3 CORPORATE REPORTING DATA AND ANALYSES

If we only have observations of the speed performances of sports cars, we can try to predict their next speed test results with a different set of road conditions. However, our predictions will reasonably be improved if we also know how the cars were made, their engine structures, gear transmission dynamics, wheel types, parts, and so on. Similarly, we saw that we could improve in predicting a company's performance – its earnings, stock returns (for a listed stock), and so on – not by just knowing the past historical data of these performances, but also by knowing the structure and business operations of the company. The latter are captured largely in accounting statements that are mandatory reporting statements to be made by the company, at least for listed companies that tap on funding from the public.

A company can produce its accounting statements as frequently as monthly for its internal use, but official submissions to the regulators as public documents for public access, and hence availing such data for data analyses, are typically on standard time frames in U.S. The U.S. Securities and Exchange Commission (SEC) requires publicly listed firms to submit annual reports called 10-K forms/filings that include audited financial statements. The firms are also required to disclose corporate performance information on an ongoing quarterly basis in 10-Q forms.

In this chapter, we show how accounting variables can be used to predict future (following year's) earnings or sometimes profit margins.

3.1 Financial Statements

There are 3 key financial reporting statements in the corporate reports – the Balance Sheet, the Income Statement, and the Cashflow Statement. The Balance Sheet is a snapshot of a company's assets and liabilities at a specific time point. The Income Statement is a record of revenues, expenses, and net profits incurred by a company over the specified reporting period, e.g., a year or a quarter. The Cashflow Statement is a report of the aggregated sources and uses of cashflows of the company, including operating cashflows, investing cashflows, and financing cashflows.

The basic ingredients of a Balance Sheet are as follows in a fictitious firm Z. The units are in $ millions. Year is YY.

Z Inc.
Balance Sheet
(As of) December 31, YY

Assets		Liabilities and Equity	
Current Assets		Current Liabilities	
Cash	10	Accounts Payable	10
Accounts Receivable	30	Short-term Debt⁺	10
Inventories	20	Taxes Payable	5
Prepaid Expense*	10	Total Current Liabilities	25
Total Current Assets	70		
		Long-Term Debt	45
Investments**	10		
		Total Liabilities	70
Property, Plant, & Equipment			
Land & Buildings	50		
Equipment	20	Stockholder's Equity	
Less accumulated dep'n	(10)	Paid-in Capital	70
Net PP&E	60	Retained Earnings	20
		Total Owner's Equity	90
Intangible Assets			
Goodwill	10		
Tradenames	10		
Total Intangible Assets	20		
Total Assets	160	Total Liabilities and Equity	160

*Insurance coverage, leased office equipments, etc. where expenses have been paid but advantages will accrue over time in incomes. These are recognized as assets. ** Long-term over 1 year, e.g., long bonds. ⁺ Such as bank overdrafts payable in less than a year whereas long-term debts are due in more than a year. Note: dep'n stands for "depreciation" – this amount is not expensed yet, i.e., not yet a cash outflow, but is typically kept to pay for replacement of the capital in due time.

The Balance Sheet shows the net working capital (NWC) that is the company's current assets less current liabilities. Positive NWC enables the company to fund its current operations on an ongoing daily basis as the current assets can be easily liquidated for cash or enable more short-term loans.

The Balance Sheet also indicates the Invested Capital (total long-term debt and equity) = NWC + PP&E + Intangibles. The company's net profits should be measured versus the invested capital to see the return rate of capital amongst other measures. The amount of long-term debt versus equity also provides a measure of the leverage, whether the company is excessively using debt to fund its business operations, and whether this may become detrimental when interest costs of debt rise.

The basic ingredients of an Income Statement are as follows in the fictitious firm Z. The units are in $ millions. Report is for Year ended December 31, YY.

Z Inc.
Income Statement
For the year ended December 31, YY

Sales	110
Cost of Goods Sold (COGS)	80
Gross Profit (Total Income)	30
Selling, General & Admin Expenses (SG&A)	12
Depreciation & Amortization* (D&A)	5
Total Expenses	17
Operating Profit/Income (EBIT**)	13
Net Non-operating Revenue+	3
Less Interest Expense	(4)
Net Income before tax	12
Less Income Tax	(2)
Net Income (EAIT++)	10

*Depreciation is expensing a fixed asset to reflect its deterioration and to save the cash and accumulate over time to pay for replacement of the capital in due time. Amortization is the spreading of cost of an intangible asset over time. **Earnings before interest and tax. +Non-operating revenue includes interest received from investments, and gains (losses) in investments. ++ Earnings after interest and tax. Note here income tax rate is 2/12 or about 16 2/3%.

The Income (or Profit & Loss) Statement reflects if the company is profitable in that period. The time series of income statements also show the trend in sales and various costs, and thus gauge the sustainability of the business operations. Ratios of income items such as net income over total equity, return on equity (ROE) or net income over sales, operating margin, show how profitable the business is. These can be compared across different companies to see how well they perform relative to one another. Other financial ratios such as sales in Income Statement over total assets in the Balance Sheet – asset turnover ratio – indicate the company's efficiency or effectiveness in producing sales from assets. One key ratio in valuation of a company's share for mergers and acquisition, takeover, or leveraged buyout of a company's shares is its Price earnings ratio (P/E) where P is the share price and E is the earnings per share (EPS) or net earnings (EAIT, excluding extraordinary items) divided by total shares outstanding. For companies not facing credit risk, a low P/E often signal a good buy.

Z Inc.
Cash Flow Statement
For the year ended December 31, YY

Add Net Income	10
Add Non-cash charges (D&A)	5
Less Increase in Current Assets	(20)
Add Increase in Current Liabilities	15
Operating Cashflow	10
Less Capital Expenditures	(20)
Add Sale of PP&E	0
Investing Cashflow (net CAPEX)	(20)
Add additional (seasoned) Equity issue	15
Add new Long-term Debt issue	15
Less Dividends to Equity holders	(5)
Financing Cashflow	25
Total Cashflow	15

The financial analyses of a company is not complete without the Cash Flow Statement. A cash flow statement informs how much cash is entering and exiting the company business during a given period. It indicates if there is enough cash or liquidity to fund the company's operating expenses and pay its debts. If not, the Operating Cashflow would be negative. It complements the other two Statements. Operating Cashflow reporting is harder to manipulate than that of net income – the revenues and costs in the Income Statement are subject to accrual basis in accounting that allows discretion by the company to report revenues (accrued revenues) before the money is received or delay reporting of cost, and thus may provide a better optics of positive net income when the cashflow could be negative.

The total Cashflow of the company is the sum of Operating Cashflow, Investing Cashflow, and Financing Cashflow. Together they reflect the cash situation of the company and if there are enough Cash flows to continue smooth operations of the business. Low or negative cashflows may adversely affect the company's stock prices. However, there are peculiar situations where a company may appear to have negative cash flows due to its decision to expand and incur large capital expenses with the view of huge payoffs in the near future – this may on the contrary augur great stock returns in future.

Often, free cashflow (FCF) is also calculated. This is Operating Cashflow less Investing Cashflow (net CapEx) with after tax interest payment added -- FCF is the cash flow available for the company to repay creditors or pay dividends and interest to investors. FCF per share can augment use of the Earnings per share (EPS) as it can better reflect availability of cash, though the number may be lumpy and uneven as CapEx is lumpy over time. In the above example, FCF is $10 – 20 + 4 (1-2/12) = – 6\ 2/3$, which is generally not a positive signal. The positive total cashflow is due to issues of additional equity and long-term debt.

The 3 key accounting statements are inter-related. Income statement's net income and non-cash charges are key components of the operating cash flows in the Cash Flow statement. Net income in the Income Statement has direct impact on stock prices, hence equity value. This equity value is reflected in mark-to-market valuation of stocks in the Balance Sheet. Changes in asset and liabilities in the Balance Sheet are also reflected in the Cashflows. Total Cashflow is reflected as change in the Cash item in the Balance Sheet. In the rest of this chapter, we show how to employ accounting variables to predict earnings change using logistic regression. Performance measures of any prediction are discussed next.

3.2 Binary Classification Performance Metrics

In statistical hypothesis testing, when a null hypothesis that is true is rejected (not accepted), it is called a Type 1 error. When a null hypothesis that is false is accepted (not rejected), it is called a Type 2 error.

In a binary classification problem, there are two classes. Suppose we define Type A to be the positive class and Type B to be the negative class. The definition is subjective. For example, a company that defaults (within a specific future horizon) may be Type A and one that does not is Type B. A true positive (TP) is an outcome whereby the model correctly predicts the case/subject belongs to the positive class, i.e., correctly predicting a firm will default. A true negative (TN) is an outcome whereby the model correctly predicts the case/subject belongs to the negative class, i.e., correctly predicting a firm will not default.

A false negative (FN) is an outcome where the model incorrectly predicts that the case/subject belongs to the negative class, i.e., incorrectly predicting a firm will not default when it belongs to the positive class. This may be called a prediction Type 1 error if we associate the positive class with the classical idea of null hypothesis. A false positive (FP) is an outcome where the model incorrectly predicts the case/subject belongs to the positive class, i.e., incorrectly predicting a firm will default, when it does not. This may be called a prediction Type 2 error. However, it may be redundant to invoke use of Type 1, 2 errors in the prediction context.

The confusion matrix refers to the matrix containing the numbers of cases (values) in each of the 4 situations after prediction, viz.,

		Predicted Values	
		Positive	Negative
Actual Values	Positive	**TP**	**FN**
	Negative	**FP**	**TN**

Suppose there are a total of 1000 cases/firms in the cross-sectional sample during a specific time-period. Number of TP case is 10. This is simply written as TP = 10. FP = 80. FN = 10, and TN = 900. Note that the number of positive cases and number of negative cases are imbalanced, i.e., 20 versus 980.

For imbalanced cases, the accuracy measure, depicting the percent of correct predictions of positive or negative cases, may not be very informative. In this confusion matrix,

Accuracy = (TP + TN) / (TP + FN + FP + TN) = 910/1000 = 91%.

The high accuracy is due to the imbalance. This is because when knowing 980 out of 1000 cases are negative cases, one could use a naïve approach of predicting all cases as negative. This leads to TP = 0 and TN = 980, so accuracy is 98%. Even if someone attempts to predict the 20 positive cases (but getting them wrong), leaving the rest as predicted negatives, we could have TP = 0, FN = 20, FP = 20, and TN = 960. Accuracy would still be a high 96%. Naïve prediction of negative cases or trusting high accuracy in imbalanced data prediction could be bad if the context is to be able to predict the default (positive) cases and where inability to predict such cases that would default is very costly. Imbalanced data as such can be helped by randomly deleting some negative cases or resampling on positive cases. Besides the misleading accuracy measure in such a situation, other measures as follows are computed and analyzed for the predictive ability of the algorithm.

Another prediction performance measure is precision, which is

Precision = TP / (TP + FP)

that shows what percent of the predicted positive cases turns out to be correctly positive. Here it is 10 / (10 + 80) = 1/9 or about 11.11%. This provides an indication of the performance of the model if predicting positive cases is important. A naïve approach could not be of use when precision is measured.

Another prediction performance measure is recall, which is

Recall = TP / (TP + FN)

that shows the percentage of the actual number of positive cases that were predicted correctly. Here it is 10 / (10 + 10) = 50%. Recall is sometimes also called the true positive rate (TPR) or sensitivity.

Whether precision or recall is a more informative measure depends on whether FP or FN is more of a problem, respectively. Suppose a bank is making a prediction if the firms it lends money to will default or not in the short run, i.e., predicting if the firms are positive cases (default cases). If a prediction is correct, i.e., TP, the bank can either recall the loan before the firm defaults or else hedge the default by buying credit protection from a secondary market such as credit default swaps. If the prediction is incorrect, i.e., FN, which is predicting no default when the firm defaults, then it is costly to the bank due to the default losses. In this situation, a high FN is bad. Thus, a large recall (ratio) TP / (TP + FN) with small FN is an important prediction performance metric for this bank. Recall (or TPR) marks the percentage of the actual number of default firms that were predicted correctly. In this situation, an FP (predicting a firm to default when it does not) may only cost the bank insurance payment fees in insuring the credit risk, which is a smaller sum. Thus, high FP or lower precision in TP / (TP + FP) is not as harmful. In other words, in this situation, higher recall is more important and more informatively significant to the bank than higher precision.

Suppose a company selling a line of products uses advertisements to reach out to its potential customers. The company makes predictions on which segments of the retail market are most likely to respond to the advertisements and to buy its products. Products could be footwear, specialized garments, jewelleries, etc. When a segment/neighbourhood location is predicted as the correct one (positive case), the company puts in advertising expenses on channels linked to the segment. If it is correctly predicted, i.e., TP, then the company will make profits. However, if the prediction in incorrect, i.e., FP, then the company would have sunk in a lot of advertising costs but got little in sales and profit. This situation of low precision or low TP / (TP + FP) due to high FP produces losses and is bad for the company. On the other hand, if the company does not predict that a segment would respond when it would, it is a case of FN. But as the company does not predict a positive case here, it does not spend money on advertisements on this segment, and the direct cost is much less, except for missed opportunities. Thus, low recall TP / (TP + FN) with a large FN is not as harmful or bad for the company. In other words, in this situation, higher precision is more important and more informatively significant to the company than higher recall. For the company, a higher percentage of the correctly predicted positive cases or higher percentage of correctly identified segments for advertising is more critical.

Other important examples include requiring high precision when high FP is a problem such as recommending music of type A (positive class) wrongly to new platform subscribers who dislike music of type A – that may cause business loss due to customer dissatisfaction and churning.[1] Negative cases are situations when platform subscribers are predicted as disliking music of type A, so no recommendations are made.

In some situations, requiring high recall is important when high FN is a problem such as not being able to correctly screen patients for allergies (positive class) to some treatment drugs. Such FN may result in fatal treatments using those medications/drugs.

Sometimes a mixture measure is used such as maximizing the F1-Score which is the harmonic mean of precision and recall, i.e.,

$$\text{F1-Score} = 2/(\text{recall}^{-1} + \text{precision}^{-1}).$$

F1-Score increases with increases in recall and precision. This is useful when both recall and precision need to be high.

The above prediction performance metrics of precision, recall, and F1-score are measured with respect to the positive (P class), or type A. If we define type B as the P case, then the precision, recall, and F1-score will be different numbers with respect to type B. Suppose we define TP with respect to type A (P class) as TP_A and TP with respect to type B (P class) as TP_B. Similarly, we define FP with respect to type A (P class) as FP_A and FP with respect to type B (P class) as FP_B. Then another type of average prediction performance is

$$\text{Micro average precision} = (TP_A + TP_B)/(TP_A+FP_A+TP_B+FP_B).$$

Such "averaged" measures with respect to both types of cases (A and B, or positive and negative) are useful when precision on type A and also precision on type B are both important information. It could be predicted a stock price going up (positive case) or going down (negative case, including not changing), in which situation precision of positive case leading to buy or precision of negative case leading to sell are both important.

If precision, recall, and F1-score with respect to type A as P are $prec_A$, rec_A, and $f1_A$, and precision, recall, and F1-score with respect to type B as P are

[1] Based on history of past subscribers, supervised learning on them may lead to predictive analytics that recommends music types to new subscribers with the associated profiles (features). For existing subscribers with log-in history, unsupervised learning may make further recommendations (such as highly correlated music types) to each subscriber based on the listening history.

$prec_B$, rec_B, and $f1_B$, and suppose there are n_A number of cases/subjects that are of type A and n_B number of cases/subjects that are of type B, then

Macro average of precision = $(prec_A + prec_B)/2$
Macro average of recall = $(rec_A + rec_B)/2$
Macro average of F1-score = $(f1_A + f1_B)/2$

that are based on simple arithmetic averages. The weighted averages of these measures are respectively $n_A/(n_A+n_B) \times prec_A + n_B/(n_A+n_B) \times prec_B$, $n_A/(n_A+n_B) \times rec_A + n_B/(n_A+n_B) \times rec_B$, and $n_A/(n_A+n_B) \times f1_A + n_B/(n_A+n_B) \times f1_B$.

Another measure is specificity or true negative rate (TNR)

Specificity = TN / (TN + FP)

This is the percentage of the actual number of negative cases that were predicted correctly. The false positive rate

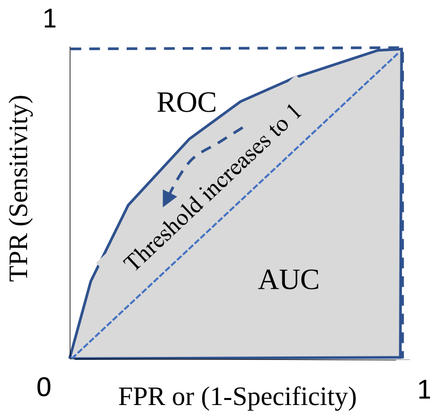
FPR = 1 – Specificity = FP / (TN + FP).

FPR is the percentage of the actual number of negative cases that were predicted as FP, i.e., wrongly predicted. Note that TPR and FPR both lie between 0 and 1.

In a binary classification, the model typically will output a predicted probability of positive class (where the positive class is appropriately defined). It is typically the case to use a hyperparameter threshold of 0.5 whereby if the predicted probability is > (<=) 0.5, then the case is assigned as a predicted positive (negative) case. The threshold can however be adjusted to improve performance if the types are highly imbalanced. For example, in the 1000 firms' study, given the predicted probabilities by the algorithm, precision can be raised if the threshold for predicting positive is raised so that FP, which is a relatively large number to TP, is decreased as less cases are then predicted as positive due to the increased threshold. Less of the negative cases are incorrectly predicted as positive – hence smaller FP. Thus, changing threshold is an interesting way to find more information about the predictive ability of the algorithm.

One of the most important more comprehensive performance metrics in a classification model's performance is the AUC (Area Under The Curve) of the ROC (Receiver Operating Characteristics) curve. It may also be called the AUROC (Area Under the Receiver Operating Characteristics).

The ROC is a curve that plots the TPR against FPR at various threshold values. This is shown below.



Given the predicted probabilities, as the threshold is increased toward 1, less cases are predicted as P, hence TP decreases toward zero, and FN increases (since TP + FN add up to a fixed number of actual positive cases). Thus, TPR or sensitivity TP / (TP + FN) goes toward 0. Similarly, as the threshold is increased toward 1, more cases are predicted as N, hence TN increases, and FP decreases (since TN + FP add up to a fixed number of actual negative cases). Thus, FPR also goes toward 0.

Conversely, as the threshold approaches zero, more cases are predicted as P, hence TP increases, and FN decreases (since TP + FN add up to a fixed number of actual positive cases). Thus, TPR or sensitivity TP / (TP + FN) increases toward 1. Similarly, as the threshold is decreased toward 0, less cases are predicted as N, hence TN decreases, and FP increases (since TN + FP add up to a fixed number of actual negative cases). Thus, FPR also goes toward 1.

In the diagram, any point on the diagonal dotted line represents a situation when TPR = FPR. Points of ROC curve above this diagonal line represent the situations TPR > FPR, where the proportion of correctly predicted positive cases of all positive cases is larger than the proportion of incorrectly predicted positive cases of all negative class.

When TPR > FPR, solving TP/(TP+FN) > FP/(TN+FP) gives TP × TN > FP × FN. Here there is a high chance that the model can distinguish the positive cases from the negative cases because the model can detect more numbers of TPs and TNs (in product term) than FPs and FNs. On the diagonal line when TP × TN = FP × FN, the model is not able to distinguish between positive and negative cases, i.e., akin to predicting the classes randomly.

Across all thresholds, higher AUC (where AUC > 0.5), or higher averages of TP × TN than averages of FP × FN, the better is the algorithm's ability to distinguish between positive cases as positive and negative classes as negative. For competing predicting models, the one with a higher AUC is the better model that can predict. AUC measures predictive performance in a more comprehensive way by considering all thresholds and not just the usual threshold of 0.5. When AUC is below 0.5 or close to 0.5, then the algorithm is not effective in prediction. It is possible for ROC curve to be partly below and partly above the diagonal. If this flip occurs around the threshold of typical 0.5, then the algorithm is also not effective.

In a multi-class classification, the computations will be based on TP, FN, FP, and TN for each of the category/class. Take for example a prediction of one of 3 classes, viz., whether a business class airline passenger prefers Eastern (E), Western (W), or Mediterranean (M) food on board, in order to make optimal food meals stock-up before flight. The numbers of (actual preference j, predicted preference k) cases are indicated in the (j,k) box below.

		Predicted Preference		
		E	W	M
Actual Preference	E	X_{EE}	X_{EW}	X_{EM}
	W	X_{WE}	X_{WW}	X_{WM}
	M	X_{ME}	X_{MW}	X_{MM}

The TP, FN, FP, and TN for the E class (those who prefer E – the positive cases, those who do not prefer E are the negative cases) are X_{EE}, $(X_{EW} + X_{EM})$, $(X_{WE} + X_{ME})$, and $(X_{WW} + X_{WM} + X_{MW} + X_{MM})$. The TP, FN, FP, and TN for the W class (those who prefer W – the positive cases, those who do not prefer W are the negative cases) are: X_{WW}, $(X_{WE} + X_{WM})$, $(X_{EW} + X_{MW})$, and $(X_{EE} + X_{EM} + X_{ME} + X_{MM})$. The TP, FN, FP, and TN for the M class (those who prefer M – the positive cases, those who do not prefer M are the negative cases) are: X_{MM}, $(X_{ME} + X_{MW})$, $(X_{EM} + X_{WM})$, and $(X_{EE} + X_{EW} + X_{WE} + X_{WW})$. Thus, we can compute an AUC for each class, and a clearly better performing model would produce higher AUC in as many classes.

3.3 Logistic Regression

Logistic regression or logit model can be used to estimate the probabilities of events, given attributes or features that can possibly determine such probabilities. If an event is a binary classification of either type A or type B (yes or no; default or no default; contagious or not contagious; criminal or not criminal; exit or no exit, pass or fail, and so on), we can define the ith case as type A with label as $Y_i = 1$. Without loss of generality, if the ith case is not type A, then it is type B with label $Y_i = 0$.

Then the logistic/logit regression model can be applied, viz.

$E(Y_i|X_i) = \text{Prob}(Y_i = 1|X_i) = \exp(\beta^T X_i)/[1+ \exp(\beta^T X_i)] = 1/[1+ \exp(-\beta^T X_i)]$

where X_i is a vector of feature variables associated with case/subject i. β is vector of coefficients on X_i.

For example, in studying many firms during a fiscal year, firm i could see an earnings increase ($Y_i = 1$) while another firm j could see an earnings decrease ($Y_j = 0$). The latter includes case of unchanged earnings. Now $\exp(\beta^T X_i)/[1+ \exp(\beta^T X_i)]$ is the logistic (cumulative) distribution function that lies between 0 and 1. This function is also the probability of a binomial distribution with binary outcome 1, and with probability $[1 - \exp(\beta^T X_i)/[1+ \exp(\beta^T X_i)]]$ for outcome 0. Then the likelihood function of an observed training sample of N cases (N_1 cases of outcome 1 and $N - N_1$ cases of outcome 0) is

$$L = \prod_{i=1}^{N} [F(\beta^T X_i)]^{Y_i} [1 - F(\beta^T X_i)]^{1-Y_i}$$

where $F(\beta^T X_i) = \text{Prob}(Y_i = 1 \mid X_i)$, and $\Sigma_i Y_i = N_1$. X_i and Y_i are from the training set. The loss function (same as negative of log likelihood function in classical statistics) is

$$loss = -\sum_{i=1}^{N} \left(Y_i \, logF + (1 - Y_i) \, log(1 - F) \right)$$

which is to be minimized. This is done via a numerical iterative procedure to find the optimal parameters $\hat{\beta}$ that is then used to evaluate the predicted probability of $\hat{Y}_k$ based on X_k from the test set cases k. In sklearn LogisticRegression app, the options can be set to (loss function) of multi_class= ‘multinomial’. This includes the binary, hence binomial situation. Solver can be by the ‘newton-cg’ method. Apps metrics, classification_report, accuracy score, confusion_matrix, etc. from sklearn can then be applied to find the various prediction performance results. Also, it should be noted that by default sklearn LogisticRegression applies L2 regularization in the estimation. For sklearn LogisticRegression, when the estimated probability of

$$F\left(\hat{\beta}^T X_i\right) = \text{Prob}(Y_i = 1 \mid X_i)$$

exceeds 0.5, then the predicted class is considered as $Y_i = 1$. Otherwise, it is $Y_i = 0$.

However, just as in linear regression, logistic regression faces some problems when there is large multicollinearity amongst the features – even failure to converge to the optimal solution. Large multi-collinearity amongst features (or regressors in classical terminology) can be detected using ‘variance_inflation_factor’ app in statsmodels. Each feature is regressed, via linear regression, on all the other features, and the variance-inflation-factor (VIF) = $1/(1-R^2)$ can be computed where R^2 is the coefficient of determination of the linear regression. If the $R^2 > 0.9$ (very high collinearity), then VIF > 10.

With multi-collinearity, one problem is that prediction (or forecasting in classical statistics) can become unstable and can produce bad results as the coefficients or weight vector on features for prediction can be unstable. Secondly, since these weight vectors β can change in unstable ways, it is also not accurate to try to find out which features X_i are more important, i.e., have bigger weights in the prediction.

To more conveniently assess the logistic regression estimation results and check on the statistical significance of the various feature impact (eliminating those that are multi-collinear and also which may have negligible impact), logit model regression in statsmodels is used to complement the prediction exercise, besides using sklearn LogisticRegression.

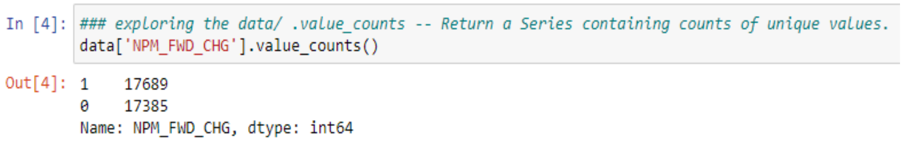
3.4 Worked Example – Data

U.S. listed company annual (end December) accounting ratios from 1971 till 2015 are obtained from subscribed WRDS. The data set is co_fin_ratios.csv and the interactive Python Notebook is Chapter3-1.ipynb. Some companies appeared in the data set after 1971 while some showed data for only a few years. Some companies whose accounting ratios were not complete were not included.

The data consist of 35,074 rows of annual company data and 47 columns of accounting ratios. Some rows are different annual data of the same company. The last column 'NPM_FWD_CHG' (net profit margin forward change) is not an accounting ratio. It is 1 if the net profit margin increased in the following year. It is 0 otherwise. Note that this forward variable is of course not observed at the time-stamp of the features, but it is used for training and for testing the predictive abilities of the model using the features.

The purpose of the exercise is to harness the companies' accounting ratios each year to see if they can help predict if the following year's net profit margin would increase or not.

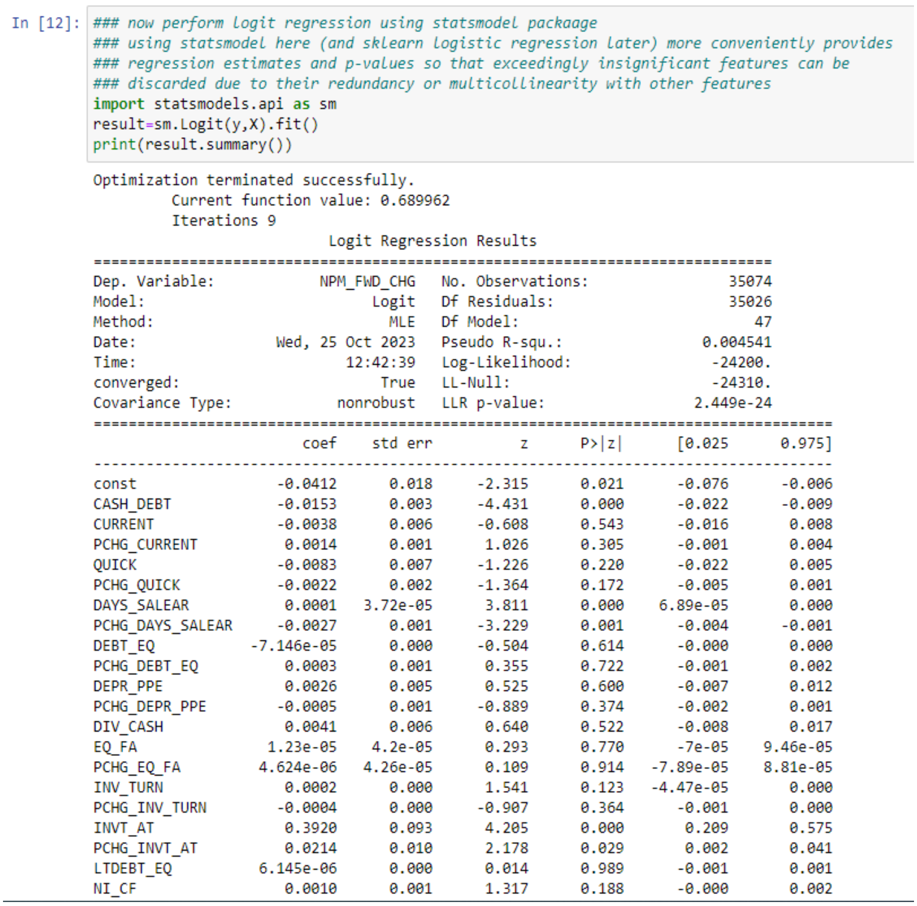
Code line [4] shows that in the data set there are 17,689 annual cases of NPM increase and 17,385 cases no NPM increase. Thus, the data set is balanced in the two categories.



```
In [4]: ### exploring the data/ .value_counts -- Return a Series containing counts of unique values.
        data['NPM_FWD_CHG'].value_counts()

Out[4]: 1    17689
        0    17385
        Name: NPM_FWD_CHG, dtype: int64
```

The columns with objects such as 'datadate', 'fyear', 'tic', and 'Company Name' are dropped, leaving 47 accounting ratios to form the feature space.

Logit regression using statsmodel is first performed on the entire data set with 'NPM_FWD_CHG' as the target variable or label. This is to have an initial idea of how the various features impact or affect the label. Using statsmodel here (and sklearn logistic regression later) more conveniently provides regression estimates and their p-values so that exceedingly insignificant features may be discarded due to their redundancy or multicollinearity with other features. Note that in linear regression, multi-collinear features often have estimated coefficients that are not statistically significantly different from zero due to the large standard errors of the coefficient estimators. See code line [12]. Only a part of the output table is shown below to economize on space.



```
In [12]: ### now perform logit regression using statsmodel packaage
         ### using statsmodel here (and sklearn logistic regression later) more conveniently provides
         ### regression estimates and p-values so that exceedingly insignificant features can be
         ### discarded due to their redundancy or multicollinearity with other features
         import statsmodels.api as sm
         result=sm.Logit(y,X).fit()
         print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.689962
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:            NPM_FWD_CHG   No. Observations:                35074
Model:                          Logit   Df Residuals:                    35026
Method:                           MLE   Df Model:                           47
Date:                Wed, 25 Oct 2023   Pseudo R-squ.:                0.004541
Time:                        12:42:39   Log-Likelihood:                -24200.
converged:                       True   LL-Null:                       -24310.
Covariance Type:            nonrobust   LLR p-value:                 2.449e-24
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -0.0412      0.018     -2.315      0.021      -0.076      -0.006
CASH_DEBT            -0.0153      0.003     -4.431      0.000      -0.022      -0.009
CURRENT              -0.0038      0.006     -0.608      0.543      -0.016       0.008
PCHG_CURRENT          0.0014      0.001      1.026      0.305      -0.001       0.004
QUICK                -0.0083      0.007     -1.226      0.220      -0.022       0.005
PCHG_QUICK           -0.0022      0.002     -1.364      0.172      -0.005       0.001
DAYS_SALEAR           0.0001   3.72e-05      3.811      0.000    6.89e-05       0.000
PCHG_DAYS_SALEAR     -0.0027      0.001     -3.229      0.001      -0.004      -0.001
DEBT_EQ           -7.146e-05      0.000     -0.504      0.614      -0.000       0.000
PCHG_DEBT_EQ          0.0003      0.001      0.355      0.722      -0.001       0.002
DEPR_PPE              0.0026      0.005      0.525      0.600      -0.007       0.012
PCHG_DEPR_PPE        -0.0005      0.001     -0.889      0.374      -0.002       0.001
DIV_CASH              0.0041      0.006      0.640      0.522      -0.008       0.017
EQ_FA               1.23e-05    4.2e-05      0.293      0.770      -7e-05    9.46e-05
PCHG_EQ_FA         4.624e-06   4.26e-05      0.109      0.914   -7.89e-05    8.81e-05
INV_TURN              0.0002      0.000      1.541      0.123   -4.47e-05       0.000
PCHG_INV_TURN        -0.0004      0.000     -0.907      0.364      -0.001       0.000
INVT_AT               0.3920      0.093      4.205      0.000       0.209       0.575
PCHG_INVT_AT          0.0214      0.010      2.178      0.029       0.002       0.041
LTDEBT_EQ          6.145e-06      0.000      0.014      0.989      -0.001       0.001
NI_CF                 0.0010      0.001      1.317      0.188      -0.000       0.002
```

....

....

The logistic regression shows that estimated probability of Prob($Y_i = 1 \mid X_i$) is found via maximization of likelihood function or minimization of loss function in estimating $\hat{\beta}$. The estimated probability is also

$$F(\hat{\beta}^T X_i) = 1/[1+ \exp(-\hat{\beta}^T X_i)]$$

where X_i is the vector of 47 features for a particular company in a particular year, and $\hat{\beta}$ is the vector of 47 estimated coefficients as shown in the Logit Regression Results Table. In other words,

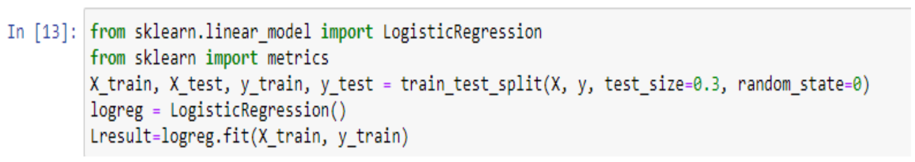
$$\hat{\beta}^T X_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2} + \cdots + \hat{\beta}_k X_{ik} + \cdots + \hat{\beta}_{47} X_{i47}$$

where the kth element of $\hat{\beta}$ is scalar $\hat{\beta}_k$, and its associated feature of the ith case in the logistic regression is X_{ik}. If $\hat{\beta}_k > 0$, then when X increases, $F(\hat{\beta}^T X_i)$ or estimated Prob ($Y_i = 1|X_i$) increases. The latter implies that there is more likelihood that earnings would increase. On the contrary, if $\hat{\beta}_k \le 0$, then when X increases, $F(\hat{\beta}^T X_i)$ or estimated Prob ($Y_i = 1|X_i$) decreases. The latter implies that there is more likelihood that earnings would stagnate or decrease.

The results in the Table indicate that the coefficients on DAYS_SALEAR, INVT_AT, PCHG_INVT_AT, PCHG_OPMBD, SALE_AT, SALE_NWC are significantly positive at p-value < 5%, and increase in their associated feature values would increase the likelihood of positive NPM change the following year. The coefficients on CASH_DEBT, PCHG_DAYS_SALEAR, SALE_FA, TIE, PCHG_INVT are significantly negative at p-value < 5%, and increase in their associated feature values would reduce the likelihood of positive NPM change the following year.

Some of these results are easier to explain. For example, increase in PCHG_OPMBD or percentage change in operating profit margin before depreciation often augurs improvement in NPM. Increase in sales to asset (SALE_AT) and increase in sales to net working capital (SALE_NWC) signals asset and funding efficiency and future improved NPM. Increase in percentage of days' sales in accounts receivable (PCHG_DAYS_SALEAR) which is the number of days in a year divided by the accounts receivable turnover ratio or the average time to collect cash on receivables, implies weaker collection policy and hence lower future NPM expectation. Increase in sale of fixed asset SALE_FA is often linked with financial troubles (other than strategic business realignment) at the firm, and hence would imply poor NPM the following year. Percentage increase in investments (PCHG_INVT) would typically mean short-run drain on NPM but perhaps with long run increases when the investments become successful. Some of the other accounting ratio such as times interest earned (TIE) impacts are harder to explain with the diverse set of firms.

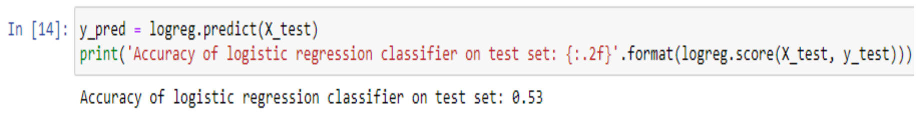
Next the data is randomly split into 70% training set (24,551 rows), and 30% test set (10,523 rows).



```
In [13]: from sklearn.linear_model import LogisticRegression
         from sklearn import metrics
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
         logreg = LogisticRegression()
         Lresult=logreg.fit(X_train, y_train)
```

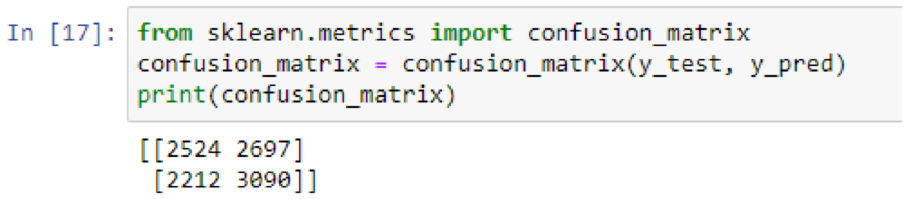
sklearn LogisticRegression is then applied to the training set fitting and then the prediction of the test set. One difference between the outputs from sklearn Logistic Regression and statsmodel logit regression is that the former adds L2 regularization in the estimation, so their results may differ by a little.

The accuracy of the test set prediction is shown to be about 53%. See code line [14].



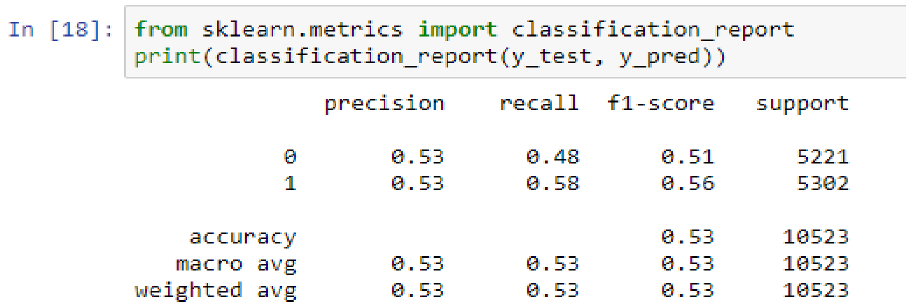
```
In [14]: y_pred = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

         Accuracy of logistic regression classifier on test set: 0.53
```

The confusion matrix and the various prediction performance metrics are shown in [17], [18]. Note for confusion matrix in [17] here in sklearn, the number of true positives (‘NPM_FWD_CHG’ = 0, and predicted value 0) is 2524. The number of true negatives (‘NPM_FWD_CHG’ = 1, and predicted value 1) is 3090. Hence accuracy is (2,524+3,090)/10,523 = 53.34%.



```
In [17]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred)
         print(confusion_matrix)

         [[2524 2697]
          [2212 3090]]
```

In other words, the first line of the confusion matrix denotes the case ‘NPM_FWD_CHG’ = 0 as true positive, or the case that next net profit margin does not increase. The second line denotes the case ‘NPM_FWD_CHG’ = 1 as true negative, or the case that next net profit margin increases.

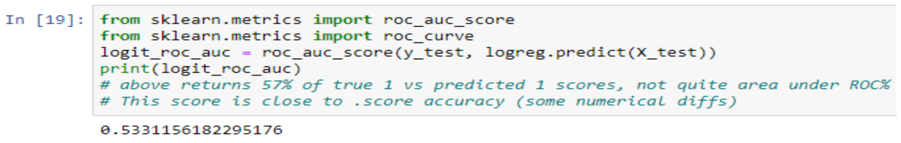


```
In [18]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                  0       0.53      0.48      0.51      5221
                  1       0.53      0.58      0.56      5302

           accuracy                           0.53     10523
          macro avg       0.53      0.53      0.53     10523
       weighted avg       0.53      0.53      0.53     10523
```

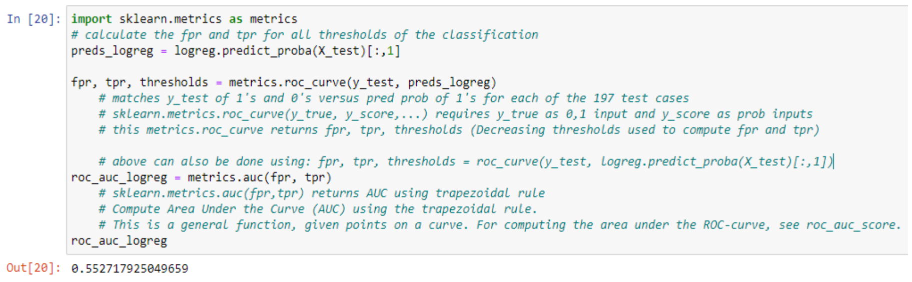
The tables show the following for precision. For the 5221 positive cases of no increase in NPM, the percentage of the predicted positive cases that turns out to be correctly positive is 2524/(2524+2212) = 53.29%. For the 5302 negative cases of increase in NPM, the percentage of the predicted negative cases that turns out to be correctly negative is 3090/(2697+3090) = 53.39%.

In terms of recall, the tables show that for the 5221 positive cases of no increase in NPM, a very low percentage of 48% is correctly predicted as being positive. But for the 5302 negative cases of increase in NPM, a higher 58% is correctly predicted as being negative.

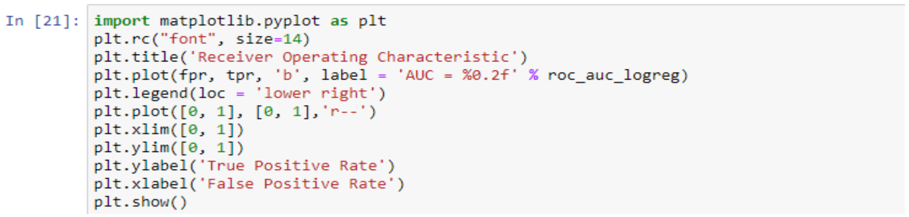
It may be argued that ability to predict negative cases (increase in NPM) may be more important. However, it is much better to be able to predict both negative and positive cases. The AUC of the ROC curve is shown in the output of code lines [19], [20], and [21]. AUC is 55.27%.



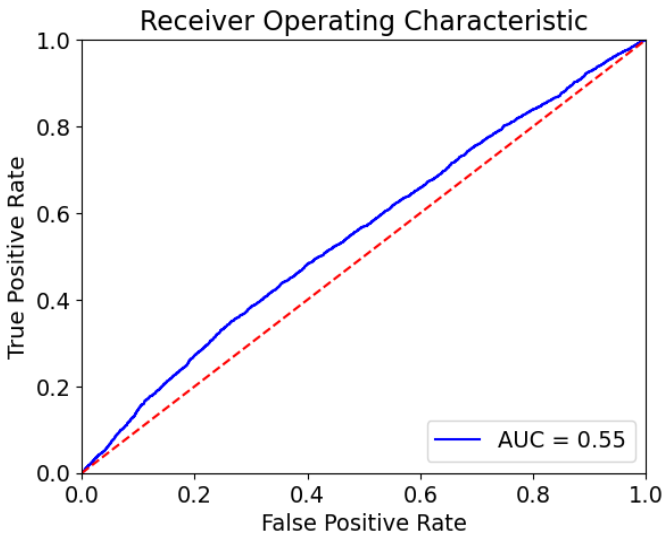
```
In [19]: from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve
         logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
         print(logit_roc_auc)
         # above returns 57% of true 1 vs predicted 1 scores, not quite area under ROC%
         # This score is close to .score accuracy (some numerical diffs)

         0.5331156182295176
```



```
In [20]: import sklearn.metrics as metrics
         # calculate the fpr and tpr for all thresholds of the classification
         preds_logreg = logreg.predict_proba(X_test)[:,1]

         fpr, tpr, thresholds = metrics.roc_curve(y_test, preds_logreg)
             # matches y_test of 1's and 0's versus pred prob of 1's for each of the 197 test cases
             # sklearn.metrics.roc_curve(y_true, y_score,...) requires y_true as 0,1 input and y_score as prob inputs
             # this metrics.roc_curve returns fpr, tpr, thresholds (Decreasing thresholds used to compute fpr and tpr)

             # above can also be done using: fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
         roc_auc_logreg = metrics.auc(fpr, tpr)
             # sklearn.metrics.auc(fpr,tpr) returns AUC using trapezoidal rule
             # Compute Area Under the Curve (AUC) using the trapezoidal rule.
             # This is a general function, given points on a curve. For computing the area under the ROC-curve, see roc_auc_score.
         roc_auc_logreg

Out[20]: 0.552717925049659
```



```
In [21]: import matplotlib.pyplot as plt
         plt.rc("font", size=14)
         plt.title('Receiver Operating Characteristic')
         plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc_logreg)
         plt.legend(loc = 'lower right')
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0, 1])
         plt.ylim([0, 1])
         plt.ylabel('True Positive Rate')
         plt.xlabel('False Positive Rate')
         plt.show()
```



3.5 Dimension Reduction and Principal Component Analysis

If we were able to use all 47 features in the logistic regression, it may be overfitting. One way to reduce the dimensionality and overfitting is to prune the number of features. Chen et. Al. (2022) and other showed how careful selection of the features (sometimes using stepwise approach by removing insignificant features that do not affect R^2) could be important.

A common dimension reduction technique is principal component analysis (PCA) whereby the 47 features/variables could be collapsed (reduced) to a smaller number of variables. Each of the resulting variable is a principal component (PC) that is a linear combination of the 47 variables. One nice feature of PCA is that each PC is orthogonal to another (zero correlation) and with enough number of PCs, their total variance could match close to if not equal to the total variance of all the individual features. The orthogonality would alleviate the multicollinearity issue in regression problems, including the logistic regression.

A brief theory of PCA can be explained as follows. Suppose the N features are random variables $x_1, x_2, x_3, \ldots., x_N$. Now let $X_{N\times1} = (x_1, x_2, x_3, \ldots., x_N)^T$ be a random vector. Let the mean of $X_{N\times1}$ be $\mu_{N\times1}$. Let the covariance matrix of X be $\Sigma_{N\times N}$. Let $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_N$ be the N eigenvalues and $v_1, v_2, v_3, \ldots, v_N$ be the N corresponding eigenvectors (dimension $N \times 1$) of Σ such that $\Sigma v_j = \lambda_j v_j$ for any j=1,2,…,N. It can be shown that the eigenvalues of Σ, a real symmetric positive-definite matrix, are all (real) positive, though some may not be unique.

The solutions to the eigenvalues and eigenvectors are constrained by normalized orthogonal eigenvectors so that $v_j^T v_j = 1$ for all j, and $v_j^T v_k = 0$ for $j \neq k$. Let $\Lambda_{N\times N} = \text{diag}(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_N)$, a diagonal matrix. Without loss of generality, let the diagonal elements be arranged in descending order so that the left-top element λ_1 is the largest eigenvalue and the right-bottom element λ_N is the smallest eigenvalue.

Let $\Gamma_{N\times N} = (v_1, v_2, v_3, \ldots, v_N)$ such that $\Gamma^T\Gamma = I_{N\times N}$, the N-dimension identity matrix. Now let random variable $Y_j = v_j^T(X - \mu)$ which has a 1×1 dimension. This linear combination of the N detrended random features is called the j^{th} scalar principal component (PC) of X. There are N such PCs since there are j number of eigenvectors v_j. Let $Y_{N\times1} = (Y_1, Y_2, Y_3, \ldots, Y_N)^T$ be vector of the Y_j's.

Properties of the PC vector Y can be seen as follows. Given Γ, its expectation or mean, $E(Y) = \Gamma^T E(X - \mu) = 0$. Variance of vector Y is var(Y) or $[\text{cov}(Y_i,Y_j)]_{N\times N} = E(YY^T) = E(\Gamma^T(X - \mu)(X - \mu)^T\Gamma) = \Gamma^T\Sigma\Gamma$ given Γ. But by the definition of eigenvalues and eigenvectors, $\Sigma\Gamma = \Gamma\Lambda$. Hence,

$$\text{var}(Y) = E(YY^T) = \Gamma^T\Sigma\Gamma = \Gamma^T\Gamma\Lambda = \Lambda.$$

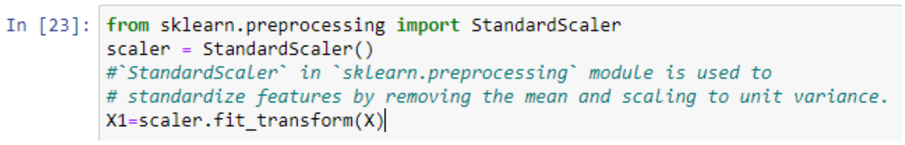
Clearly, the variance of the jth principal component Y_j is the jth diagonal element of $\Lambda_{N\times N}$ or λ_j. Different principal components are also uncorrelated with each other. If the PCs are normally distributed, then they are independent.

From the way we construct Λ, the first PC, Y_1, has the largest variance followed by the variance of the second PC Y_2, and so on.

Since $\Gamma^T\Gamma = \Gamma\Gamma^T = I_{N\times N}$, trace of Λ or $\text{tr}(\Lambda) = \text{tr}(\Gamma^T\Sigma\Gamma) = \text{tr}(\Sigma\Gamma\Gamma^T) = \text{tr}(\Sigma) = \sum_{j=1}^{N} \text{var}(x_j)$ or the total variance of the features. Note that the trace operation is invariant under circular shifts shown above. Therefore, the sum of all the variances of the PCs is equal to $\text{tr}(\Lambda)$ which is also the total variance of the features. The contribution of each PC variance is also called the explained variance (information explained using the eigenvectors) and is often expressed as a ratio to the total variance, i.e., $\text{var}(Y_j)/\sum_{j=1}^{N}\text{var}(x_j)$. The sum of these ratios across all eigenvalues equal to one.

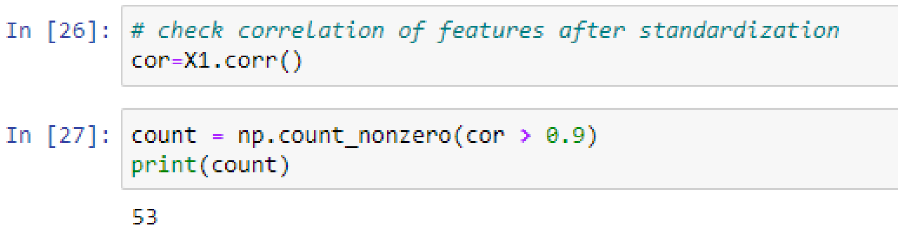
Since we do not know the exact Σ nor mean μ of the vector X, they must be estimated using the time series of the features, assuming the features are stationary. Estimate of μ is $\hat{\mu} = (\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_N)$ where $\hat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} x_{jt}$. The jth random feature takes realized time series values $x_{j1}, x_{j2}, \ldots, x_{jT}$. T is the sample size. Estimate of Σ is $\hat{\Sigma} = \left[\widehat{\text{cov}}(x_i, x_j)\right]$ where $\widehat{\text{cov}}(x_i, x_j) = \frac{1}{T}\sum_{t=1}^{T}(x_{it} - \hat{\mu}_i)(x_{jt} - \hat{\mu}_j)$. Any computed eigenvalue $\hat{\lambda}_j$ and eigenvector $\hat{v}_j$ would be based on sample estimates $\hat{\Sigma}$ and $\hat{\mu}$. Solution of $\hat{\lambda}_j$ is via a polynomial equation involving determinant $\det(\hat{\Sigma} - \hat{\lambda}_j I) = 0$. Eigenvector $\hat{v}_j$ is then solved in a second step. Once the eigenvectors are estimated, the jth PC based on realized features data is $Y_j = \hat{v}_j^T(X^* - \hat{\mu})$ where X^* is the realized values of X.

We will perform a principal component analysis and possible dimension reduction of the feature space. Dimension reduction occurs if we choose only a subset of the PCs incorporating PCs with the largest variances. First, we standardize the features. See code line [23].



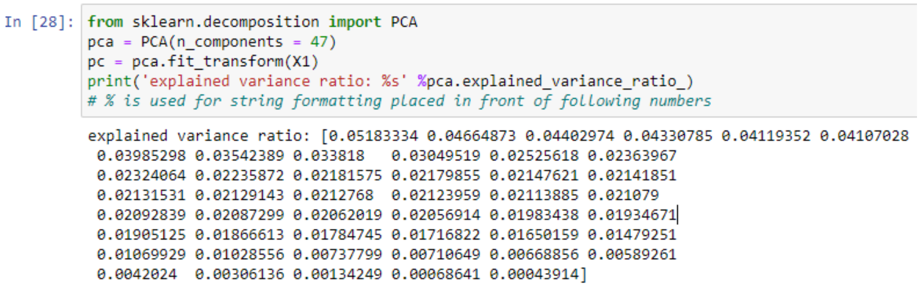
```
In [23]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         #`StandardScaler` in `sklearn.preprocessing` module is used to
         # standardize features by removing the mean and scaling to unit variance.
         X1=scaler.fit_transform(X)
```

Then we check the degree of multicollinearity that may affect the logistic regression. Code lines [26], [27] show that there are 53 correlations of pairs of features in X that have correlation exceeding 0.9.



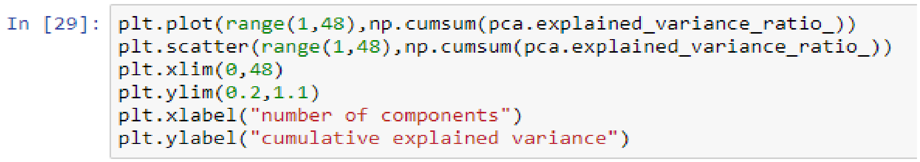
```
In [26]: # check correlation of features after standardization
         cor=X1.corr()

In [27]: count = np.count_nonzero(cor > 0.9)
         print(count)

         53
```

In code line [28], all 47 PCs are extracted after the features are normalized. The normalization or standardization is important as it then assigns "equal weights" to all features, so they are on the same scale. This is particularly important in feature space where some features with large measures may swam those with smaller measures and produce distorted combined features.

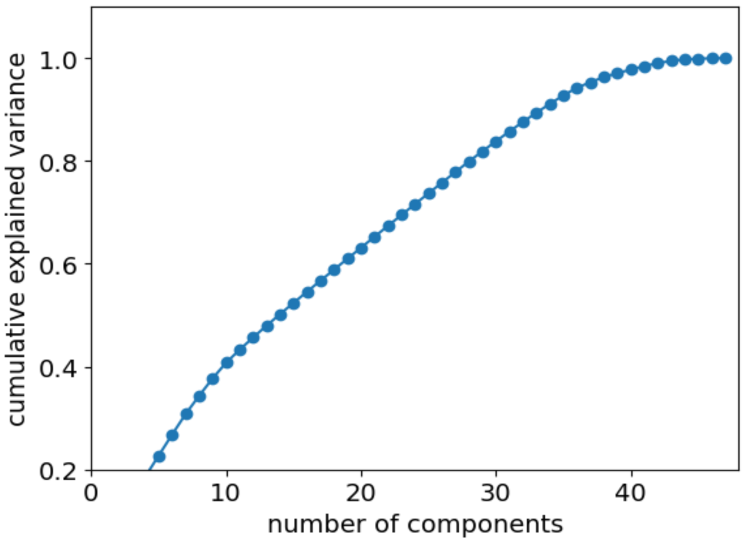


```
In [28]: from sklearn.decomposition import PCA
         pca = PCA(n_components = 47)
         pc = pca.fit_transform(X1)
         print('explained variance ratio: %s' %pca.explained_variance_ratio_)
         # % is used for string formatting placed in front of following numbers

         explained variance ratio: [0.05183334 0.04664873 0.04402974 0.04330785 0.04119352 0.04107028
          0.03985298 0.03542389 0.033818   0.03049519 0.02525618 0.02363967
          0.02324064 0.02235872 0.02181575 0.02179855 0.02147621 0.02141851
          0.02131531 0.02129143 0.0212768  0.02123959 0.02113885 0.021079
          0.02092839 0.02087299 0.02062019 0.02056914 0.01983438 0.01934671
          0.01905125 0.01866613 0.01784745 0.01716822 0.01650159 0.01479251
          0.01069929 0.01028556 0.00737799 0.00710649 0.00668856 0.00589261
          0.0042024  0.00306136 0.00134249 0.00068641 0.00043914]
```

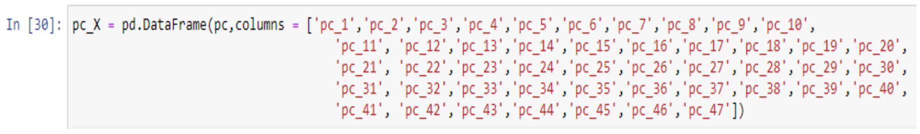
The 'pca.explained_variance_ratio' is the ratio of each eigenvalue to the sum of all eigenvalues which represents the total sample variances of all the feature variables. The cumulative contribution of the explained variances ratios is shown in [29].



```
In [29]: plt.plot(range(1,48),np.cumsum(pca.explained_variance_ratio_))
         plt.scatter(range(1,48),np.cumsum(pca.explained_variance_ratio_))
         plt.xlim(0,48)
         plt.ylim(0.2,1.1)
         plt.xlabel("number of components")
         plt.ylabel("cumulative explained variance")
```



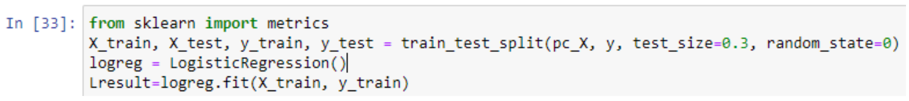
The PCs are used to form the new 47 features in pcX. See [30].



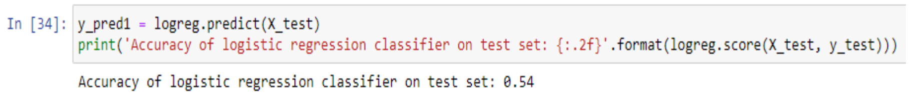
```
In [30]: pc_X = pd.DataFrame(pc,columns = ['pc_1','pc_2','pc_3','pc_4','pc_5','pc_6','pc_7','pc_8','pc_9','pc_10',
                                          'pc_11', 'pc_12','pc_13','pc_14','pc_15','pc_16','pc_17','pc_18','pc_19','pc_20',
                                          'pc_21', 'pc_22','pc_23','pc_24','pc_25','pc_26','pc_27','pc_28','pc_29','pc_30',
                                          'pc_31', 'pc_32','pc_33','pc_34','pc_35','pc_36','pc_37','pc_38','pc_39','pc_40',
                                          'pc_41', 'pc_42','pc_43','pc_44','pc_45','pc_46','pc_47'])
```

We can experiment with a smaller number of PCs and check their performances in the predictions. Here we utilize all PCs to perform the same logistic regresson and predictions. One problem with regression using the principal components is that it is now difficult to interpret the impact of each PC which is a linear combination of many accounting ratios.

Then accuracy measurement of the test cases is performed in code lines [33], [34].



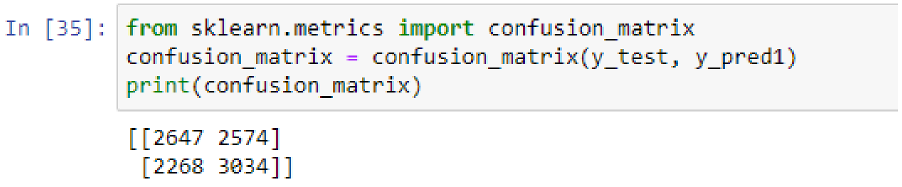
```
In [33]: from sklearn import metrics
         X_train, X_test, y_train, y_test = train_test_split(pc_X, y, test_size=0.3, random_state=0)
         logreg = LogisticRegression()
         Lresult=logreg.fit(X_train, y_train)
```



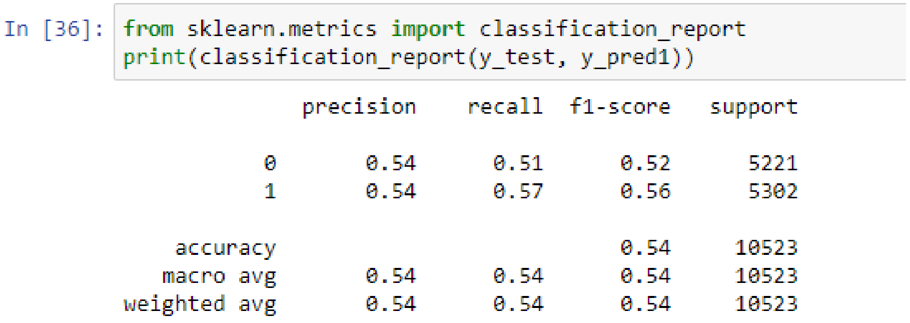
```
In [34]: y_pred1 = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

         Accuracy of logistic regression classifier on test set: 0.54
```

There is an improvement of about 1% relative to the 53% in the feature space before PCA transformations.

The confusion matrix and classification reports are shown via code lines [35], [36].

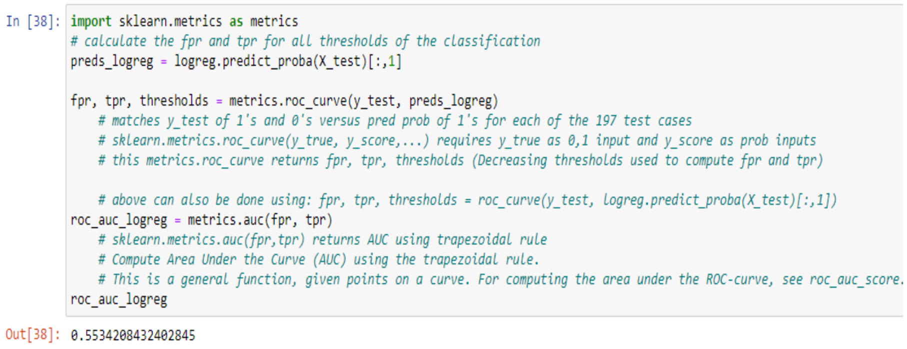


```
In [35]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred1)
         print(confusion_matrix)

         [[2647 2574]
          [2268 3034]]
```

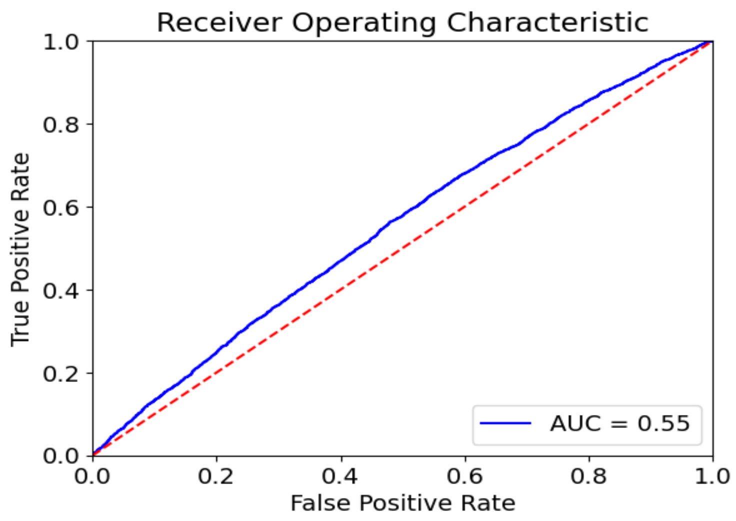


```
In [36]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred1))

                       precision    recall  f1-score   support

                    0       0.54      0.51      0.52      5221
                    1       0.54      0.57      0.56      5302

             accuracy                           0.54     10523
            macro avg       0.54      0.54      0.54     10523
         weighted avg       0.54      0.54      0.54     10523
```

The area under ROC is now 55.34%, also about 1% increase from the original feature space. See code lines [38], [39].



```
In [38]: import sklearn.metrics as metrics
         # calculate the fpr and tpr for all thresholds of the classification
         preds_logreg = logreg.predict_proba(X_test)[:,1]

         fpr, tpr, thresholds = metrics.roc_curve(y_test, preds_logreg)
             # matches y_test of 1's and 0's versus pred prob of 1's for each of the 197 test cases
             # sklearn.metrics.roc_curve(y_true, y_score,...) requires y_true as 0,1 input and y_score as prob inputs
             # this metrics.roc_curve returns fpr, tpr, thresholds (Decreasing thresholds used to compute fpr and tpr)

             # above can also be done using: fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
         roc_auc_logreg = metrics.auc(fpr, tpr)
             # sklearn.metrics.auc(fpr,tpr) returns AUC using trapezoidal rule
             # Compute Area Under the Curve (AUC) using the trapezoidal rule.
             # This is a general function, given points on a curve. For computing the area under the ROC-curve, see roc_auc_score.
         roc_auc_logreg

Out[38]: 0.5534208432402845
```



It is seen that PCA transformed features in this case (eliminating multi-collinearity) provide marginally better prediction results.There is no guarantee that PCA transformation will produce significantly better results in prediction even when multi-collinearity is reduced or eliminated. However, it is correct to say that reducing multi-collinearity (with or without PCA) generally should improve statistical tests and interpretations of significance of features.

3.6 Other References

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
LogisticRegression.html
https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
Chen Xi, Yang Ha Cho, Yiwei Dou, and Baruch Lev (2022), "Predicting Future Earnings Changes Using Machine Learning and Detailed Financial Data," Journal of Accounting Research Vol 60, No 2.
Kian Guan Lim, "Financial Valuation and Econometrics", 2nd edition, chapter 24, World Scientific 2015.