Assignment: ECO764A(Financial Econometrics)

Team Members

Apoorv Chauhan	180129
Dwaipayan Karak	180257
Himank Kothari	180291
Krishn Kumar Maddhesiya	180361
Rakesh Kumar Parewa	180587
Rohit Kumar	180624
Shivani Jha	180727
Vineet Kumar	180867
Vishvajeet Kumar Gond	180875

ABSTRACT

Our aim is to calculate and analyse the Capital Asset Pricing Model and Fama and French 3 factor model and Carhart four-factor model for …….. A proper understanding of the working of above three models is necessary to our assessment and final results. Later, we will also examine the results obtained from various models and how they are relevant to our analysis.

Capital Asset Pricing Model (CAPM)

The Capital Asset Pricing Model (CAPM) describes the relationship between systematic risk and expected return for assets, particularly stocks. It is widely used throughout finance for pricing risky securities and generating expected returns for assets given the risk of those assets and cost of capital.

$R_{it} - R_{ft} = \alpha_p + \beta_{mkt}(R_{mt} - R_{ft})$ where,

$(R_{it} - R_{ft})$= expected return of investment,

α_{it} = risk-free rate

β_{mkt} = beta of the investment

$(R_{mt} - R_{ft})$= market risk premium

Fama and French Three Factor Model

The Fama and French Three-Factor Model (or the Fama French Model for short) is an asset pricing model developed in 1992 that expands on the capital asset pricing model (CAPM) by adding size risk value risk factors to the market risk factor in CAPM. This model considers the fact that value and small-cap stocks outperform markets on a regular basis. By including these two additional factors, t model adjusts for this outperforming tendency, which is thought to make it a better tool for evaluat manager performance. The formula for the Fama French model is

$R_{it} - R_{ft} = \alpha_p + \beta_{mkt}(R_{mt} - R_{ft}) + \beta_{SMT}(SMB) + \beta_{HML}(HML)$ where,

R_{it} = Total return of a stock or portfolio i at time t

R_{ft}= risk free rate of return at time t

R_{mt}= total market portfolio return at time t

$(R_{pt} - R_{ft})$= expected excess return

$(R_{mt} - R_{ft})$= excess return on the market portfolio (index)

SMB= size premium (small minus big)

HML= value premium (high minus low)

Carhart Four Factor model

The Carhart four-factor model is an extra factor addition in the Fama–French three-factor model including a momentum factor for asset pricing of stocks, proposed by Mark Carhart. Momentum in stock is described as the tendency for the stock price to continue rising if it is going up and to conti declining if it is going down.

$R_{it} - R_{ft} = \alpha_p + \beta_{mkt}(R_{mt} - R_{ft}) + \beta_{SMT}(SMB) + \beta_{HML}(HML) + \beta_{MOM}(MOM)$ where,

$(R_{it} - R_{ft})$= expected excess return

$(R_{mt} - R_{ft})$= excess return on the market portfolio (index)

SMB= size premium (small minus big)

HML= value premium (high minus low)

MOM= Momentum (average return on the two high prior return portfolios
minus the average return on the two low prior return portfolios)

Data Analysis Steps:

We can construct SMB, HML and MOM factors following the steps:

(i) Calculate the excess returns, i.e. $R_{it} - R_{ft}$

(ii) Sort the excess returns on Size (market capitalization, denoted as ***S***) and then on value (ROE).

(vii) Use the formula to calculate SMB:

SMB = 1/3 (small value + small neutral + small growth) – 1/ 3 (big value + big neutral +big growth)

SMB = 1/3(S_1V11 + S1V2 + S1V3) – 1/ 3 (S3V1 + S3V2 + S3V3)

Code -

```
impor
t
panda
s as
pd
        from pandas import ExcelWriter
        from pandas import ExcelFile
        import numpy as np
        from sklearn.linear_model import LinearRegression
        import matplotlib.pyplot as plt
        import random as rand
        from datetime import datetime
        import sys



        eps = 1
        r_f = 0.5
        if(len(sys.argv)!=2 or (sys.argv[1] == "help")):
               # Inp format
               print("python3 solver.py help : For help")
               print("python3 solver.py plot : Generate plots")
               print("python3 solver.py appendix : Generate tables")
               exit(1)
```

```
def checkifnan(num):
	if num is np.nan:
		return 0
	else:
		return num


def calculateWeightedParam(companies,listofcompanies,x):
	totalsum = 0.0
	capsum = 0.0
	for company_name in listofcompanies:
		totalsum +=
companies[company_name]['CAP'][x]*companies[company_name]['PIreturn'][x]
		capsum += companies[company_name]['CAP'][x]
	if capsum == 0:
		return 0
	return (totalsum/(capsum))




df = pd.read_excel('Canada50.xlsx')
columns = list(df.loc[2])[1:]
row_count = df.shape[0] - 5
companies = dict()
j = 0
company_name = ""
prev_err = False


date=[]
l=df.iloc[4:(4+row_count),0].tolist()
```

```
for x in range(row_count):
	date.append(str(l[x]).split()[0])




for i,col in enumerate(columns):
	idx = i + 1
	if(col == "#ERROR"):
		prev_err = True
		j = (j+1)%3
		continue
	elif(j == 0):
		company_name = col.split('-')[0].strip()
		companies[company_name] = dict()
		# CAP
		a = np.nan_to_num(np.array(df.iloc[4:(4+row_count),idx]).astype(float))
		companies[company_name]['CAP'] = a


	elif(j == 1):
		# ROI
		if(prev_err == True):
			company_name = col.split('-')[0].strip()
			if(company_name not in companies):
				companies[company_name] = dict()
		a = np.nan_to_num(np.array(df.iloc[4:(4+row_count),idx]).astype(float))
		companies[company_name]['ROI'] = a
	elif(j == 2):
		# PI
		if(prev_err == True):
			company_name = col.split('-')[0].strip()
			if(company_name not in companies):
				companies[company_name] = dict()
```

```
                a = np.nan_to_num(np.array(df.iloc[4:(4+row_count),idx]).astype(float))
                companies[company_name]['PI'] = a


        j = (j+1)%3
        prev_err = False
for company_name in companies:
        if 'PI' not in companies[company_name]:
                companies[company_name]['PI'] = np.array([0.]*row_count)
        if 'ROI' not in companies[company_name]:
                companies[company_name]['ROI'] = np.array([0.]*row_count)
        if 'CAP' not in companies[company_name]:
                # print(company_name)
                companies[company_name]['CAP'] = np.array([0.]*row_count)


marketcaparr = []
priceindexarr = []
growtharr = []
booktomarketarr = []
companylist =[]
smblist = []
hmllist = []
flist=[]
momlist=[]
t=[]
y_r=[]


for company_name  in companies:
        companies[company_name]['YR'] = []
        for x in range(row_count):
                companies[company_name]['YR'].append(0.0)
```

```
for x in range(0,row_count,12):
	for company_name in companies:
		# companies[company_name]['YR'] = []
		# for  iters in range(0,row_count):
		#	companies[company_name]['YR'].append(0.0)
		for y in range(12):
			if y == 0:
				if x == 0:
					if companies[company_name]['PI'][x] > 0:
						companies[company_name]['YR'][x+y] =
100.00*(companies[company_name]['PI'][x+11]-companies[company_name]['PI'][x])/(companies
[company_name]['PI'][x])
				elif x > 0:
					if companies[company_name]['PI'][x-11] > 0:
						companies[company_name]['YR'][x+y] =
100.00*(companies[company_name]['PI'][x]-companies[company_name]['PI'][x-11])/(companies
[company_name]['PI'][x-11])
			elif y > 0:
				if x == 0:
					if companies[company_name]['PI'][x] > 0:
						companies[company_name]['YR'][x+y] =
100.00*(companies[company_name]['PI'][x+11]-companies[company_name]['PI'][x])/(companies
[company_name]['PI'][x])
				elif x > 0:
					if companies[company_name]['PI'][x-11] > 0:
						companies[company_name]['YR'][x+y] =
100.00*(companies[company_name]['PI'][x]-companies[company_name]['PI'][x-11])/(companies
[company_name]['PI'][x-11])




for company_name in companies:
```

```
        companylist.append(company_name)


three_factors = []
four_factors = []
new_mom = []
capm=[]
# print(row_count)


for x in range(row_count):
        marketcap = []
        priceindex = []
        growth = []
        booktomarket = []
        pireturnarr = []
        new_mom_arr = []


        for company_name in companies:
                if(x == 0):
                        companies[company_name]['PIreturn'] = []
                        companies[company_name]['PIreturn'].append(0.0)


                marketcap.append(companies[company_name]['CAP'][x])
                growth.append(companies[company_name]['ROI'][x])
                if x > 0:
                        if companies[company_name]['PI'][x-1] ==0:
                                pireturn=0
                        else:
                                pireturn =
100.00*(companies[company_name]['PI'][x]-companies[company_name]['PI'][x-1])/(companies[
company_name]['PI'][x-1])
                        companies[company_name]['PIreturn'].append(pireturn)
```

```
        pireturnarr.append(companies[company_name]['PIreturn'][x])
        booktomarket.append(companies[company_name]['ROI'][x])
        new_mom_arr.append(companies[company_name]['YR'][x])



high = np.percentile(booktomarket,70)
low = np.percentile(booktomarket,30)
small = np.percentile(marketcap,30)
big = np.percentile(marketcap,70)


win = np.percentile(new_mom_arr,70)
lose = np.percentile(new_mom_arr,30)


valuefirm = []
growthfirm = []
neutralfirm = []
smallfirm = []
bigfirm = []
winnerfirm = []
loserfirm = []


for y in range(len(companylist)):
        if marketcap[y] > big:
                bigfirm.append(companylist[y])
        elif marketcap[y] < small:
                smallfirm.append(companylist[y])
        if booktomarket[y] < low:
                growthfirm.append(companylist[y])
        elif booktomarket[y] > low and booktomarket[y] < high:
                neutralfirm.append(companylist[y])
```

```
        elif booktomarket[y] > high:
                valuefirm.append(companylist[y])
        if new_mom_arr[y] > win:
                winnerfirm.append(companylist[y])
        elif new_mom_arr[y] < lose:
                loserfirm.append(companylist[y])



BV = list(set(bigfirm) & set(valuefirm))
BN = list(set(bigfirm) & set(neutralfirm))
BG = list(set(bigfirm) & set(growthfirm))
SV = list(set(smallfirm) & set(valuefirm))
SN = list(set(smallfirm) & set(neutralfirm))
SG = list(set(smallfirm) & set(growthfirm))
WB = list(set(winnerfirm) & set(bigfirm))
WS = list(set(winnerfirm) & set(smallfirm))
LB = list(set(loserfirm) & set(bigfirm))
LS = list(set(loserfirm) & set(smallfirm))


if len(BV)> 0:
        bvreturn = calculateWeightedParam(companies,BV,x)
else:
        bvreturn = 0


if len(BN)> 0:
        bnreturn = calculateWeightedParam(companies,BN,x)
else:
        bnreturn = 0


if len(BG)> 0:
```

```
        bgreturn = calculateWeightedParam(companies,BG,x)
    else:
        bgreturn = 0



    if len(SV)> 0:
        svreturn = calculateWeightedParam(companies,SV,x)
    else:
        svreturn = 0


    if len(SN)> 0:
        snreturn = calculateWeightedParam(companies,SN,x)
    else:
        snreturn = 0


    if len(SG)> 0:
        sgreturn = calculateWeightedParam(companies,SG,x)
    else:
        sgreturn = 0


    if len(WB)>0:
        wbreturn = calculateWeightedParam(companies,WB,x)
    else:
        wbreturn = 0


    if len(WS)>0:
        wsreturn = calculateWeightedParam(companies,WS,x)
    else:
        wsreturn = 0
```

```
if len(LB)>0:
	lbreturn = calculateWeightedParam(companies,LB,x)
else:
	lbreturn = 0


if len(LS)>0:
	lsreturn = calculateWeightedParam(companies,LS,x)
else:
	lsreturn = 0


smb = (svreturn+snreturn+sgreturn)/3 - (bvreturn+bnreturn+bgreturn)/3
smblist.append(smb)


hml = (svreturn+bvreturn)/2 - (sgreturn+bgreturn)/2
hmllist.append(hml)


f = calculateWeightedParam(companies,companylist,x)
flist.append(f)


mom = (wsreturn+wbreturn)/2 - (lsreturn+lbreturn)/2
momlist.append(mom)


three_factors.append([f-r_f,hml,smb])
capm.append([f-r_f])
four_factors.append([f-r_f,hml,smb,mom])


t.append(x)
```

```
C = np.array(capm, np.float32)
Z = np.array(four_factors, np.float32)
X = np.array(three_factors, np.float32)


smblist[0]=1
for i in range(1,len(smblist)):
       smblist[i]=smblist[i]/100+smblist[i-1]


hmllist[0]=1
for i in range(1,len(hmllist)):
       hmllist[i]=hmllist[i]/100+hmllist[i-1]


flist[0]=1
for i in range(1,len(flist)):
       flist[i]=flist[i]/100+flist[i-1]


momlist[0]=1
for i in range(1,len(momlist)):
       momlist[i]=momlist[i]/100+momlist[i-1]


for i in range(len(date)):
       date[i] = datetime.strptime(date[i], '%Y-%m-%d')
#print(date)
if(sys.argv[1] == "plot"):
       plt.plot(date,smblist,label='SMB')
       plt.plot(date,hmllist,label='HML')
       plt.plot(date,flist,label='F')
```

```
        plt.title('SMB+HML+F vs date')
        plt.xlabel('Date')
        plt.ylabel('Commulative return')
        plt.locator_params(axis='x', nbins=10)
        plt.legend(loc=2)
        plt.show()


if(sys.argv[1] == "appendix"):
        file_w = open("FF3factor.txt", "w")
        for company_name in companies:
                y = np.array(companies[company_name]['PIreturn'])
                y = y - r_f
                # print(y)
                model = LinearRegression().fit(X, y)
                r_sq = model.score(X, y)
                file_w.write(company_name+" & "+str(round(model.intercept_,3))+" &
"+str(round(model.coef_[0],3))+" & "+str(round(model.coef_[1],3))+" &
"+str(round(model.coef_[2],3))+" \\\\ \n")


        print("Latex Table stored for FamaFench 3 factor model in FF3factor.txt ...")
        file_w.close()
        file_w = open("C4factor.txt", "w")
        for company_name in companies:
                y = np.array(companies[company_name]['PIreturn'])
                y = y - r_f
                model = LinearRegression().fit(Z, y)
                r_sq = model.score(Z, y)
                file_w.write(company_name+" & "+str(round(model.intercept_,3))+" &
"+str(round(model.coef_[0],3))+" & "+str(round(model.coef_[1],3))+" &
"+str(round(model.coef_[2],3))+" & "+str(round(model.coef_[3],3))+" \\\\ \n")


        print("Latex Table stored for Carhart 4 factor model in C4factor.txt ...")
```

```
        file_w.close()
        file_w = open("CAPM1factor.txt", "w")
        for company_name in companies:
            y = np.array(companies[company_name]['PIreturn'])
            y = y - r_f
            # print(y)
            model = LinearRegression().fit(C, y)
            r_sq = model.score(C, y)
            file_w.write(company_name+" & "+str(round(model.intercept_,3))+" &
"+str(round(model.coef_[0],3))+" \\\\ \n")
        print("Latex Table stored for CAPM 1 factor model in CAPM1factor.txt ...")
        file_w.close()
```

	Market Return (Rm)	Avg Portfolio Return (Rp)	Rf	Rp-Rf	Rm-Rf	SMB	HML	MOM
Sept' 18	-1.387509202	-2.23422	2.36	-4.59422	-3.747509202	2.121002	-12.1321352	-9.153465135
Oct'18	-0.899606647	-1.75044	2.46	-4.21044	-3.359606647	0.144664	-33.3455	-7.52346
Nov' 18	-3.761667439	-1.86314	2.47	-4.33314	-6.231667439	0.736324	-27.4682	1.009602
Dec' 18	-3.730392805	-1.57844	2.39	-3.96844	-6.120392805	15.45562	-23.3605	11.7178
Jan'19	-1.292386632	0.583904	2.12	-1.537096	-3.413386632	1.208663	-8.69164	5.921234
Feb' 19	6.803096732	6.255241	2.11	4.145241	4.693096732	0.271638	44.99327	4.320739
Mar' 19	2.294664562	2.101445	2.04	0.061445	0.254664562	0.845132	-1.2546343	-0.7598965
Apr'19	2.537392908	2.88603	1.96	0.92603	0.577392908	1.005692394	-2.141460708	-1.812966722
may'	-0.060296258	0.038436	1.8	-1.7615	-1.860296	6.914602	-12.22360	-10.65732

19				64	258	156	922	028
June'19	-1.020225618	0.234454	1.68	-1.445546	-2.700225618	0.910448373	-8.412954022	-5.036550115
july'19	1.419187205	1.549995	1.7	-0.150005	-0.280812795	-0.086452818	0.105774731	-5.100704992
Aug'19	-0.355488535	-1.17091	1.4	-2.57091	-1.755488535	-4.835486649	-18.80012967	2.866927452
Sept'19	0.79417877	1.316728	1.5	-0.183272	-0.70582123	-0.040128601	0.355350931	1.983224189
Oct'19	-1.470543063	-0.79563	1.65	-2.44563	-3.120543063	1.706027848	-1.911576242	3.483679979

CURRENCY	C$	Market Return (Rm)
2/8/2016	12535.4	
3/8/2016	13311.05	6.00378756
4/8/2016	13396.73	0.641612998
5/8/2016	13701.47	2.249247902
6/8/2016	14313.1	4.367207549
7/8/2016	14259.84	-0.372800711
8/8/2016	14755.62	3.417683244
9/8/2016	14803.26	0.322339962
10/8/2016	14566.26	-1.613953136
11/8/2016	14656.8	0.619922528

	4	
12/8/2016	15295.2	4.263193299
1/8/2017	15496.05	1.304609905
2/8/2017	15554.04	0.373525922
3/8/2017	15496.98	-0.367524572
4/8/2017	15667.13	1.091972084
5/8/2017	15652.08	-0.096107157
6/8/2017	15423.09	-1.473807834
7/8/2017	15027.16	-2.600650668
8/8/2017	15256.35	1.513657934
9/8/2017	14985.32	-1.79247546
10/8/2017	15728.32	4.839185385
11/8/2017	16105.35	2.368860585
12/8/2017	16096.07	-0.057637211
1/8/2018	16317.65	1.367220126
2/8/2018	15065.61	-7.98326811
3/8/2018	15538.7	3.091902339

4/8/2018	15207.41	-2.155087729
5/8/2018	15842.71	4.092664855
6/8/2018	16202.69	2.246782023
7/8/2018	16371.78	1.038184313
8/8/2018	16315.08	-0.346928747
9/8/2018	16090.27	-1.387509202
10/8/2018	15946.17	-0.899606647
11/8/2018	15357.47	-3.761667439
12/8/2018	14795.13	-3.730392805
1/8/2019	14605.15	-1.292386632
2/8/2019	15633.33	6.803096732
3/8/2019	15996.21	2.294664562
4/8/2019	16407.29	2.537392908
5/8/2019	16397.4	-0.060296258
6/8/2019	16230.96	-1.020225618
7/8/2019	16462.95	1.419187205
8/8/2019	16404.5	-0.355488535

	3	
9/8/2019	16535.33	0.79417877
10/8/2019	16293.95	-1.470543063
11/8/2019	16877.42	3.518276061
12/8/2019	16996.97	0.705845899
1/8/2020	17167.82	1.000160833
2/8/2020	17655.49	2.801008196
3/8/2020	16175.02	-8.757870631
4/8/2020	13925.71	-14.97313049
5/8/2020	14966.56	7.208160716
6/8/2020	15974.91	6.520098734
7/8/2020	15629.19	-2.187904677
8/8/2020	16544.48	5.691219177
9/8/2020	16099.52	-2.726305356
10/8/2020	16534.54	2.666206839
11/8/2020	16282.83	-1.534034788
12/8/2020	17639	8.000118124

1/8/2021	18042.07	2.259389343
2/8/2021	18330.26	1.584699309

SUMMARY OUTPUT

Regression Statistics

Multiple R	0.932199702
R Square	0.868996284
Adjusted R Square	0.858079308
Standard Error	0.915784278
Observations	14

ANOVA

	df	SS	MS	F	Significance F
Regression	1	66.75778505	66.75778505	79.60045534	1.21041E-06
Residual	12	10.06393013	0.838660844		
Total	13	76.82171518			

	Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%

Intercept	0.007213949	0.302330569	0.023861135574	0.98135	-0.651507773	0.665935671	-0.651507773	0.665935671
Rm-Rf	0.798273911	0.089473445	8.921908728	1.21041E-06	0.603328021	0.993219801	0.603328021	0.993219801

SUMMARY OUTPUT

Regression Statistics

Multiple R	0.954994753
R Square	0.912014979
Adjusted R Square	0.885619473
Standard Error	0.822141121
Observations	14

ANOVA

	df	SS	MS	F	Significance F
Regression	3	70.06255494	23.35418498	34.551903 16	1.37402E-05
Residual	10	6.759160233	0.675916023		
Total	13	76.82171518			

	Coefficie	Standard	t Stat	P-value	Lower	Upper	Lower	Upper

	nts	Error			95%	95%	95.0%	95.0%
Intercept	-0.17920896	0.285606983	-0.627467011	0.544413612	-0.815580976	0.457163056	-0.815580976	0.457163056
Rm-Rf	0.580338049	0.17906701	3.240898758	0.008855926	0.181351888	0.979324211	0.181351888	0.979324211
SMB	0.043155565	0.057848489	0.746010242	0.472833469	-0.085738	0.172050003	-0.085738	0.172050003
HML	0.043914678	0.025518401	1.720902386	0.116003621	-0.012943864	0.100773219	-0.012943864	0.100773219

SUMMARY OUTPUT

Regression Statistics

Multiple R	0.963669302
R Square	0.928658524
Adjusted R Square	0.896951201
Standard Error	0.780354232
Observations	14

ANOVA

	df	SS	MS	F	Significance F
Regression	4	51.34114	17.83528	29.2884	3.58329E

		063	516	5601	-05
Residual	9	5.480574543	0.608952727		
Total	13	76.82171518			

	Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%
Intercept	-0.057187019	0.283868633	-0.201455927	0.844821694	-0.699348248	0.584968443	-0.699348248	0.584968443
Rm-Rf	0.677182028	0.182633859	3.707866831	0.004861002	0.264035535	1.090328521	0.264035535	1.090328521
SMB	0.034116768	0.055261423	0.617370424	0.552287954	-0.090893255	0.159126792	-0.090893255	0.159126792
HML	0.027468043	0.026748879	1.026885752	0.331281002	-0.033042126	0.087978211	-0.033042126	0.087978211
MOM	0.056724253	0.039146745	1.449015831	0.181263714	-0.031831836	0.145280341	-0.031831836	0.145280341