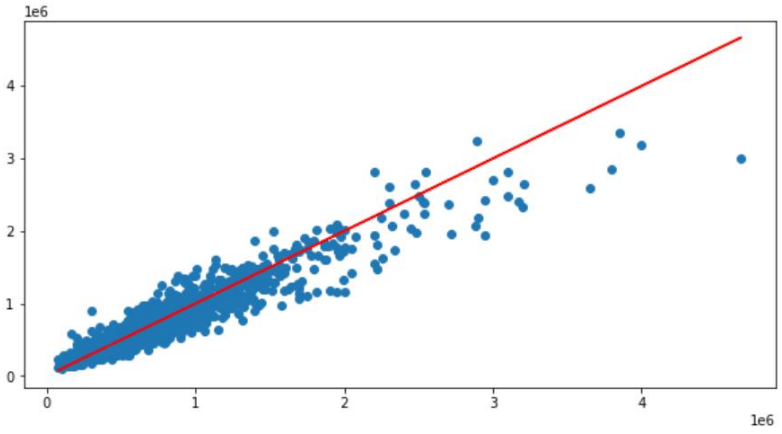


Fig: prediction on validation dataset and real price.