Credit Suisse Error Check Automation Project

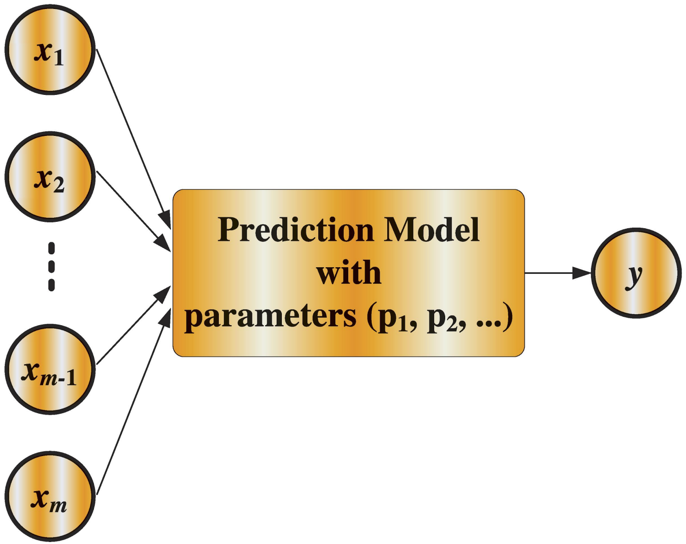
11.22.2019

Yuru Cao
Mentors: Felipe Grilli, Sree Kotni

Problem Definition and Goal

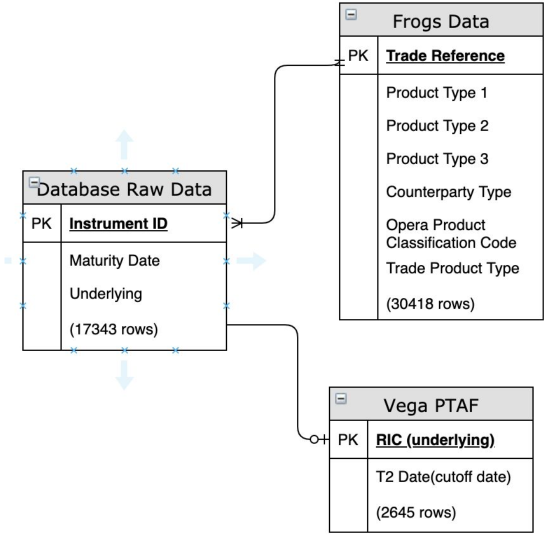
Perform an assessment of appropriateness of Fair Value Leveling and Trade Product Type (error check automation)

1. Determined by cut-off date and maturity date
2. Using Product Type 1/2/3 and Counterparty Type to detect potential fair value error



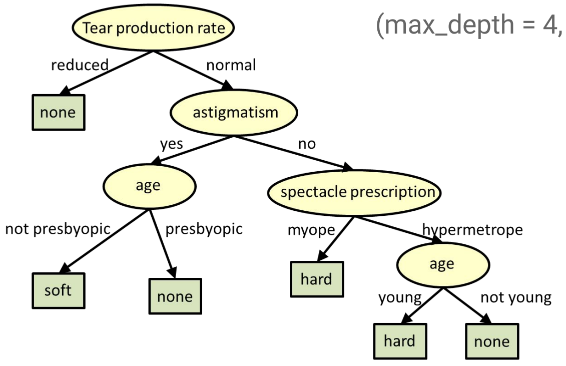
Methodology

1. Remove duplicates, remove NAs, converting Datetime format
2. Join Raw Data and Vega, compare Maturity date with Cut-off date, determine primary Fair Value level
3. Join merged table and Frogs Data, one-hot encoding predictor variables, build predicting model for remaining Fair Value Level



Models

Scikit-learn Decision Tree model



(max_depth = 4, min_samples_split=10)

- ➔ Train the model using all dataset, predicting back the tag

(Overfitting, less errors)

- ➔ 10-fold training and testing data, train the model using 9 folder data, predict the remaining 1 folder, combing 10 folder result

(training data not enough, more errors detected)

Future Improvements

1. The classification model can not be evaluated. For now we assume the fair value is mostly correct, and ask the model to learn from the data. It still requires manual check to determine whether the errors are true

2. Classification model need more consideration to better fit our dataset. Ideally, each of our predictor variable are the node, its categories are the branches. The layer and node are pre-defined for the model.

Code Preview

```python
# --- Data Cleaning ---
# Converting excel integer date into Python format datetime (int->tuple->datetime),
# Only using three colunms,
# Remove nan maturity date
raw_cleaned = raw[raw['Maturity Date'].notnull()][['Maturity Date','Instrument Id','Und
raw_cleaned['Maturity Date'] = raw_cleaned['Maturity Date'].apply(lambda x: x if isinst
raw_cleaned['Maturity Date'] = raw_cleaned['Maturity Date'].apply(lambda x:x if isinsta

# Vega Cutoffdate already in datetime format
# Remove duplicated underlying index
vega.sort_values('T2 Date (cutoff date)', inplace = True, ascending = False)
vega.drop_duplicates(subset = 'RIC (underlying)',keep ='first',inplace = True)

# Merge raw with vega(underlying) left join keep records on raw data
cutoff_merge = pd.merge(raw_cleaned, vega, left_on = 'Underlying',right_on = 'RIC (unde
# Remove rows where undelying not found in vega -> compare maturity date to determine f
cutoff_merge = cutoff_merge[cutoff_merge['RIC (underlying)'].notnull()]
# Primary Fair value by comparing maturity and cut-off date
cutoff_merge['primaryflv'] = cutoff_merge.apply(lambda x: 2 if x['Maturity Date'] < x['

# Concerns: Same instrument with different underlying -> different fair value reference
# cutoff_merge['Instrument Id'].duplicated().unique()
cutoff_merge.sort_values('primaryflv',inplace = True)
cutoff_merge.drop_duplicates(subset = 'Instrument Id',keep = 'first', inplace = True)

# Combine the raw into frogs with fvl
sepmerge = pd.merge(frogsep,cutoff_merge,left_on='Trade Reference',right_on='Instrument Id',how = 'left')
sepmerge = sepmerge.drop(columns=['Business Date','Structure','Structure Product Type(Final)','Contract Code / PT3','Product Id','Amount Type'
# dummy encoding pt1-3 features, spread into 77 columns
pt1 = pd.get_dummies(sepmerge['Product Type 1'])
pt1 = pt1.rename(columns={'Funding':'Funding1','Nostro':'Nostro1','OTC Derivatives':'OTC Derivatives1'})
pt2 = pd.get_dummies(sepmerge['Adjustment Type / PT2']) #Convert categorical var into label
pt2 = pt2.rename(columns={'Funding':'Funding2','Nostro':'Nostro2','Equities':'Equities2'})
pt3 = pd.get_dummies(sepmerge['Opera Product Type3'])
pt3 = pt3.rename(columns={'Funding':'Funding3','Nostro':'Nostro3','Equities':'Equities3','OTC Derivatives':'OTC Derivatives3'})
sepmerge = sepmerge.drop(columns=['Product Type 1','Adjustment Type / PT2','Opera Product Type3'])
sepmerge = sepmerge.join(pt1).join(pt2).join(pt3)
```

```python
# -- Model part and FVL deciding ---
primary = sepmerge[sepmerge['primaryflv'].notnull()]
primary[primary['primaryflv']!=primary['Final Fair Value Category']].shape

# Use dataset left for training classfication model
# 1.Use the whole dataset as training
left = sepmerge[sepmerge['primaryflv'].isnull()]
tree = DecisionTreeClassifier(max_depth = 4,min_samples_split=10)
X = left.iloc[:,-61:-1]
tree.fit(X, left['Final Fair Value Category'])
left['predict'] = tree.predict(X)
left[left['predict']!=left['Final Fair Value Category']].shape

# 2.K-fold training and testing
left = shuffle(left)
kf = KFold(n_splits = 10)
left['predict2'] = -1
for train,test in kf.split(left):
    tree.fit(X.iloc[train],left['Final Fair Value Category'].iloc[train])
    left['predict2'].iloc[test] = tree.predict(X.iloc[test])
left[left['predict2']!=left['Final Fair Value Category']].shape

# --- Model fitting for trade product type
type = sepmerge[['Trade Product Type(Final)','Opera Product Classification Code']]
code = pd.get_dummies(type['Opera Product Classification Code'])
type = type.drop(columns=['Opera Product Classification Code'])
type = type.join(code)
type.dropna(inplace = True)


X = type.iloc[:,1:]
reg = LogisticRegression().fit(X,type['Trade Product Type(Final)'])
type['predict'] = reg.predict(X)
type[type['predict']!=type['Trade Product Type(Final)']].shape
```