QF634 APPLIED QUANTITATIVE RESEARCH METHODS
LECTURE 3

Lecturer: Prof Lim Kian Guan

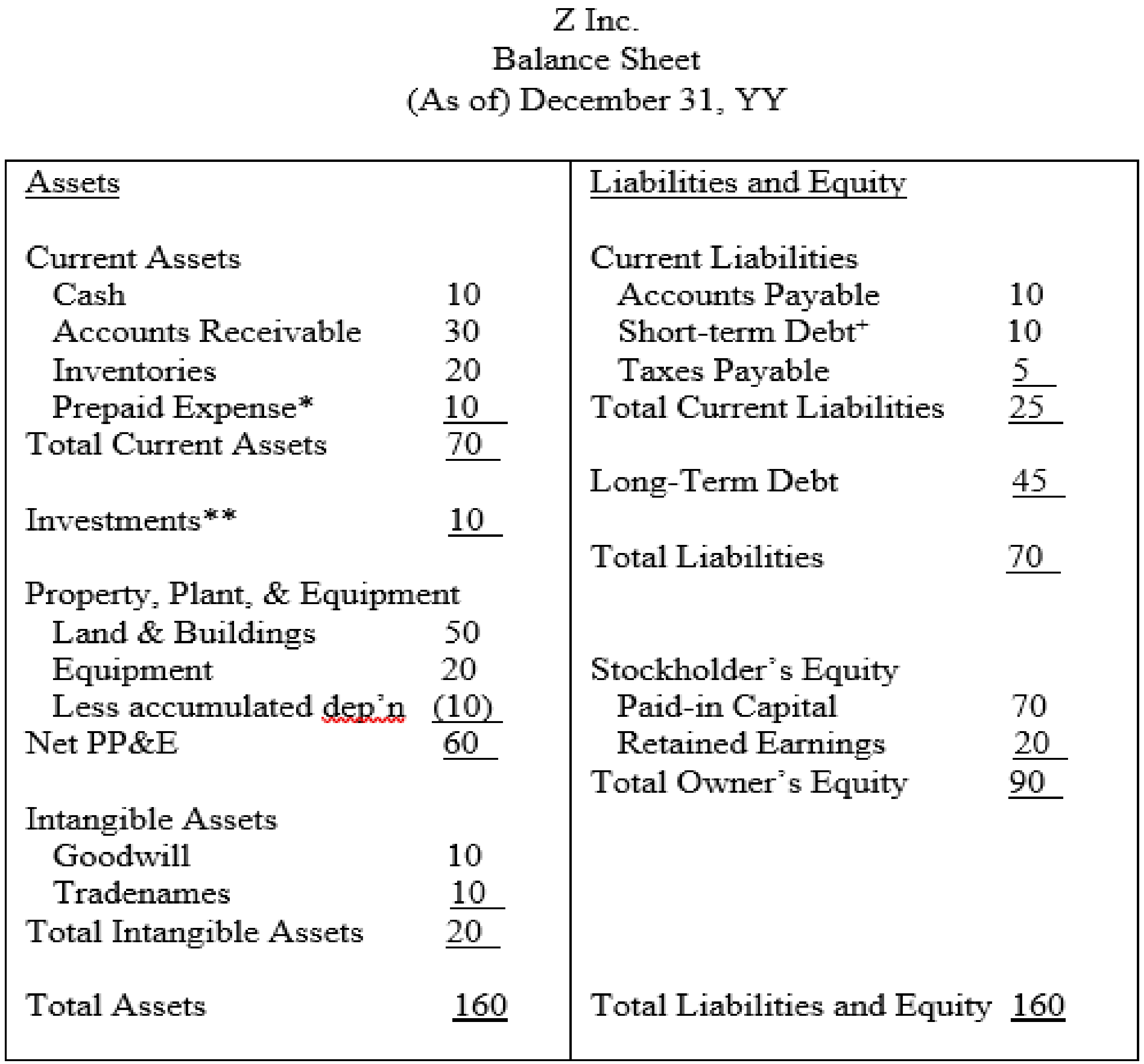
Corporate Reporting Analyses

Earnings Forecasts

Corporate Earnings Reports

- Publicly listed companies in U.S. must file quarterly (there is also the annual) reports with the SEC. The quarterly financial report is called the 10-Q report. 10-Q includes unaudited financial documents and provides a view into the company's financial condition throughout the year. Metrics like revenue, net profit/loss and adjusted profit/loss in the company's most recent financial quarter to previous quarters are reported
- Earnings report showing better (worse) company performance than that expected will usually lead to higher (lower) stock price
- Besides projected earnings, project riskiness of the firm's future earnings also affect the stock price
- In this lecture we will see how to read financial statements and do projections of the company's earnings. There are 3 key financial statements -- the Balance Sheet, the Income Statement, and the Cashflow Statement.

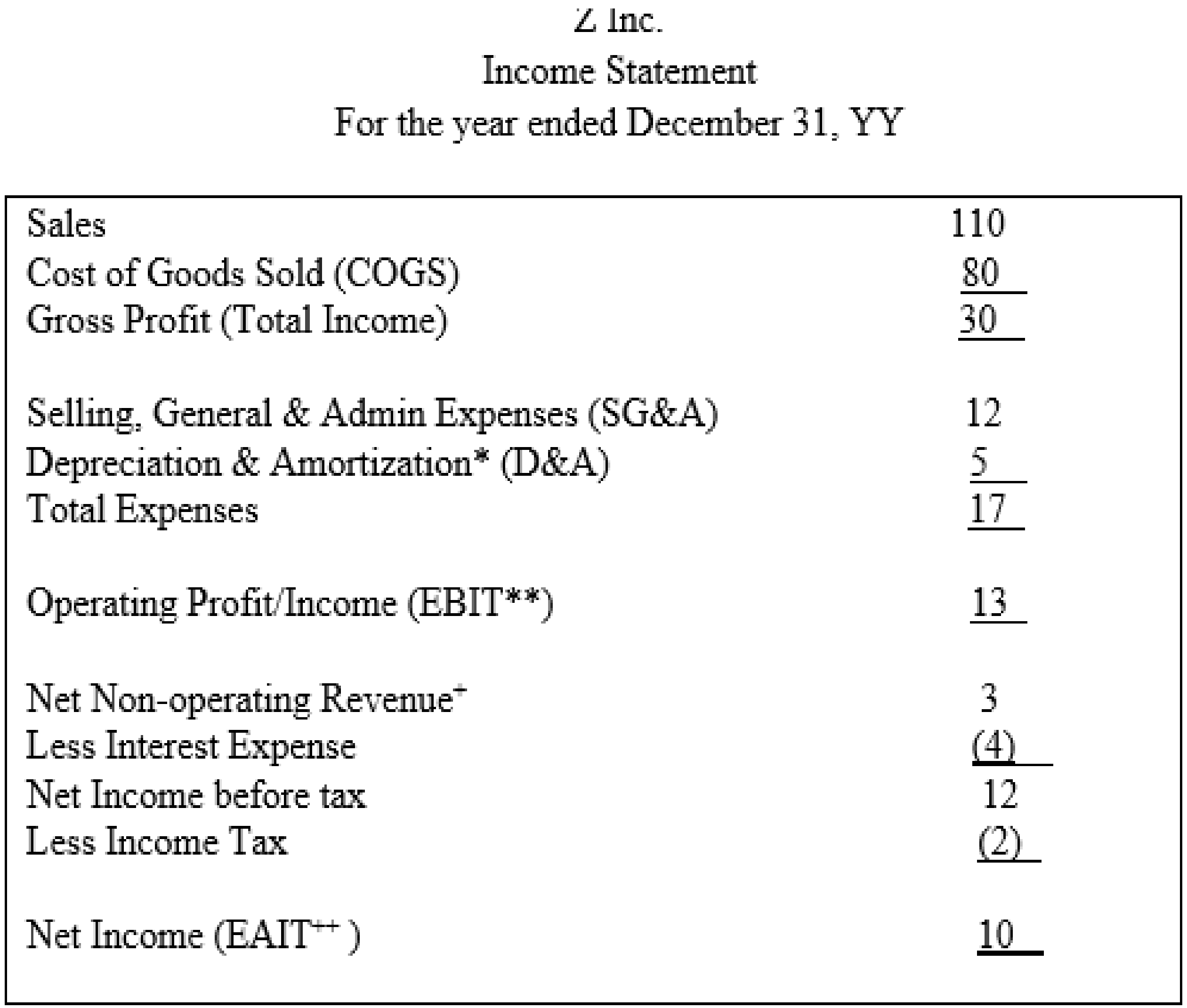
Balance Sheet



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

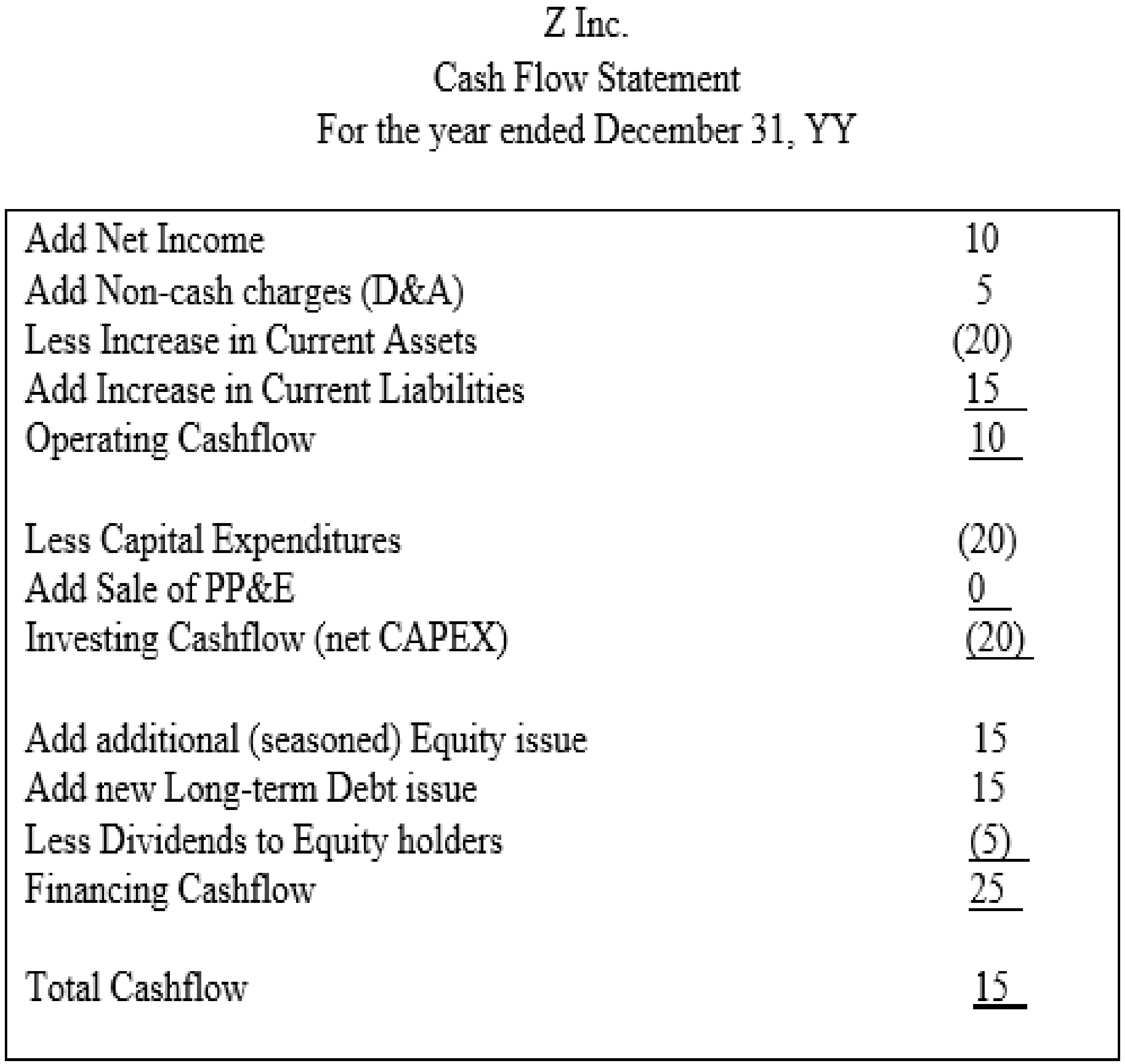
Income Statement



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

Cash Flow Statement



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

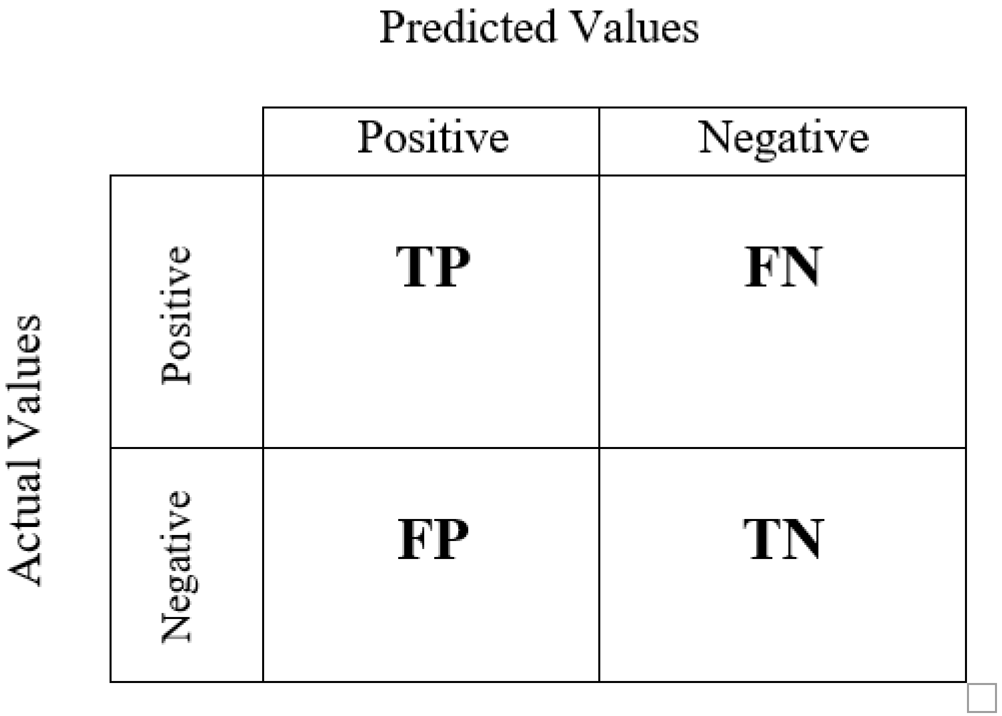
- Operating Cashflow reporting is harder to manipulate than that of net income – the revenues and costs in the Income Statement are subject to accrual basis in accounting that allows discretion by the company to report revenues (accrued revenues) before the money is received or delay reporting of cost, and thus may provide a better visual of positive net income when the cashflow could be negative.
- 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 using logistic regression. Performance measures of any prediction are discussed next.

Binary Classification Performance Metrics

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. 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 Confusion Matrix



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

Binary Classification Performance Metrics

- 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 = 900. Accuracy would still be a high 90%.

Binary Classification Performance Metrics

- 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 such as precision, recall, F1-score are computed and analyzed for the predictive ability of the algorithm.

Binary Classification Performance Metrics

Another prediction performance measure is precision, which is

Precision = TP / (TP + FP)

that shows what percent of the predicted positive cases actually 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.

Another prediction performance measure is recall, which is

Recall = TP / (TP + FN)

that shows what 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.

Binary Classification Performance Metrics

Whether precision or recall is a more informative measure depends on whether FP or FN is more of a problem, respectively.

Example: Bank lends to firms

	Predicted Pos (Default)	Predicted Neg (No Default)
Actual Positive (Firm defaults)	Correct prediction – bank can recall loan or hedge	FN is costly as bank suffers from default loss
Actual Negative (Firm does not default)	Incorrect prediction – cost is just hedging (if there is no loan recall)	TN, nothing happens to bank

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.

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.

Higher recall is more important and more informatively significant to the bank than higher precision.

Binary Classification Performance Metrics

Example: 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.

	Predicted Positive on a given segment	Predicted Negative on a given segment
Actual Positive (correct segment)	Company puts in advertising expenses – enjoys profit due to correct targeting	Company does not expense on advertisement – some lost opportunities
Actual Negative (Incorrect segment)	FP: Company puts in advertising expenses but sees little sales -- loss	Company does not expense on advertisement – nothing happens

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. Higher precision is more important and more informatively significant to the company than higher recall.

Binary Classification Performance Metrics

- Sometimes a mixture measure is used such as maximizing the F1-Score which is the harmonic mean of precision and recall, i.e., 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.
- 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 TP with respect to type B (P class) as FP_B.
- Then another type of average prediction performance is

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

- 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 $prec_B$, 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

$$\text{Macro average of precision} = (prec_A + prec_B)/2\ ,\ \text{Macro average of recall} = (rec_A + rec_B)/2$$

$$\text{Macro average of F1-score} = (f1_A + f1_B)/2$$

- 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 $n_A/(n_A+n_B) \times f1_A + n_B/(n_A+n_B) \times f1_B$.

Binary Classification Performance Metrics

Another measure is specificity or true negative rate (TNR)

$$\text{Specificity} = TN / (TN + FP)$$

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

$$FPR = 1 – \text{Specificity} = FP / (TN + FP)$$

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.

Binary Classification Performance Metrics

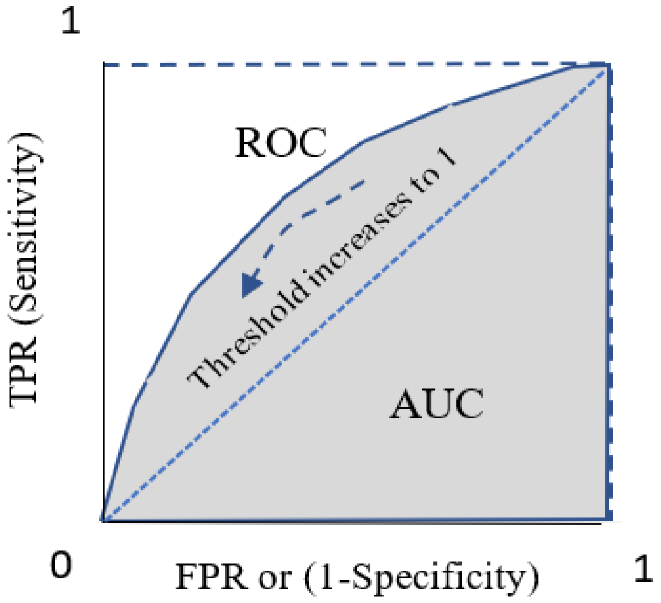
The threshold can however be adjusted to improve performance if the types are highly imbalanced. For example, in the 1000 firms' study, given the predicted probabilities by the algorithm, precision can be raised if the threshold for predicting positive is raised so that FP, which is a relatively large 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.

Binary Classification Performance Metrics

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.



ROC > 50% :
Detect more numbers of TPs and TNs (in product term) than FPs and FNs at various thresholds

Logistic Regression

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.

Logistic Regression

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 $F(\beta^T X_i) = \text{Prob}(Y_i \ = \ 1\,|X_i)$, and Σ_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.

Logistic Regression

By default sklearn LogisticRegression applies L2 regularization in the estimation. For sklearn LogisticRegression, when the estimated probability of

$$F\left(\hat{\beta}^T X_i\right) = \mathrm{Prob}(Y_i \ = \ 1 \,|X_i)$$

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

Just as in linear regression, logistic regression faces some problems when there is large multicollinearity amongst the features. 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) = $1/(1\text{-}R^2)$ is 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.

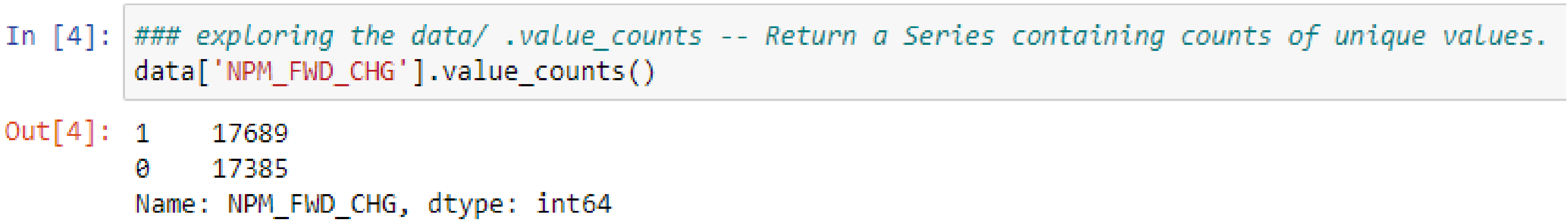
Worked Example – Data

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

- 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. 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 variable is of course not available at the time of knowing 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.

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

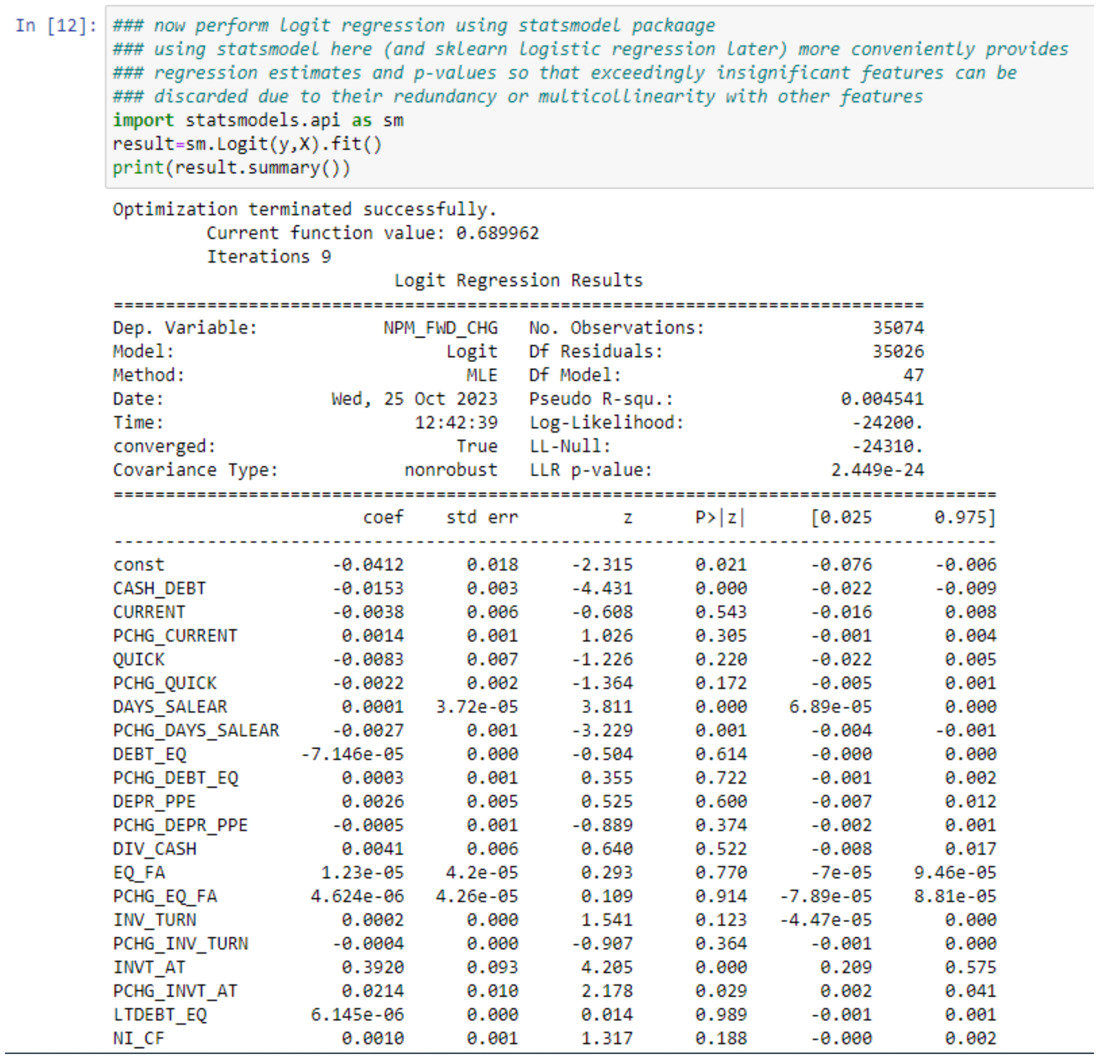
Code line [4] shows that in the data set there are 17,689 annual cases of NPM increase and 17,385 cases no NPM increase. Thus, the data set is balanced in the two categories.



```
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.



```
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
```

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

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

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

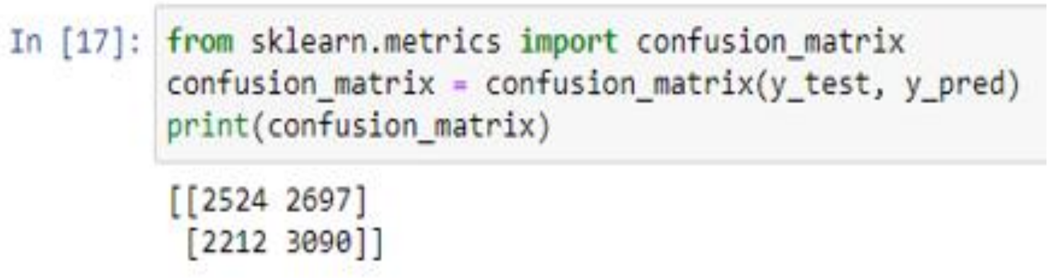
where $\mathrm{X_i}$ is the vector of 47 features for a particular company in a particular year, and $\hat{\beta}$ is the vector of 47 estimated coefficients as shown in the Logit Regression Results Table. In other words,

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

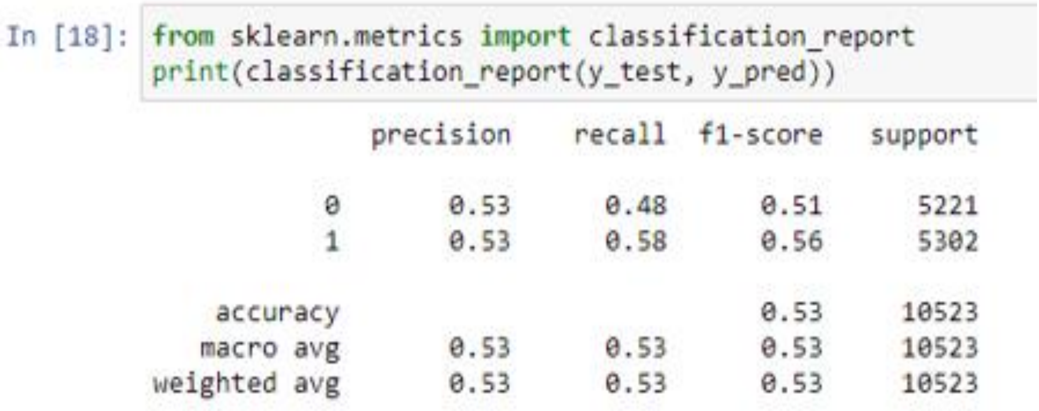
where the kth element of $\hat{\beta}$ is scalar $\hat{\beta}_k$, and its associated feature of the ith case in the logistic regression is $\mathrm{X_{ik}}$. If $\hat{\beta}_k > 0$, then when X increases, $F(\hat{\beta}^T X_i)$ or estimated Prob ($\mathrm{Y_i} = 1|\mathrm{X_i}$) increases ⇒ more likelihood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

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

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook



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

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



```
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
```

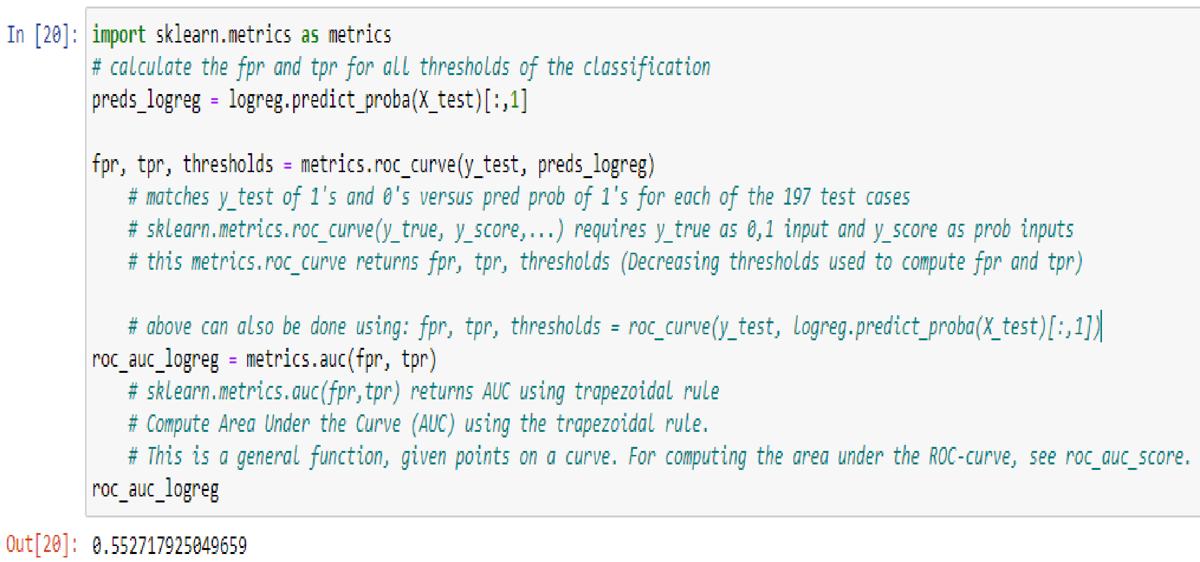
‘NPM_FWD_CHG’ = 0 as true positive

No increase in NPM

	pos	neg
pos	TP	FN
neg	FP	TN

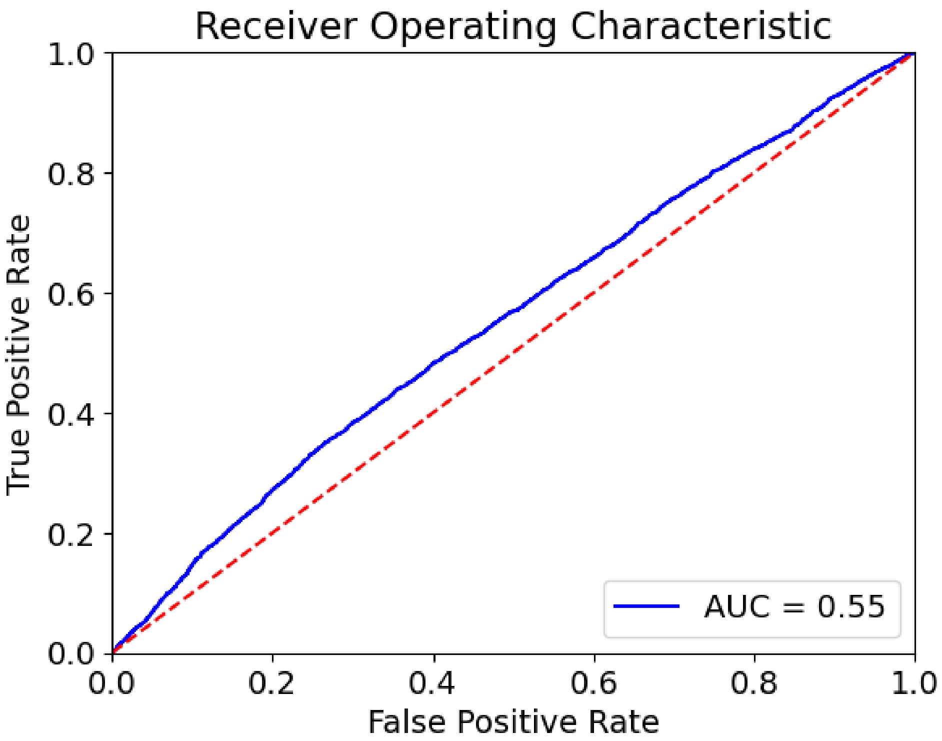
For the 5221 positive cases of no increase in NPM, the percentage of the predicted positive cases that turns out to be correctly positive is 2524/(2524+2212) = 53.29%. For the 5302 negative cases of increase in NPM, the percentage of the predicted negative cases that turns out to be correctly negative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

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.



```
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
```

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook



Illustrating How to Compute Eigenvalue, Eigenvector

Time Series
X_1: 2.1, 3.2, -0.2, 0.5, 3.1, 4.7, 1.6, -2.5, ……
X_2: 3.9, 2.8, -0.3, 5.1, 6.6, 2.8, -1.3, -4.7,……

Suppose sample $\mathrm{Cov}(X_1,X_2)/(\sigma_1\sigma_2)$ produces A
Assume $E(X_1) = E(X_2) = 0$

Solve the PCA

Take one eigenvector $V = (1/\sqrt{2}, 1/\sqrt{2})^T$
Then PCA transformed series
Z_1= $1/\sqrt{2}$ (2.1) + $1/\sqrt{2}$ (3.9), $1/\sqrt{2}$ (3.2) + $1/\sqrt{2}$ (2.8), $1/\sqrt{2}$ (-0.2) + $1/\sqrt{2}$ (-0.3), ……..

Z_2= $1/\sqrt{2}$ (2.1) - $1/\sqrt{2}$ (3.9), $1/\sqrt{2}$ (3.2) - $1/\sqrt{2}$ (2.8), $1/\sqrt{2}$ (-0.2) - $1/\sqrt{2}$ (-0.3), ……..

Correlation matrix of Z_1, Z_2 is now I_{2x2}

$$A = \begin{bmatrix} 1 & 1/3 \\ 1/3 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$A\,V = \lambda V$$

$$\begin{bmatrix} 1 & 1/3 \\ 1/3 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Thus $\begin{bmatrix} 1-\lambda & 1/3 \\ 1/3 & 1-\lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\begin{vmatrix} 1-\lambda & 1/3 \\ 1/3 & 1-\lambda \end{vmatrix} = 0$ or $(1-\lambda)^2 = 1/9$

Solve, λ =2/3. x = $1/\sqrt{2}$, y = $-1/\sqrt{2}$

λ =4/3. x = $1/\sqrt{2}$, y = $1/\sqrt{2}$

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 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. This would alleviate the multicollinearity issue in regression problems, including the logistic regression.

A Brief Theory of Principal Component Analysis

- Suppose the N features are random variables x_1, x_2, x_3, …., x_N. Now let $X_{N\times1} = (x_1, x_2, x_3, …., x_N)^T$ 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 λ_1, λ_2, λ_3, …, λ_N be the N eigenvalues and v_1, v_2, v_3, …, v_N be the N corresponding eigenvectors (dimension N × 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 $\Lambda_{N\times N} = \text{diag}(\lambda_1, \lambda_2, \lambda_3, …, \lambda_N)$, 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.

A Brief Theory of Principal Component Analysis

- Let $\Gamma_{N\times N} = (v_1, v_2, v_3, \dots, v_N)$ 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 jth scalar principal component (PC) of X. There are N such PCs since there are j number of eigenvectors v_j.
- Let $Y_{N\times 1} = (Y_1, Y_2, Y_3, \dots, Y_N)^T$ 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 $[\mathrm{cov}(Y_i,Y_j)]_{N\times N} = E(YY^T) = E(\Gamma^T(X - \mu)(X - \mu)^T\Gamma) = \Gamma^T\Sigma\Gamma$ given Γ.
- But by the definition of eigenvalues and eigenvectors, $\Sigma\Gamma = \Gamma\Lambda$. Hence,

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

- 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.

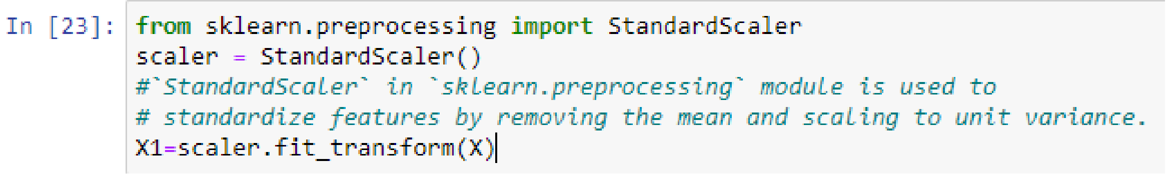
A Brief Theory of Principal Component Analysis

- Since $\Gamma^T\Gamma = \Gamma\Gamma^T = I_{N\times N}$, trace of Λ or $tr(\Lambda) = tr(\Gamma^T\Sigma\Gamma) = tr(\Sigma\Gamma\Gamma^T) = tr(\Sigma) = \sum_{j=1}^{N} var(x_j)$ 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 $tr(\Lambda)$ 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., $var(Y_j)/\sum_{j=1}^{N} var(x_j)$. 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 $\hat{\mu} = (\hat{\mu}_1, \hat{\mu}_2, \dots, \hat{\mu}_N)$ 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}, …., x_{jT}. T is the sample size. Estimate of Σ is $\hat{\Sigma} = \left[\widehat{cov}(x_i, x_j)\right]$ where $\widehat{cov}(x_i, x_j) = \frac{1}{T}\sum_{t=1}^{T}(x_{it} - \hat{\mu}_i)(x_{jt} - \hat{\mu}_j)$.
- 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.

Principal Component Analysis

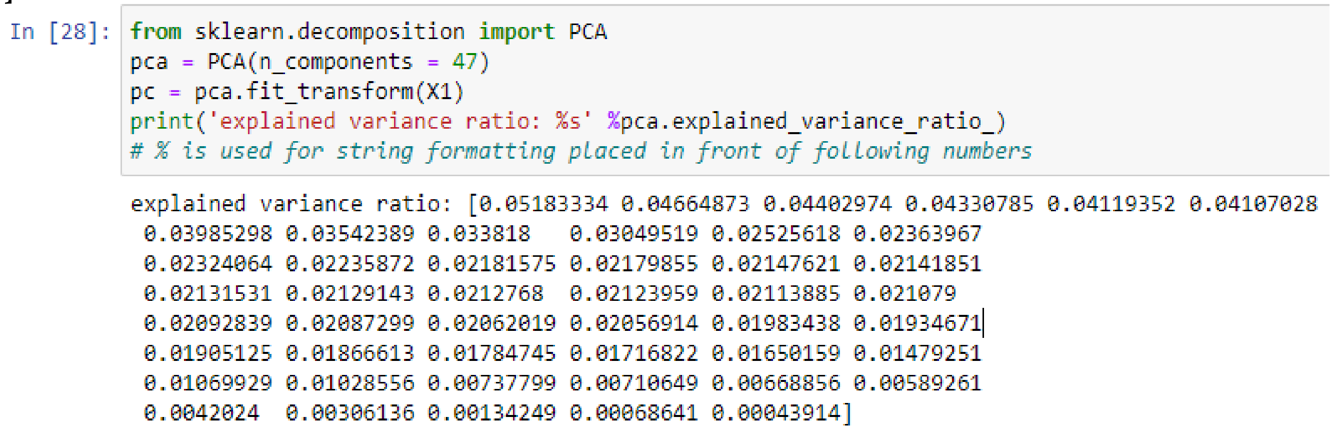
Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

We will perform a principal component analysis and possible dimension reduction of the feature space. 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)
```

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.

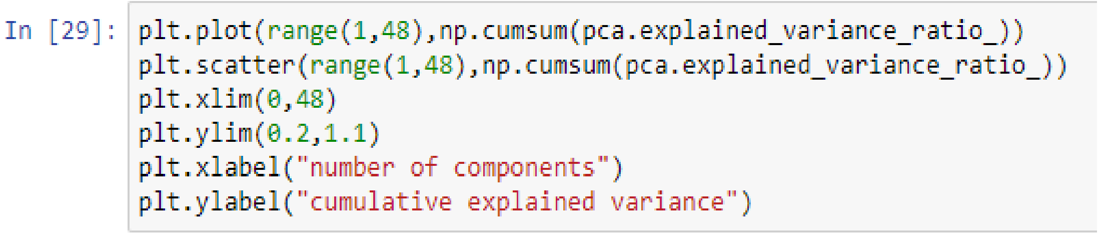


```
In [28]: from sklearn.decomposition import PCA
         pca = PCA(n_components = 47)
         pc = pca.fit_transform(X1)
         print('explained variance ratio: %s' %pca.explained_variance_ratio_)
         # % is used for string formatting placed in front of following numbers

         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]
```

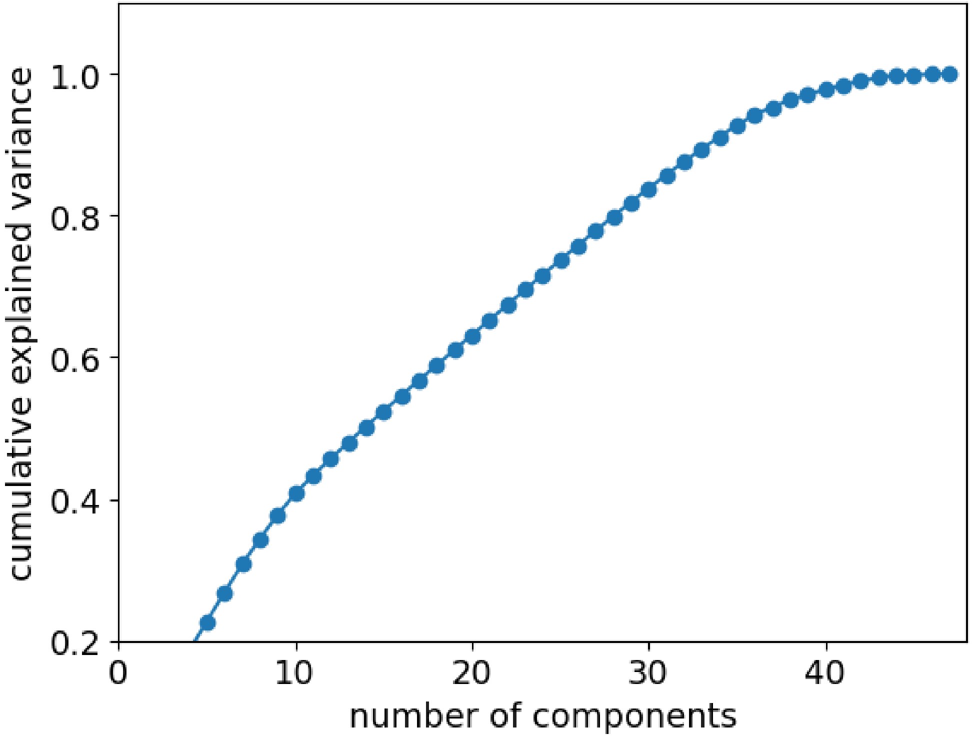
Principal Component Analysis

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

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")
```

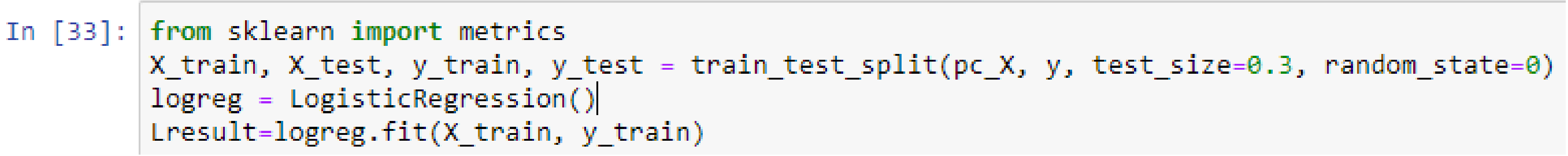


Principal Component Analysis

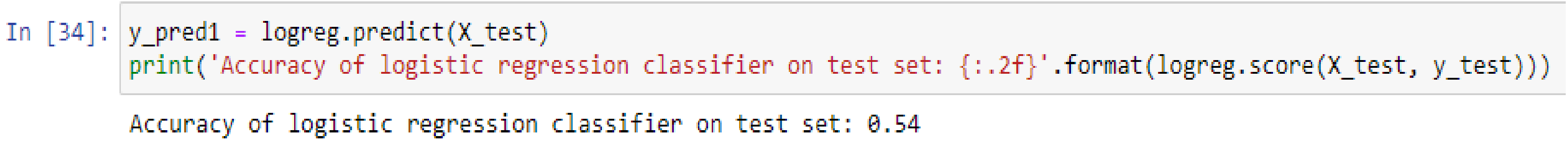
Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

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)
```

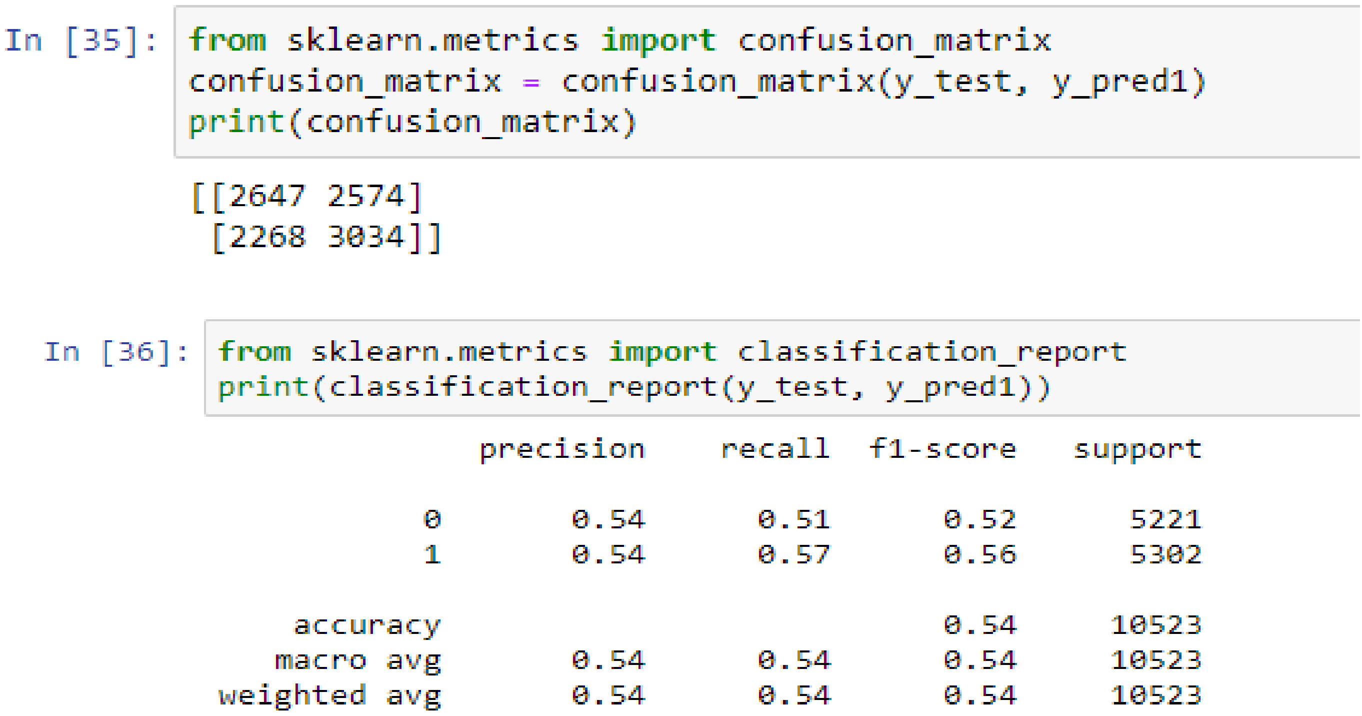


```
In [34]: y_pred1 = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

         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

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

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

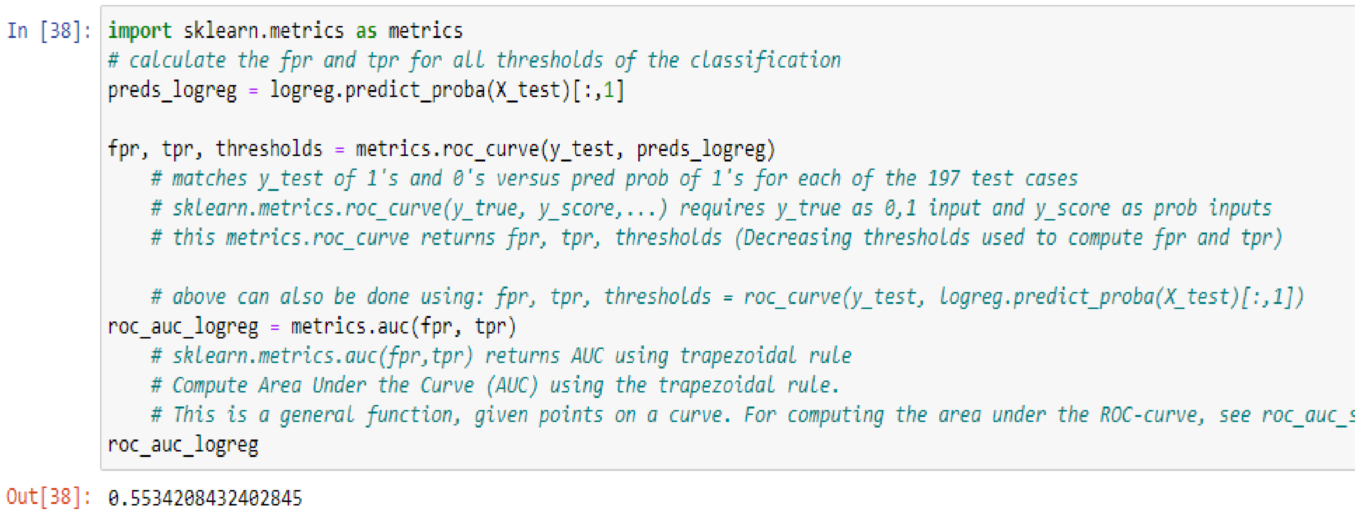


```
In [35]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred1)
         print(confusion_matrix)

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

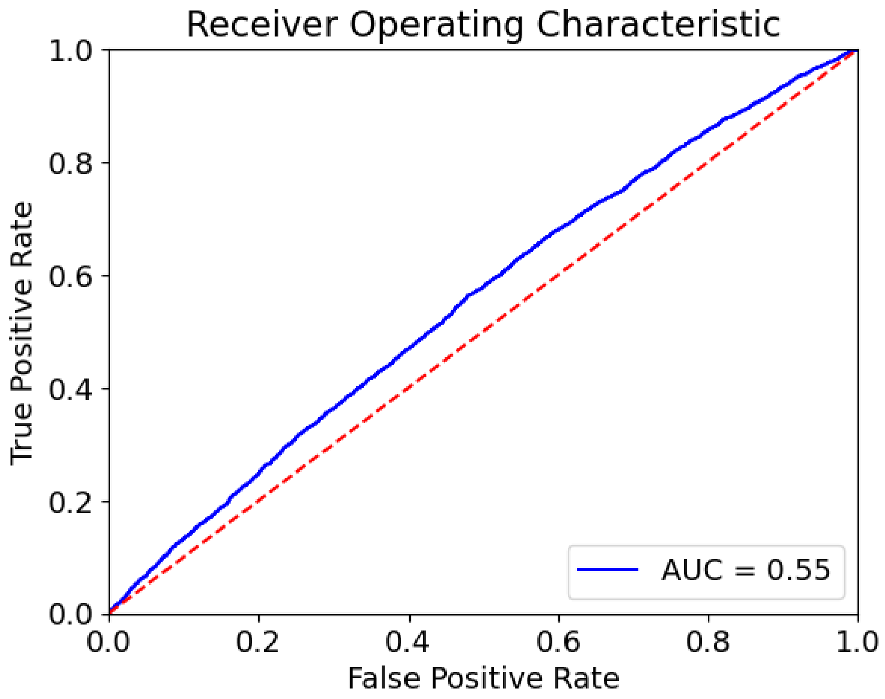
```
In [36]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred1))

                      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
```

Please upload Chapter3-1.ipynb and follow the computing steps in Jupyter Notebook

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_s
         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.

Homework 2 Graded Exercise:

Chapter3-2.ipynb (this file is provided only after grading)

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed. Data set is public – acknowledgement: S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.

Download from ‘banking3.csv’. This is a simplified version of the original data set. There are 18 features and one label variable y. y = 1 denotes success in getting client to put in a term deposit. y=0 denotes failure to do so.

The features are self-explanatory. Many variables, e.g., job_blue-collar = 1 if the potential client is a blue-collar worker, 0 otherwise. This is a dummy variable. Default_no = 1 means client has defaulted in some loans before, 0 otherwise. Contact_cellular = 1 if it is contact by handphone (not land line telephone) and 0 otherwise. Month_apr = 1 means last contact timing is previous April, 0 otherwise.

Poutcome_failure = 1 means client was approached in previous marketing campaign and there was failure to secure a deposit from client, 0 means client was not in previous campaign listing.

Poutcome_success = 1 means client was approached in previous marketing campaign and there was success to secure a deposit from client, 0 means client was not in previous campaign listing.

Euribor3m is the 3-month Euribor interest rate prevailing.

Perform a logistic regression as follows:

```
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)
 ……..
```

and so on.
Report the confusion matrix, the classification report, and show the Area under the ROC Curve.
(Follow format of the MCQ in answering the above.)

End of Class