Name – Sarang .S.Gaikwad

DALL = ALL

```
TD                = {}  #Dictionary for transformed data

FA                = {}  #Dictionary for factors

N                 = np.int_(5)  #Used to control data cleaning, see notes

#Next 4 lines were added so forward returns will be used

Returns_AP           = D['Returns_AP'].copy()

TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()


TD['OperatingMargin_AP']        = -D['OperatingMargin_AP'].copy()      #Change the sign of price to book because high is bad, low is good

TD['OperatingMargin_AP']        = td.CleanData(TD['OperatingMargin_AP'],N) #Moderate outliers so model building is better

TransformList         = ['ZSBT','ZSCSBS']            #z-score by time, decile crosssection by sector

TD['OperatingMargin_AP']        = td.Transform(TD['OperatingMargin_AP'],TransformList,D)

X = TD['OperatingMargin_AP']


NumTile             = np.int_(5)

FA['OperatingMargin_AP']     = 
ft.AnalyzeFactor('OperatingMargin_AP',TD['OperatingMargin_AP'],TD['ForwardReturns_AP'],NumTile
)
```

Applying transformations ...

ZSBT

z-score by time

ZSCSBS

z-score by crosssection by sector

C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1) (1)\TransformData.py:70: RuntimeWarning: invalid value encountered in divide

DataZ   = (DataZ-np.nanmean(DataZ))/np.nanstd(DataZ)

Transformations applied

C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1) (1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice

R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA.keys()

Out[102]: dict_keys(['OperatingMargin_AP'])

FA['OperatingMargin_AP'].keys()

Out[103]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod', 'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p', 'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['OperatingMargin_AP']['ICitp']

Out[104]: 0.6317900576815089

TD = {} #Dictionary for transformed data

FA = {} #Dictionary for factors

N = np.int_(5) #Used to control data cleaning, see notes

#Next 4 lines were added so forward returns will be used

Returns_AP = D['Returns_AP'].copy()

TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Price_Book_AP'] = -D['Price_Book_AP'].copy() #Change the sign of price to book because high is bad, low is good

TD['Price_Book_AP'] = td.CleanData(TD['Price_Book_AP'],N) #Moderate outliers so model building is better

TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector

TD['Price_Book_AP'] = td.Transform(TD['Price_Book_AP'],TransformList,D)

X = TD['Price_Book_AP']

```
NumTile             = np.int_(5)

FA['Price_Book_AP']    =
ft.AnalyzeFactor('Price_Book_AP',TD['Price_Book_AP'],TD['ForwardReturns_AP'],NumTile)

Applying transformations ...

ZSBT

z-score by time

ZSCSBS

z-score by crosssection by sector

Transformations applied

C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1)
(1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice

  R['EW_BenchMark_p']            =   np.nanmean(Returns_AP,axis=0)#equaly weighted mean by
period


FA.keys()

Out[96]: dict_keys(['Price_Book_AP'])


FA['Price_Book_AP'].keys()

Out[97]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod', 'AnnPeriods',
'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p', 'SignalTile_AP',
'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC',
'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn', 'IRdecay_n',
'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t', 'RsquareBySignalTile_t',
'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt',
'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])


FA['Price_Book_AP']['ICitp']

Out[98]: 0.655915725445113


TD              = {}  #Dictionary for transformed data

FA              = {}  #Dictionary for factors

N               = np.int_(5)  #Used to control data cleaning, see notes
```

#Next 4 lines were added so forward returns will be used

Returns_AP = D['Returns_AP'].copy()

TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['ROE_AP'] = -D['ROE_AP'].copy() #Change the sign of price to book because high is bad, low is good

TD['ROE_AP'] = td.CleanData(TD['ROE_AP'],N) #Moderate outliers so model building is better

TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector

TD['ROE_AP'] = td.Transform(TD['ROE_AP'],TransformList,D)

X = TD['ROE_AP']

NumTile = np.int_(5)

FA['ROE_AP'] = ft.AnalyzeFactor('ROE_AP',TD['ROE_AP'],TD['ForwardReturns_AP'],NumTile)

Applying transformations ...

ZSBT

z-score by time

ZSCSBS

z-score by crosssection by sector

Transformations applied

C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1) (1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice

R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA.keys()

Out[83]: dict_keys(['ROE_AP'])

FA['ROE_AP'].keys()

Out[84]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod', 'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p', 'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['ROE_AP']['ICitp']

Out[85]: 0.3582827250172443

```
TD                  = {}  #Dictionary for transformed data
FA                  = {}  #Dictionary for factors
N                   = np.int_(5)  #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP          =  D['Returns_AP'].copy()
TD['ForwardReturns_AP'] =  D['ForwardReturns_AP'].copy()

TD['Rho_AP']     = -D['Rho_AP'].copy()       #Change the sign of price to book because high is bad, low is good
TD['Rho_AP']     = td.CleanData(TD['Rho_AP'],N) #Moderate outliers so model building is better
TransformList         = ['ZSBT','ZSCSBS']             #z-score by time, decile crosssection by sector
TD['Rho_AP']     = td.Transform(TD['Rho_AP'],TransformList,D)
X = TD['Rho_AP']

NumTile              = np.int_(5)
FA['Rho_AP']     = ft.AnalyzeFactor('Rho_AP',TD['Rho_AP'],TD['ForwardReturns_AP'],NumTile)
```

Applying transformations ...

ZSBT

z-score by time

ZSCSBS

z-score by crosssection by sector

Transformations applied

C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1) (1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice

R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA.keys()

Out[78]: dict_keys(['Rho_AP'])

FA['Rho_AP'].keys()

Out[79]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod', 'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p', 'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['Rho_AP']['ICitp']

Out[80]: 0.05613639653377726

Based on the provided Information Coefficient (IC) values, the "Price to Book" factor (Price_Book_AP) appears to be the best factor among the ones analyzed. It has an IC value of 0.655915725445113, which suggests a stronger relationship with future returns compared to the other factors.

Here are the IC values for each factor:

1. Price to Book (Price_Book_AP): IC = 0.655915725445113
2. Operating Margin (OperatingMargin_AP): IC = 0.6317900576815089
3. Return on Equity (ROE_AP): IC = 0.3582827250172443
4. Rho (Rho_AP): IC = 0.05613639653377726

Therefore, based on the IC values, the "Price to Book" factor seems to be the most effective predictor of future returns among the factors analyzed. It exhibits a relatively high IC value, indicating a stronger predictive power compared to the other factors.

DALL = Low

```
TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['OperatingMargin_AP'] = -D['OperatingMargin_AP'].copy() #Change the sign of price to
book because high is bad, low is good
TD['OperatingMargin_AP'] = td.CleanData(TD['OperatingMargin_AP'],N) #Moderate outliers
so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['OperatingMargin_AP'] = td.Transform(TD['OperatingMargin_AP'],TransformList,D)
X = TD['OperatingMargin_AP']

NumTile = np.int_(5)
FA['OperatingMargin_AP'] =
ft.AnalyzeFactor('OperatingMargin_AP',TD['OperatingMargin_AP'],TD['ForwardReturns_AP']
,NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied
C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1)
(1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice
R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA.keys()
Out[31]: dict_keys(['OperatingMargin_AP'])

FA['OperatingMargin_AP'].keys()
Out[32]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod',
'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p',
'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p',
'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn',
'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t',
'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t',
'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt',
'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])
```

FA['OperatingMargin_AP']['ICitp']
Out[33]: 0.5855530380173047

TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Earnings_AP'] = -D['Earnings_AP'].copy() #Change the sign of price to book because high is bad, low is good
TD['Earnings_AP'] = td.CleanData(TD['Earnings_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Earnings_AP'] = td.Transform(TD['Earnings_AP'],TransformList,D)
X = TD['Earnings_AP']

NumTile = np.int_(5)
FA['Earnings_AP'] =
ft.AnalyzeFactor('Earnings_AP',TD['Earnings_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied
C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2 project (1)
(1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice
R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA.keys()
Out[36]: dict_keys(['Earnings_AP'])

FA['Earnings_AP'].keys()
Out[37]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod', 'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p', 'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['Earnings_AP']['ICitp']
Out[38]: 0.9521486498555631

```
TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Sales_AP'] = -D['Sales_AP'].copy() #Change the sign of price to book because
high is bad, low is good
TD['Sales_AP'] = td.CleanData(TD['Sales_AP'],N) #Moderate outliers so model
building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Sales_AP'] = td.Transform(TD['Sales_AP'],TransformList,D)
X = TD['Sales_AP']

NumTile = np.int_(5)
FA['Sales_AP'] =
ft.AnalyzeFactor('Sales_AP',TD['Sales_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied
C:\Users\91900\Dropbox\My PC (LAPTOP-BOUM6ROT)\Downloads\Mid term 2
project (1) (1)\FactorTester.py:151: RuntimeWarning: Mean of empty slice
R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean
by period

FA.keys()
Out[41]: dict_keys(['Sales_AP'])

FA['Sales_AP'].keys()
Out[42]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset',
'NumPeriod', 'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns',
'NumTile', 'EW_BenchMark_p', 'SignalTile_AP', 'ReturnTile_AP',
'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p', 'GrandMeanIC',
'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn', 'FactorICPval_pn',
'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t', 'BetaBySignalTile_t',
'RsquareBySignalTile_t', 'TstatBySignalTile_t', 'ResidualRiskBySignalTile_t',
'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt', 'CumulativeTileReturn_t',
'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['Sales_AP']['ICitp']
Out[43]: 0.723575265771014
```

Based on the provided Information Coefficient (IC) values, the "Earnings" factor appears to be the strongest predictor of future returns, with an IC value of 0.9521486498555631.

Here are the IC values for each factor:

1. Operating Margin: IC = 0.5855530380173047
2. Earnings: IC = 0.9521486498555631
3. Sales: IC = 0.723575265771014

Therefore, based on the IC values, the "Earnings" factor seems to be the best factor among the ones analyzed. It has the highest IC value, indicating a stronger relationship with future returns compared to the other factors.

DALL = High

```
DHigh=LoadData('High')
Deleting DataHighCap.pkl file from current directory ...
Initializing data dictionary D ...
Loading spreadsheet DataHighCap.xlsm to dictionary D ...
Loading variables by asset and period from sheet data ...
Variables by asset and period from sheet data are loaded.
Number of rows in sheet data: 14320
Number of Assets: 333
Number of Periods: 129
Number of Sectors: 11
Number of Industrys: 25
Number of Variables: 43
Saving D to DataHighCap.pkl
D has been saved to DataHighCap.pkl
Loading from DataHighCap.pkl
Data loaded from DataHighCap.pkl
DHigh.keys()
Out[81]: dict_keys(['Asset', 'Sector_Asset', 'Industry_Asset', 'NumAsset', 'Sector',
'SectorCode', 'NumSector', 'Industry', 'IndustryCode', 'NumIndustry', 'Variable',
'dataNumRows', 'NumVariable', 'Period', 'NumPeriod', 'FCF_Price_AP',
'FinancialLeverage_AP', 'EarningsGrowth1YR_AP', 'CommonStockRating_AP',
'Price_EarningsF12M_AP', 'Sales_AP', 'Beta_AP', 'LongTermGrowth_AP', 'Rho_AP',
'Debt_GrossEV_AP', 'ShortInterestPercentage_AP', 'LTDebtCreditRating_AP',
'AssetTurnover_AP', 'Price_Sales_AP', 'SurpriseMomentum_AP', 'LTMomentum_AP',
'Price_Book_AP', 'Cash_AP', 'Earnings_AP', 'CoefficientVariationEarnings12Qtr_AP',
'DividendYield_AP', 'Month12ChangeF12MEarningsEstimate_AP', 'ROE_AP',
'AnalystAgreementRevisions_AP', 'Price_EarningsL12M_AP', 'EV_EBITDA_AP', 'Price_AP',
'ShareBuyBack_AP', 'Quarter1SurpriseActualEarnings_AP', 'EarningsRevision_AP',
'CovarianceVariationEarningsEstimate_AP', 'QualityOfEarnings_AP', 'SalesGrowth1YR_AP',
'StdDeviationBands_AP', 'NumCompaniesSuprising_AP', 'OperatingMargin_AP',
```

```
'Returns_AP', 'Cash_EV_AP', 'OperatingLeverage_AP', 'Avg30DaydolVolume_AP',
'STMomentum_AP', 'CapX_AP', 'MarketCapitalization_AP', 'ForwardReturns_AP'])
FActore-1
TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Price_Book_AP'] = -D['Price_Book_AP'].copy() #Change the sign of price to book
because high is bad, low is good
TD['Price_Book_AP'] = td.CleanData(TD['Price_Book_AP'],N) #Moderate outliers so model
building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Price_Book_AP'] = td.Transform(TD['Price_Book_AP'],TransformList,D)
X = TD['Price_Book_AP']

NumTile = np.int_(5)
FA['Price_Book_AP'] =
ft.AnalyzeFactor('Price_Book',TD['Price_Book_AP'],TD['ForwardReturns_AP'],NumTile)

N = np.int_(5)
TD['SignalName'] = D['SignalName'].copy()
TD['SignalName'] = td.CleanData(TD['SignalName'],N) #Moderate outliers so model building
is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['SignalName'] = td.Transform(TD['SignalName'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['SignalName'] =
ft.AnalyzeFactor('SignalName',TD['SignalName'],TD['ForwardReturns_AP'],NumTile)
FA['Price_Book_AP']['ICitp']
```

Out[88]: 0.655915725445113

```
TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Sales_AP'] = -D['Sales_AP'].copy() #Change the sign of price to book because high is
bad, low is good
TD['Sales_AP'] = td.CleanData(TD['Sales_AP'],N) #Moderate outliers so model building is
better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
```

TD['Sales_AP'] = td.Transform(TD['Sales_AP'],TransformList,D)
X = TD['Sales_AP']

NumTile = np.int_(5)
FA['Sales_AP'] =
ft.AnalyzeFactor('Sales_AP',TD['Sales_AP'],TD['ForwardReturns_AP'],NumTile)
FA['Sales_AP']['ICitp'] Out[97]: 0.8685708364288796

'Earnings_AP'
TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Earnings_AP'] = -D['Earnings_AP'].copy() #Change the sign of price to book because high is bad, low is good
TD['Earnings_AP'] = td.CleanData(TD['Earnings_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Earnings_AP'] = td.Transform(TD['Earnings_AP'],TransformList,D)
X = TD['Earnings_AP'
NumTile = np.int_(5)
FA['Earnings_AP'] =
ft.AnalyzeFactor('Earnings_AP',TD['Earnings_AP'],TD['ForwardReturns_AP'],NumTile)

FA['Earnings_AP']['ICitp']= 0.864259099722394

N = np.int_(5)
TD['Price_EarningsF12M_AP'] = D['Price_EarningsF12M_AP'].copy()
TD['Price_EarningsF12M_AP'] = td.CleanData(TD['Price_EarningsF12M_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Price_EarningsF12M_AP'] =
td.Transform(TD['Price_EarningsF12M_AP'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['Price_EarningsF12M_AP'] =
ft.AnalyzeFactor('Price_EarningsF12M_AP',TD['Price_EarningsF12M_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied

FA['Price_EarningsF12M_AP']['ICitp']
Out[42]: 0.5410687843750408

Based on the provided Information Coefficient (IC) values, the "Sales" factor appears to be the strongest predictor of future returns, with an IC value of 0.8685708364288796.

Here are the IC values for each factor:

1. Price to Book (Price_Book_AP): IC = 0.655915725445113
2. Sales (Sales_AP): IC = 0.8685708364288796
3. Earnings (Earnings_AP): IC = 0.864259099722394
4. Price to Earnings (Price_EarningsF12M_AP): IC = 0.5410687843750408

Therefore, based on the IC values, the "Sales" factor seems to be the best predictor of future returns among the factors analyzed. It has the highest IC value, indicating a stronger relationship with future returns compared to the other factors.

DALL = Medium

```
 TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['Price_Book_AP'] = -D['Price_Book_AP'].copy() #Change the sign of price to book
because high is bad, low is good
TD['Price_Book_AP'] = td.CleanData(TD['Price_Book_AP'],N) #Moderate outliers so model
building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Price_Book_AP'] = td.Transform(TD['Price_Book_AP'],TransformList,D)
X = TD['Price_Book_AP']

NumTile = np.int_(5)
FA['Price_Book_AP'] =
ft.AnalyzeFactor('Price_Book',TD['Price_Book_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied

FA.keys()
Out[7]: dict_keys(['Price_Book_AP'])
```

```
FA['Price_Book_AP'].keys()
Out[8]: dict_keys(['SignalName', 'Signal_AP', 'Returns_AP', 'NumAsset', 'NumPeriod',
'AnnPeriods', 'MinNumForCorrelation', 'NumForwardReturns', 'NumTile', 'EW_BenchMark_p',
'SignalTile_AP', 'ReturnTile_AP', 'RetBySignalTile_pt', 'ICByPeriod_p', 'ICByPeriodPval_p',
'GrandMeanIC', 'GrandStdIC', 'ICitp', 'ICdecay_n', 'ICdecaySTD_n', 'FactorIC_pn',
'FactorICPval_pn', 'IRdecay_n', 'ICdecayWeighted', 'AlphaBySignalTile_t',
'BetaBySignalTile_t', 'RsquareBySignalTile_t', 'TstatBySignalTile_t',
'ResidualRiskBySignalTile_t', 'ResidualBySignalTile_pt', 'ExcessRetBySignalTile_pt',
'CumulativeTileReturn_t', 'CumulativeTileExcessReturn_t', 'ExPostIR_t'])

FA['Price_Book_AP']['ICitp']
Out[9]: 0.7364488122675225



TD = {} #Dictionary for transformed data
FA = {} #Dictionary for factors
N = np.int_(5) #Used to control data cleaning, see notes
#Next 4 lines were added so forward returns will be used
Returns_AP = D['Returns_AP'].copy()
TD['ForwardReturns_AP'] = D['ForwardReturns_AP'].copy()

TD['OperatingMargin_AP'] = -D['OperatingMargin_AP'].copy() #Change the sign of price to
book because high is bad, low is good
TD['OperatingMargin_AP'] = td.CleanData(TD['OperatingMargin_AP'],N) #Moderate outliers
so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['OperatingMargin_AP'] = td.Transform(TD['OperatingMargin_AP'],TransformList,D)
X = TD['OperatingMargin_AP']

NumTile = np.int_(5)
FA['OperatingMargin_AP'] =
ft.AnalyzeFactor('OperatingMargin_AP',TD['OperatingMargin_AP'],TD['ForwardReturns_AP']
,NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied

FA['OperatingMargin_AP']['ICitp']
Out[11]: 0.39924875175895813


TD['CovarianceVariationEarningsEstimate_AP'] =
D['CovarianceVariationEarningsEstimate_AP'].copy()
TD['CovarianceVariationEarningsEstimate_AP'] =
td.CleanData(TD['CovarianceVariationEarningsEstimate_AP'],N) #Moderate outliers so
```

model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['CovarianceVariationEarningsEstimate_AP'] =
td.Transform(TD['CovarianceVariationEarningsEstimate_AP'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['CovarianceVariationEarningsEstimate_AP'] =
ft.AnalyzeFactor('CovarianceVariationEarningsEstimate_AP',TD['CovarianceVariationEarningsEstimate_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied

FA['CommonStockRating_AP']['ICitp']
Out[21]: 0.5013412032176241

N = np.int_(5)
TD['Beta_AP'] = D['Beta_AP'].copy()
TD['Beta_AP'] = td.CleanData(TD['Beta_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Beta_AP'] = td.Transform(TD['Beta_AP'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['Beta_AP'] =
ft.AnalyzeFactor('Beta_AP',TD['Beta_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied
C:\Users\smahadik2\Downloads\Mid term 2 project (1)\FactorTester.py:151:
RuntimeWarning: Mean of empty slice
R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA['Beta_AP']['ICitp']
Out[23]: 0.08877256863249926

TD['Beta_AP'] = D['Beta_AP'].copy()
TD['Beta_AP'] = td.CleanData(TD['Beta_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['Beta_AP'] = td.Transform(TD['Beta_AP'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['Beta_AP'] =
ft.AnalyzeFactor('Beta_AP',TD['Beta_AP'],TD['ForwardReturns_AP'],NumTile)

Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied
C:\Users\smahadik2\Downloads\Mid term 2 project (1)\FactorTester.py:151:
RuntimeWarning: Mean of empty slice
R['EW_BenchMark_p'] = np.nanmean(Returns_AP,axis=0)#equaly weighted mean by period

FA['Beta_AP']['ICitp']
Out[23]: 0.08877256863249926

FA['Beta_AP']['ICitp']
Out[23]: 0.08877256863249926

N = np.int_(5)
TD['FCF_Price_AP'] = D['FCF_Price_AP'].copy()
TD['FCF_Price_AP'] = td.CleanData(TD['FCF_Price_AP'],N) #Moderate outliers so model building is better
TransformList = ['ZSBT','ZSCSBS'] #z-score by time, decile crosssection by sector
TD['FCF_Price_AP'] = td.Transform(TD['FCF_Price_AP'],TransformList,D)
NumTile = np.int_(5) #Number of quantiles to use in AnalyzeFactor
FA['FCF_Price_AP'] =
ft.AnalyzeFactor('FCF_Price_AP',TD['FCF_Price_AP'],TD['ForwardReturns_AP'],NumTile)
Applying transformations ...
ZSBT
z-score by time
ZSCSBS
z-score by crosssection by sector
Transformations applied

FA['FCF_Price_AP']['ICitp']
Out[25]: 0.1769246772456932

Based on the provided Information Coefficient (IC) values, the "Price to Book" factor appears to be the strongest predictor of future returns among the factors analyzed. It has an IC value of 0.7364488122675225.

Here are the IC values for each factor:

1. Price to Book (Price_Book_AP): IC = 0.7364488122675225

2. Operating Margin (OperatingMargin_AP): IC = 0.39924875175895813

3. Covariance Variation of Earnings Estimate (CovarianceVariationEarningsEstimate_AP): IC = 0.5013412032176241

4. Beta (Beta_AP): IC = 0.08877256863249926

5. Free Cash Flow to Price (FCF_Price_AP): IC = 0.1769246772456932

Therefore, based on the IC values, the "Price to Book" factor seems to be the best predictor of future returns among the factors analyzed. It has the highest IC value, indicating a stronger relationship with future returns compared to the other factors.

Based on the provided Information Coefficient (IC) values for the "Price to Book" factor under different market conditions, here's a summary:

1. **DALL = All:**

   - IC value for Price to Book: 0.655915725445113

2. **DALL = High:**

   - IC value for Price to Book: 0.655915725445113

3. **DALL = Medium:**

   - IC value for Price to Book: 0.736448 (Truncated)

It seems that under all market conditions (All, High, and Medium), the Price to Book factor demonstrates consistent predictive power, with relatively high IC values across the board. This suggests that Price to Book ratio is a robust predictor of future returns regardless of the market environment.