機器學習

期末專題報告

Pump It Up

組名：NTU_R05921087_羅機學

組員：　鍾智堅　R05921087

吳錦賢　R05921089

林威辰　R05921068

林展嘉　R05921060

A. Preprocessing Data

這個比賽，總共有三個檔案要處理：train.csv、test.csv 和 macro.csv。利用 macro humility factor 和 micro humility factor 以作為調整最終預測值的縮放和趨近於平均值，以降低誤差。在這個部分，可以分成四個部分去分析。

1. macro_mean

首先，我們把 macro.csv 和 train.csv 的 timestamp 這個 feature 分成 3 個 feature——year、month 以及 yearmonth。

接下來，再用分佈滯後來量化過去到最新資料，再到將來的演變過程。

```
def almonZmatrix(X, maxlag, maxdeg):
    """
    Creates the Z matrix corresponding to vector X.
    """
    n = len(X)
    Z = ml.zeros((len(X)-maxlag, maxdeg+1))
    for t in range(maxlag,  n):
       #Solve for Z[t][0].
       Z[t-maxlag,0] = sum([X[t-lag] for lag in range(maxlag+1)])
       for j in range(1, maxdeg+1):
             s = 0.0
             for i in range(1, maxlag+1):
                s += (i)**j * X[t-i]
             Z[t-maxlag,j] = s
    return Z
```

使用 statsmodels.api 中的 OLS 模型，即是屬於 linear regression model，以預測 macro.csv 的 price_doc 資料。

```
# Prepare data for macro model
y = df.price_doc.div(df.cpi).apply(np.log).loc[201108:201506]
lncpi = df.cpi.apply(np.log)
tblags = 5    # Number of lags used on PDL for Trade Balance
mrlags = 5    # Number of lags used on PDL for Mortgage Rate
cplags = 5    # Number of lags used on PDL for CPI
ztb = almonZmatrix(df.balance_trade.loc[201103:201506].as_matrix(), tblags, 1)
zmr = almonZmatrix(df.mortgage_rate.loc[201103:201506].as_matrix(), mrlags, 1)
zcp = almonZmatrix(lncpi.loc[201103:201506].as_matrix(), cplags, 1)
columns = ['tb0', 'tb1', 'mr0', 'mr1', 'cp0', 'cp1']
z = pd.DataFrame( np.concatenate( (ztb, zmr, zcp), axis=1), y.index.values, columns )
X = sm.add_constant( z )

# Fit macro model
eq = sm.OLS(y, X)
fit = eq.fit()

# Predict with macro model
test_cpi = df.cpi.loc[201507:201605]
test_index = test_cpi.index
ztb_test = almonZmatrix(df.balance_trade.loc[201502:201605].as_matrix(), tblags, 1)
zmr_test = almonZmatrix(df.mortgage_rate.loc[201502:201605].as_matrix(), mrlags, 1)
zcp_test = almonZmatrix(lncpi.loc[201502:201605].as_matrix(), cplags, 1)
z_test = pd.DataFrame( np.concatenate( (ztb_test, zmr_test, zcp_test), axis=1),
                       test_index, columns )
X_test = sm.add_constant( z_test )
pred_lnrp = fit.predict( X_test )
pred_p = np.exp(pred_lnrp) * test_cpi
```

預測后，我們再算出 macro_mean，即是 test.csv 的 monthprice 的平均值。

```
# Merge with test cases and compute mean for macro prediction
test["timestamp"] = pd.to_datetime(test["timestamp"])
test["year"]  = test["timestamp"].dt.year
test["month"] = test["timestamp"].dt.month
test["yearmonth"] = 100*test.year + test.month
test_ids = test[["yearmonth","id"]]
monthprices = pd.DataFrame({"yearmonth":pred_p.index.values,"monthprice":pred_p.values})
macro_mean = np.exp(test_ids.merge(monthprices, on="yearmonth").monthprice.apply(np.log).mean())
macro_mean
```

2. Jason’s model preprocessing data

首先聲明，對於 train.csv，我們用了三個不同的預處理數據的方法去跑同一個模型。

第一個預處理數據是屬於 jason’s model 的，如對 macro.csv 處理一樣，把 timestamp 這個 feature 分成 month_year_cnt、week_year_cnt、month 以及 dow，多出了 4 個新的 feature。

然後我們個別把每個 feature 裡面不正常的值判斷為 np.NaN，畢竟 xgboost 對於 missing data 處理非常棒，因此即使不做補償，對於訓練過程不會有太大的影響。不過重點在於當針對 full_sq 這個 feature，把 0 值都設為 50，然後降低每平方米的價格。

同樣，我們產生了新的 feature——rel_floor、rel_kitch_sq、room_size。

```
train['rel_floor'] = train['floor'] / train['max_floor'].astype(float)
train['rel_kitch_sq'] = train['kitch_sq'] / train['full_sq'].astype(float)

test['rel_floor'] = test['floor'] / test['max_floor'].astype(float)
test['rel_kitch_sq'] = test['kitch_sq'] / test['full_sq'].astype(float)

train.apartment_name=train.sub_area + train['metro_km_avto'].astype(str)
test.apartment_name=test.sub_area + train['metro_km_avto'].astype(str)

train['room_size'] = train['life_sq'] / train['num_room'].astype(float)
test['room_size'] = test['life_sq'] / test['num_room'].astype(float)
```

然後刪除了 id、timestamp 以及 price_doc。凡是屬於非數字的資料，我們做 one Hot Encoding 處理。

3. Reynaldo’s model preprocessing data

這個 model 對於資料處理比較少。對於 macro.csv 是完全不處理，而 train.csv 和 test.csv，則去掉 id、timestamp 以及 price_doc 這三個 feature。

剩下的其他非數字類的 feature，則採取 one Hot Encoding 方式。

```
for c in x_train.columns:
    if x_train[c].dtype == 'object':
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(x_train[c].values))
        x_train[c] = lbl.transform(list(x_train[c].values))

for c in x_test.columns:
    if x_test[c].dtype == 'object':
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(x_test[c].values))
        x_test[c] = lbl.transform(list(x_test[c].values))
```

4. Bruno’s model preprocessing data

首先，這個預處理是先把不合理的 feature 的值 drop 掉，然後把 test.csv 並列到 train.csv 對齊，然後把 marco 的資料併排到與 train.csv 和 test.csv 相同的 timestamp 旁，自行增加一個 feature，再把「月_年」及「週_年」填入。

接著，把 feature 中為數值的部分抽出來，也把 feature 中非數值的部分抽出來，把他們數值化成出現次序的編號，

```
X_all = np.c_[
    df_all.select_dtypes(exclude=['object']).values,
    np.array(list(map(factorize, df_obj.iteritems()))).T
]
```

```
df_obj = df_all.select_dtypes(include=['object']).copy()

for c in df_obj:
    df_obj[c] = pd.factorize(df_obj[c])[0]
```

最後再將兩組數據合併，把 train 跟 test 切開成兩個部分。

```
df_values = pd.concat([df_numeric, df_obj], axis=1)


# Convert to numpy values
X_all = df_values.values
print(X_all.shape)

X_train = X_all[:num_train]
X_test = X_all[num_train:]
```

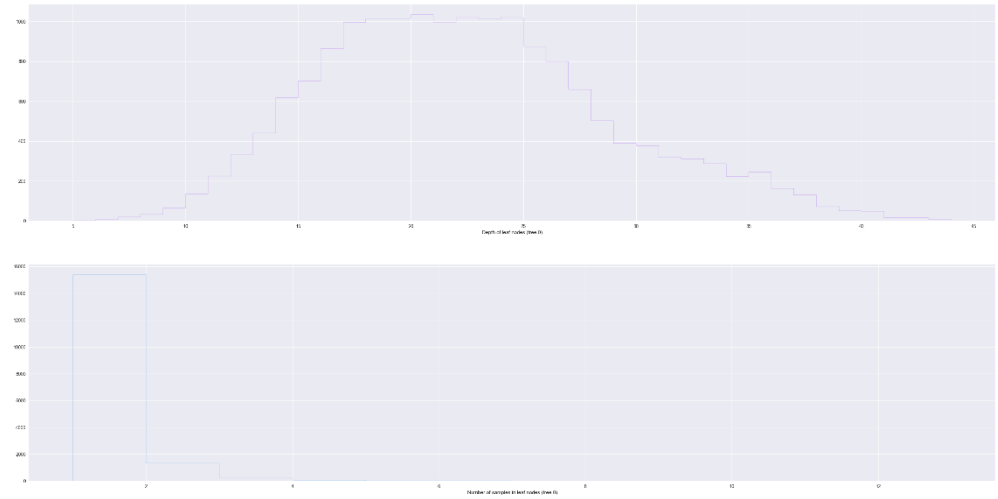
B. Model Description

1. Random Forest Regression

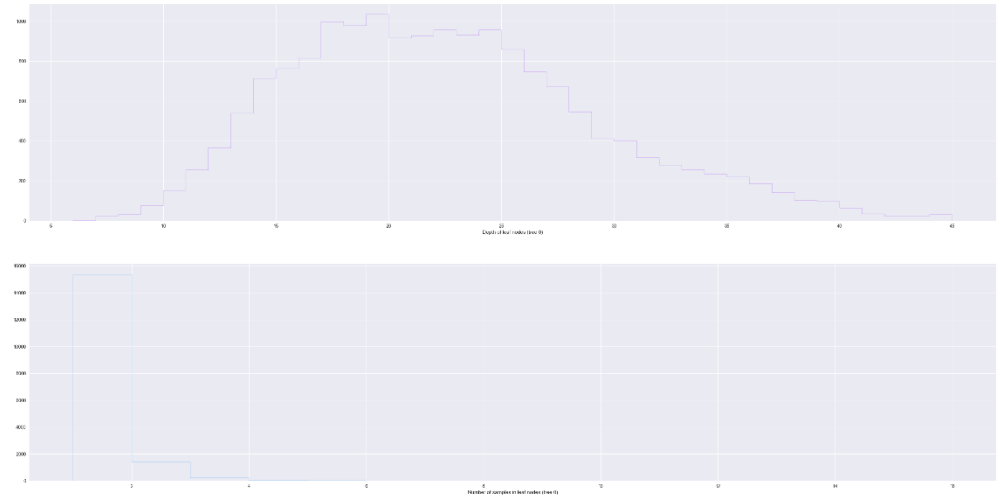
當初會選這個 model 是因為考慮到有非常多的 missing data，因此一般簡單的演算法如 linear regression、SVR 訓練成果不理想。考慮到 random forest 在許多比賽中表現非常亮眼，因此嘗試試用這個 model。

以下**是個別為** Jason’s model **和** Reynaldo’s model **的模型描述**：

Jason’s model



Reynaldo’s model



2. xgboost - eXtreme Gradient Boosting

經過第一個 model 的測試后，因為可以調動的參數非常有限，因此嘗試試用 xgboost，找出最佳化的參數以得到更好的結果。

Xgboost 是屬於 random forest 的延伸應用，即雖然演算法是相似，但是 xgboost 可以調動的參數更多，即非常靈活。Xgboost 最大的特點是自動利用 CPU 的多線程進行並行，并在演算法上加以改進提高了精準度，而且對於缺失的資料處理非常好，即使有空白或者 nan，也可以選擇性忽視而繼續訓練模型。

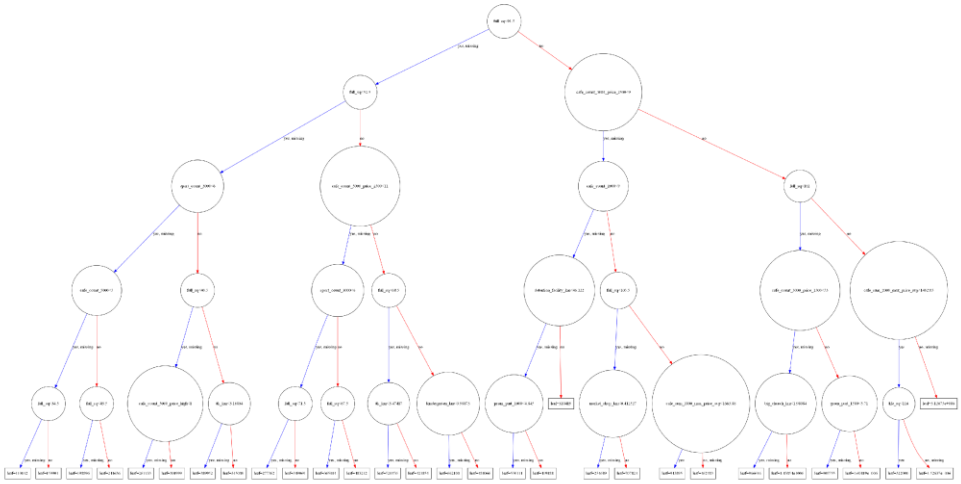
設定的最佳參數為：

```
xgb_params = {
    'eta': 0.05,
    'max_depth': 5,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'objective': 'reg:linear',
    'eval_metric': 'rmse',
    'silent': 1
}
```

我們設立的 xgboost 模型以 linear regression 為基礎去訓練，并觀察 root mean squared error 以減低誤差。不過，為了加速訓練過程，利用 xgboost 的 DMatrix，把資料轉換成適合 xgboost 的資料形態（LibSVM 格式），主要是提高內存的使用率和訓練速度。

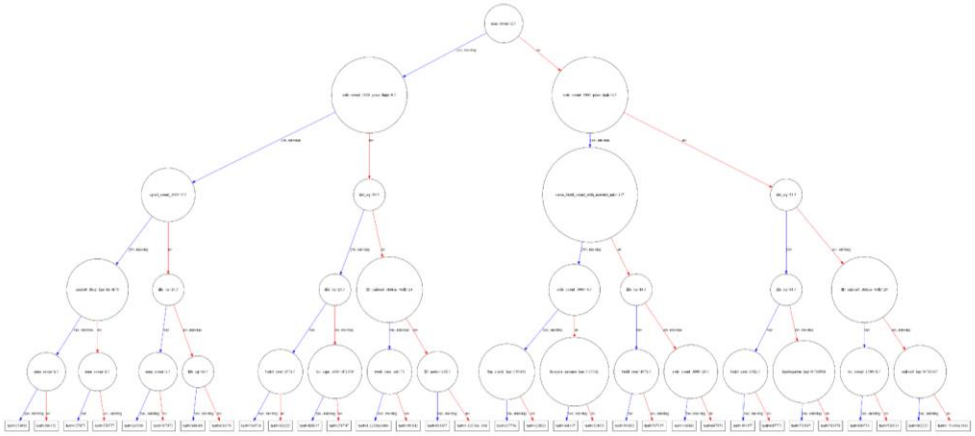
以下是各個模型的示意圖：

Jason’s model

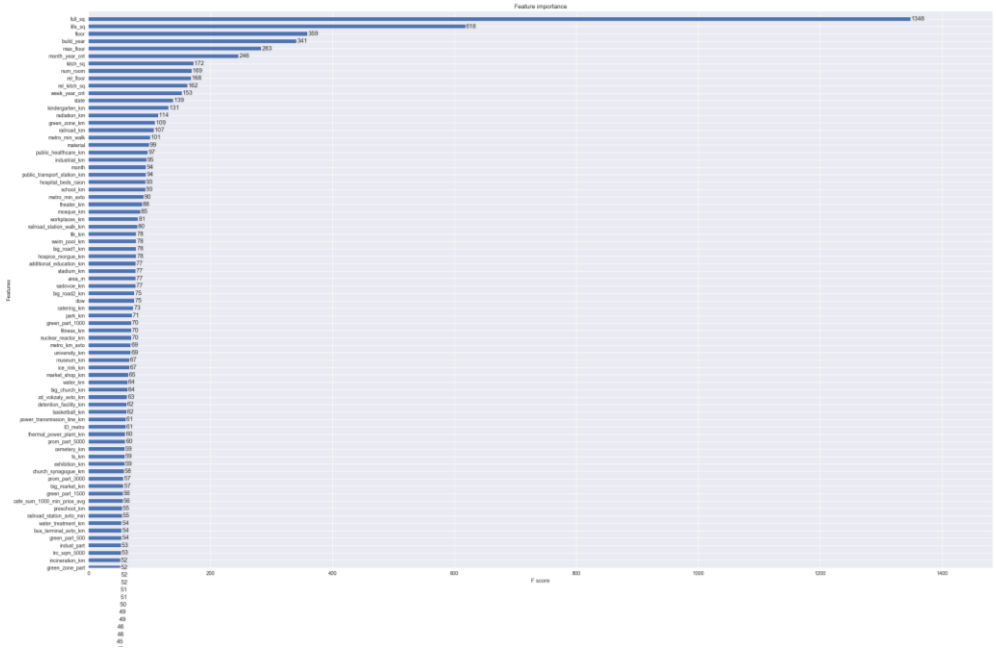


Reynaldo’s model

Bruno’s model



然後，我們也分析了在這些模型中的 feature importance：



從圖中，可以看出前 5 個最重要的 feature 依序為——full_sq、life_sq、floor、build_year、max_floor。因此，在 preprocesssing data 部分，我們會針對 full_sq 的 data cleaning。

C. Experiments and Discussion

其實通過用相同的 data，即前兩張的處理 data，然後讓 random forest regression 和 xgboost 個別去訓練并各自用 majority voting 方法測試，得到意料中的結果：xgboost 的 kaggle 分數遠遠比 random forest regression 好很多！

Random forest regression 結果：

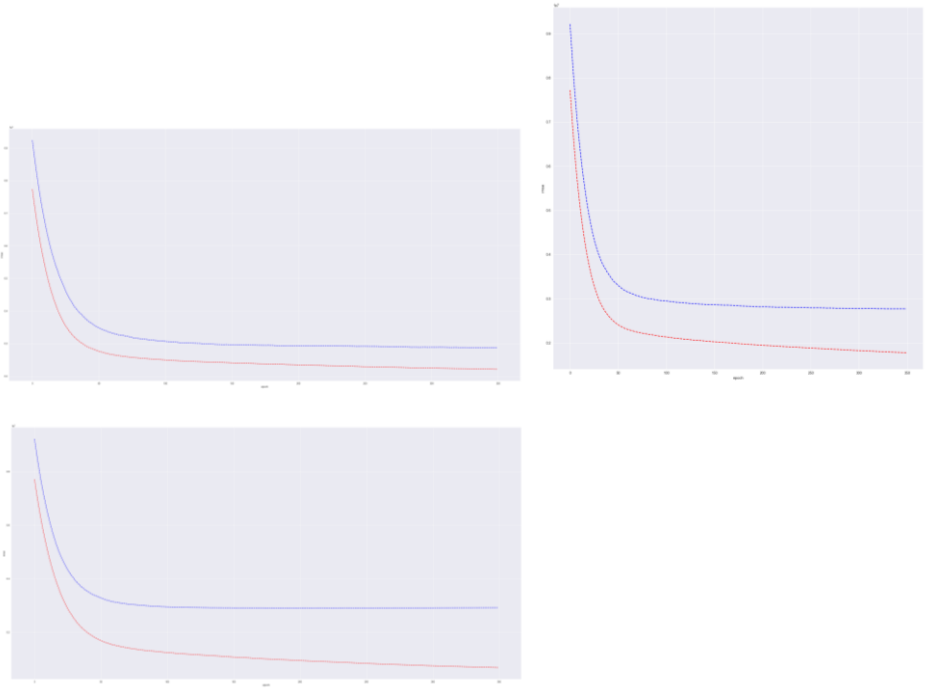
sub_R.csv 17 hours ago by r05921089_Steven add submission details	0.33174	0.33064

Xgboost 結果：

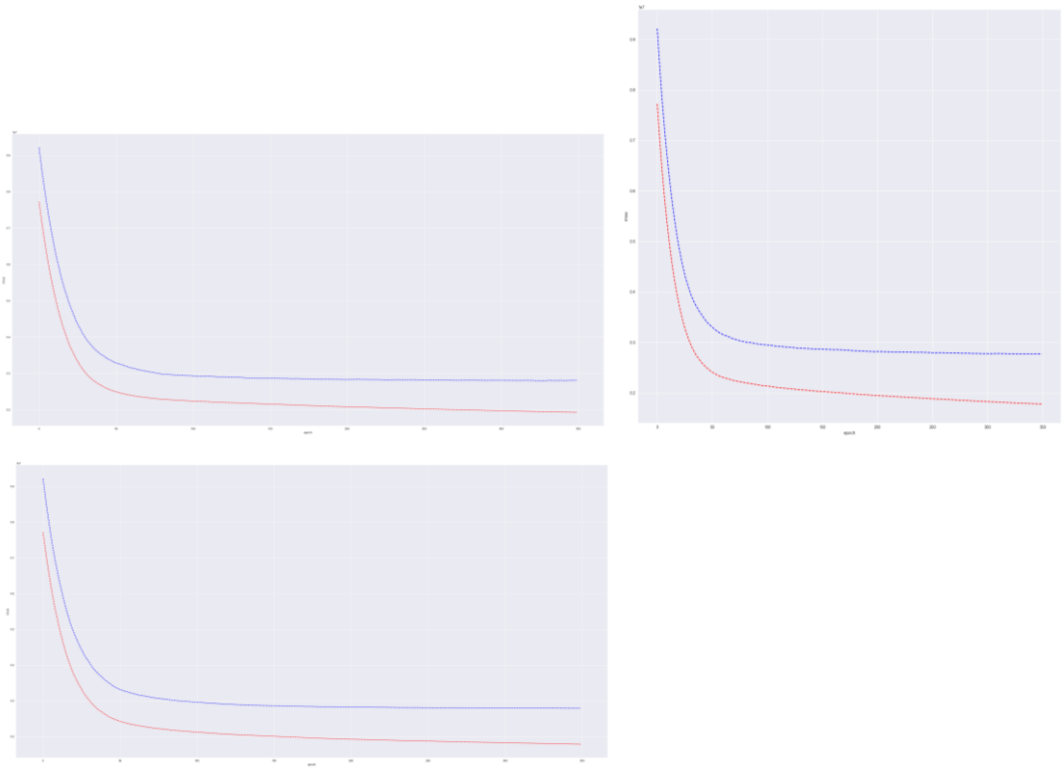
sub_2model.csv 19 hours ago by r05921089_Steven	0.31591	0.31090

因此，接下來我們只針對 xgboost 的參數做實驗，找出最佳的參數。

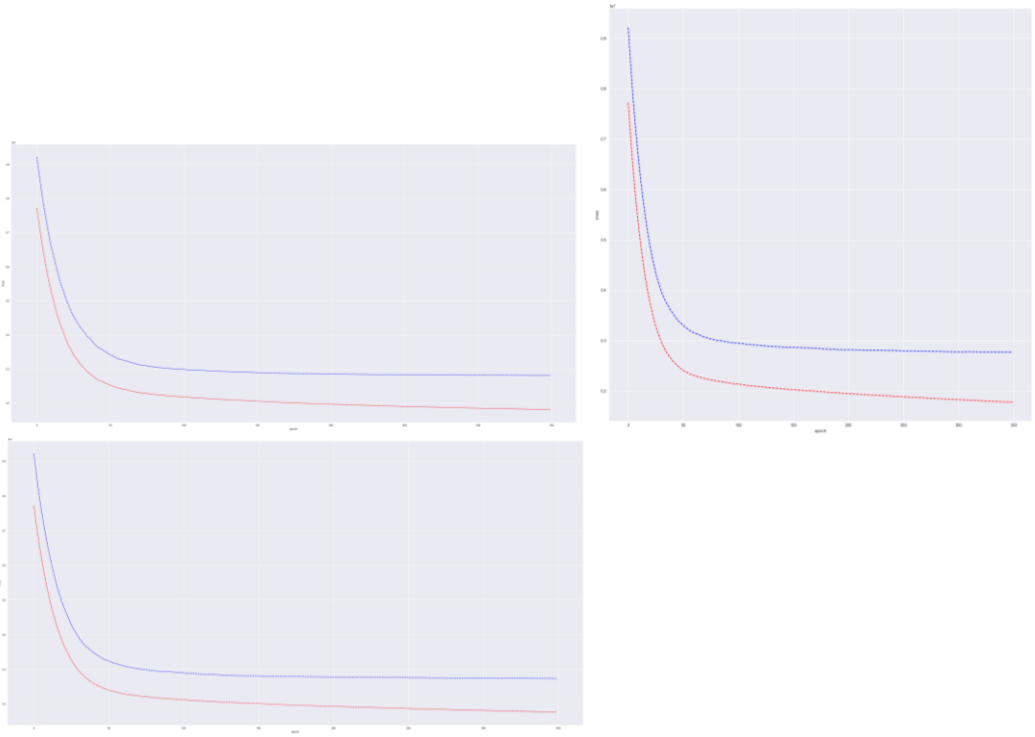
首先，我們調整 max_depth 的部分，下圖從左到右，分別為 3，5,10,從 validation error 比對，可以發現 max depth=5 為最佳。



針對 subsample 部分，我們使用了三個參數——0.3、0.7、1，對比 validation error，發現 0.7 為最佳值。



然後，有關 colsample_bytree 部分，我們各位測試了 0.3、0.7、1，同樣的對比 validation error，發現 0.7 是最好的指。



因此，總結的最佳參數為：

```
xgb_params = {
    'eta': 0.05,
    'max_depth': 5,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'objective': 'reg:linear',
    'eval_metric': 'rmse',
    'silent': 1
}
```

D. Work Division

姓名	負責項目
鍾智堅	負責編輯主程式、打報告
吳錦賢	負責編輯主程式、打報告
林威辰	打報告
林展嘉	打報告