Binary Logistic Regression

Checking Model Performance

Contents

Receiver Operating Characteristic Curve

- The **R**eceiver **O**perating **C**haracteristic (ROC) curve is

> A graphical representation of the trade off between the false positive and true positive rates for various cut off values

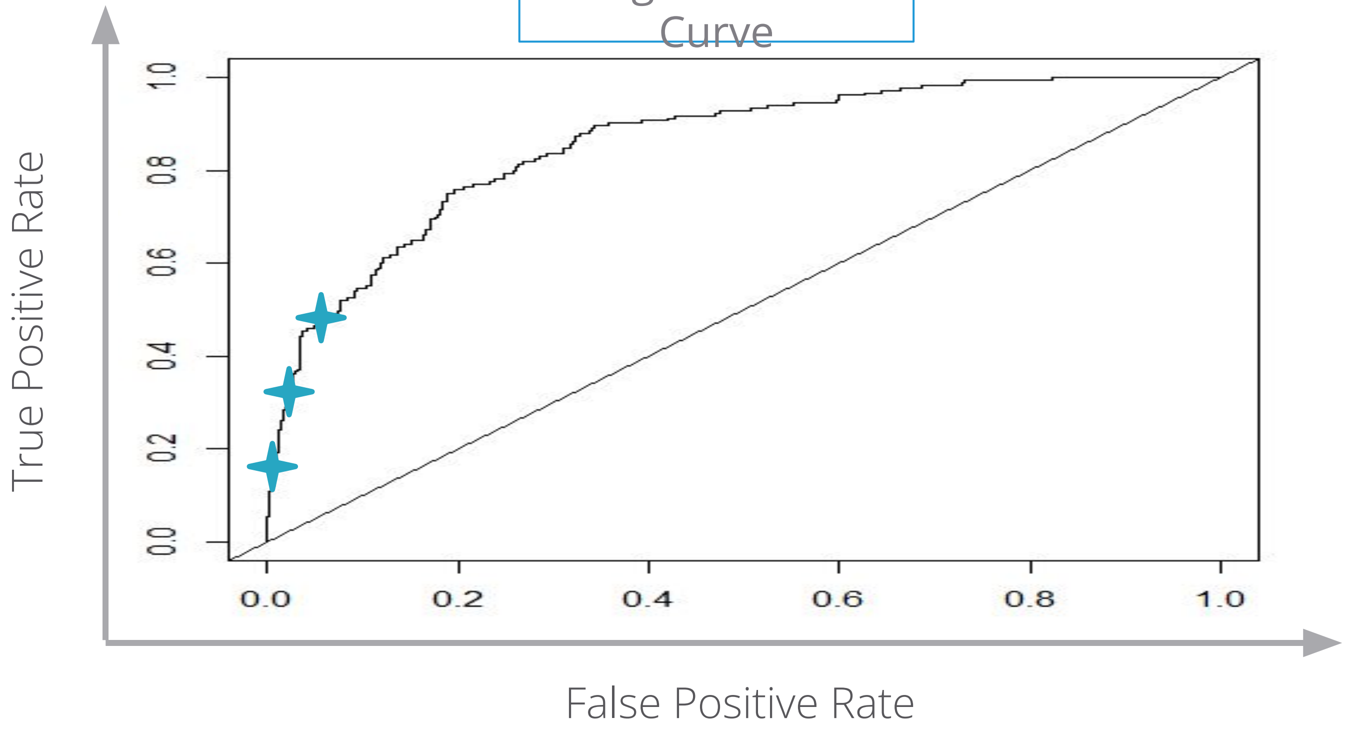
Y- axis: Sensitivity (true positive rate)

X-axis: 1-Specificity (false positive rate)

The performance of the classification model can be assessed by area under the ROC curve (C).

ROC Curve and Area Under ROC Curve

Fig. 01: ROC Curve



High TPR with low FPR is indicative of a good model. This will result in curve that is closer to the Y-axis and top left corner of the plot. It implies higher Area Under the ROC Curve.

ROC Curve and Area Under ROC Curve

Interpreting different versions of an ROC curve

Critical Points	Interpretations
TPR = 0 and FPR = 0	Model predicts every instance to be Non-event
TPR = 1 and FPR = 1	Model predicts every instance to be Event
TPR = 1 and FPR = 0	The Perfect Model

- If the model is perfect, AUC = 1
- If the model is guessing randomly, AUC = 0.5
- Thumb rule: Area Under ROC Curve > 0.65 is considered acceptable

ROC in Python

```
# Importing bank loan data & Fitting Binary Logistic Regression model
```

```
import pandas as pd
bankloan=pd.read_csv('BANK LOAN.csv')

bankloan['AGE']=bankloan['AGE'].astype('category')

import statsmodels.formula.api as smf
riskmodel = smf.logit(formula = 'DEFAULTER ~  EMPLOY + ADDRESS +
DEBTINC + CREDDEBT', data = bankloan).fit()

from sklearn.metrics import roc_curve, auc
bankloan=bankloan.assign(pred=riskmodel.predict())
fpr, tpr, thresholds = roc_curve(bankloan['DEFAULTER'],
bankloan['pred'])
```

- Import **roc_curve, auc** from sklearn.metrics
- **predict()** function prepares data required for ROC curve.
- **roc_curve()** computes Receiver operating characteristic (ROC), it returns "tpr“ (True positive rate), "fpr” (False positive rate) and threshold.

ROC in Python

```
# AUC & ROC plot
```

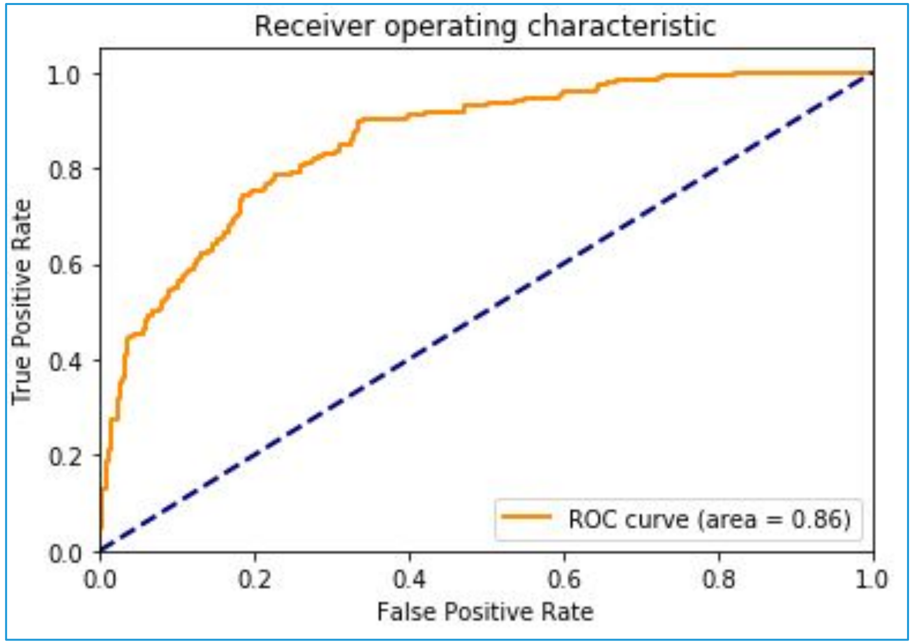
```
ruc_auc = auc(fpr,tpr)

import matplotlib.pyplot as plt

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',lw=lw, label='ROC curve (area =
%0.2f)' % ruc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0]);plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate');plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic');plt.legend(loc="lower
right")
plt.show()
```

- **auc()** gives area under curve

- **plot()** function plots the objects created using roc_curve. Entire code of plot should be run in single chunk.

ROC in Python

```
# Output:
```




```
# AUC value
print("Area under the ROC curve : %f" % ruc_auc)
# Output:
Area under the ROC curve : 0.855619
```

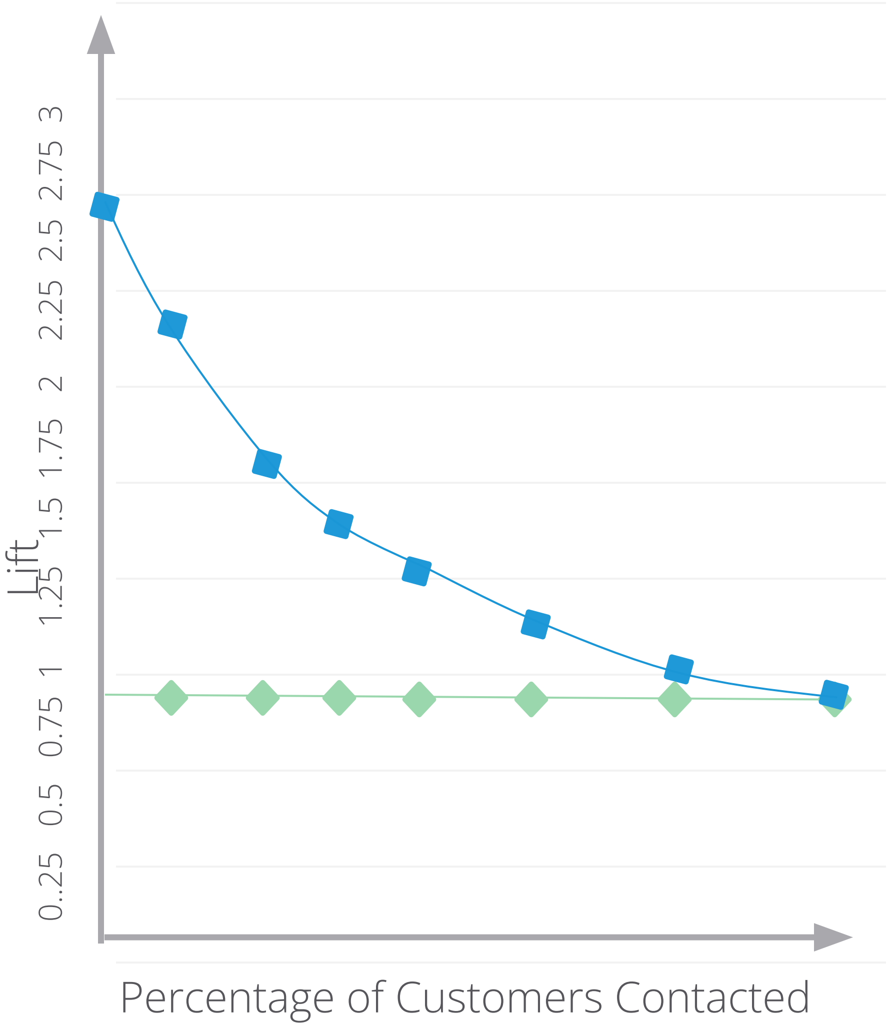
Interpretation :

- Area under the curve is 0.8556 which means model is performing well

Lift Curve

- The idea is to quantify and compare two scenarios-One is using the model to identify certain cases and second using random selection of cases for specific purpose like marketing campaign.
- Lift is the ratio of results obtained **with and without a model**.
- Although primarily used in marketing analytics, the concept finds applicability in other domains as well, such as risk modeling, supply chain analytics, etc.

Lift Curve



Lift Curve: After contacting X% of customers, Y% of respondents will be identified if statistical model is used.

Ratio Y/X is plotted

Baseline: After contacting X% of customers, X% of respondents will be identified if random method is used.

Ratio X/X is plotted

Lift Curve in Python

Install "scikit-plot" library in Anaconda Prompt and load in Python

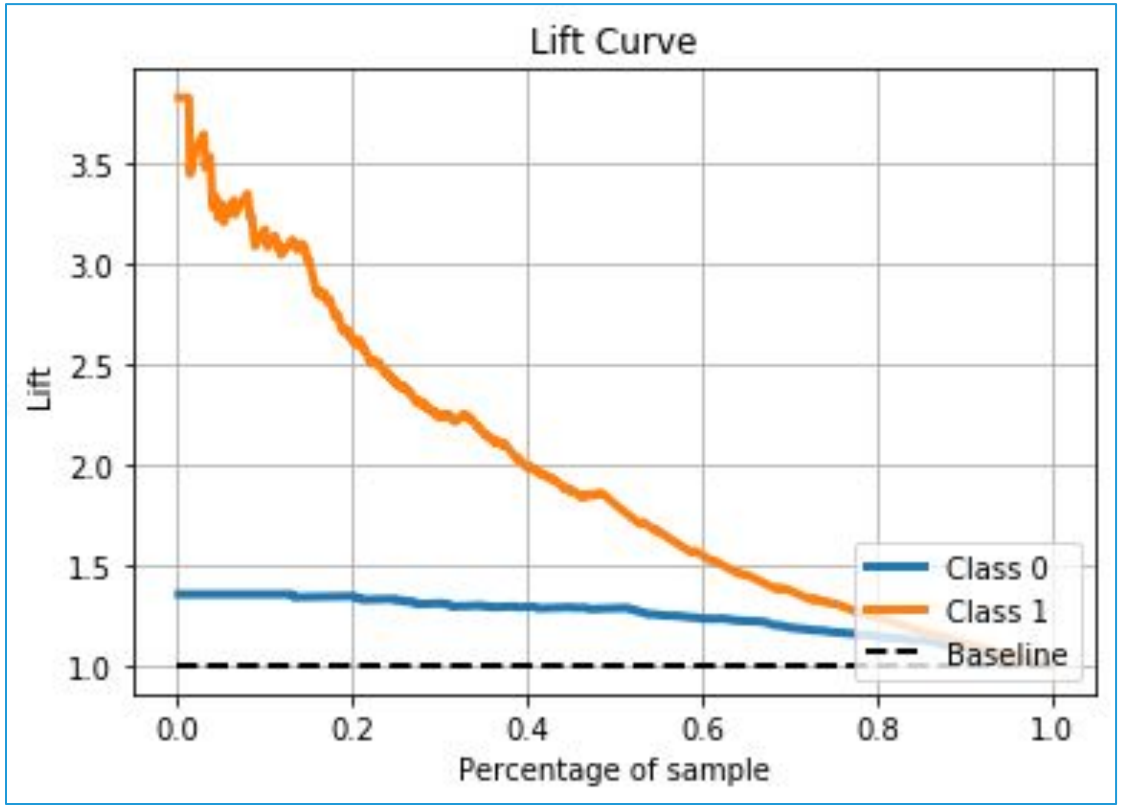
```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import scikitplot as skplt

X = bankloan[['EMPLOY', 'ADDRESS', 'DEBTINC', 'CREDDEBT’]]
y = bankloan[['DEFAULTER']]
log_model = LogisticRegression()
log_model.fit(X,y)
pred_log = log_model.predict_proba(X)

skplt.metrics.plot_lift_curve(y, pred_log)
plt.show()
```

- **LogisticRegression()** fits a Logistic Regression model
- **predict_proba()** Return probability estimates for the test vector X.
- **scikitplot()** depends on scikit-learn and … s … plots for sklearn

* Note : Incase import scikitplot as skplt gives an error, then just run pip install scikit-plot in anaconda promt & then execute import code in python

Lift Curve in Python

```
# Output:
```



Interpretation :

- Model is performing better. As more defaulters identified in earlier buckets.

Kolmogorov-Smirnov Statistic

Kolmogorov-Smirnov (KS) Statistics is one of the most commonly used measures to assess predictive power for marketing or credit risk models

KS is maximum difference between % cumulative Goods and Bads distribution across probability bands

The gains table typically has % cumulative Goods (or Non-Event) and % Cumulative Bads (Or Event) across 10 or 20 probability bands

- **KS is a point estimate**, meaning it is only one value and indicate the probability band where separation between Goods (or Non-Event) and Bads (or Event) is maximum
- Theoretically **K-S can range from 0-100**. **KS less than 25, may not indicate good model**. Too high value should also be evaluated carefully

Kolmogorov-Smirnov Statistic

BAND	Count	Percent	Count(bad)	%(bad)	Count(good)	%(good)	cum% bad	cum% good	KS
0.95-1	10	1.4%	9	4.9%	1	0.2%	4.9%	0.2%	4.7%
0.90-0.95	7	1.0%	7	3.8%	0	0.0%	8.7%	0.2%	8.5%
0.85-0.90	7	1.0%	6	3.3%	1	0.2%	12.0%	0.4%	11.6%
0.80-0.85	7	1.0%	5	2.7%	2	0.4%	14.8%	0.8%	14.0%
0.75-0.80	11	1.6%	9	4.9%	2	0.4%	19.7%	1.2%	18.5%
0.70-0.75	17	2.4%	14	7.7%	3	0.6%	27.3%	1.7%	25.6%
0.65-0.70	17	2.4%	12	6.6%	5	1.0%	33.9%	2.7%	31.2%
0.60-0.65	10	1.4%	7	3.8%	3	0.6%	37.7%	3.3%	34.4%
0.55-0.6	24	3.4%	14	7.7%	10	1.9%	45.4%	5.2%	40.1%
0.5-0.55	21	3.0%	9	4.9%	12	2.3%	50.3%	7.5%	42.7%
0.45-0.5	22	3.1%	9	4.9%	13	2.5%	55.2%	10.1%	45.1%
0.40-0.45	31	4.4%	13	7.1%	18	3.5%	62.3%	13.5%	48.8%
0.35-0.4	29	4.1%	11	6.0%	18	3.5%	68.3%	17.0%	51.3%
0.3-0.35	27	3.9%	13	7.1%	14	2.7%	75.4%	19.7%	55.7%
0.25-0.3	40	5.7%	7	3.8%	33	6.4%	79.2%	26.1%	53.1%
0.2-0.25	45	6.4%	12	6.6%	33	6.4%	85.8%	32.5%	53.3%
0.15-0.2	52	7.4%	10	5.5%	42	8.1%	91.3%	40.6%	50.6%
0.10-0.15	66	9.4%	4	2.2%	62	12.0%	93.4%	52.6%	40.8%
0.05-0.1	80	11.4%	8	4.4%	72	13.9%	97.8%	66.5%	31.3%
0-0.05	177	25.3%	4	2.2%	173	33.5%	100.0%	100.0%	0.0%
Total	**700**	**100%**	183	**100%**	517	**100%**			

Kolmogorov-Smirnov Statistic in Python

```
# Combine observed and expected frequencies
```

```
from scipy.stats import ks_2samp
ks_2samp(bankloan.loc[bankloan.DEFAULTER==0,'pred'],
bankloan.loc[bankloan.DEFAULTER==1,'pred'])
```

- **ks_2samp** computes the kolmogorov-smirnov statistic on 2 samples.
- It returns KS statistic and two-tailed p-value

```
# Output:
```

```
Ks_2sampResult(statistic=0.561552039403452, pvalue=1.909421801103993e-37)
```

Interpretation :

- Maximum difference (K-S statistic) is 0.561552.

Pearson Residuals

- Pearson residual is defined as the standardized difference between the observed and predicted frequency.it measures the relative deviations between the observed and fitted values. :

$$r_j = \frac{(Y_j - M_j p_j)}{\sqrt{M_j p_j (1 - p_j)}}$$

where
Mj : number of observations with jth covariate pattern
Y_j : Observed value (1 or 0) for jth covariate pattern
p_j : Predicted probability for j^{th} covariate pattern

- Binary Logistic Regression **does not require 'Normality' of residuals**

Pearson Residuals in Python

```
# Obtain residuals
bankloan=bankloan.assign(resid=riskmodel.resid_pearson)

bankloan.head()
```

- **resid_pearson()** calculates Pearson residuals.

```
# Output:
```

	SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER	pred	resid
0	1	3	17	12	9.3	11.36	5.01	1	0.808346726	0.486921868
1	2	1	10	6	17.3	1.36	4	0	0.198114704	-0.497052463
2	3	2	15	14	5.5	0.86	2.17	0	0.010062815	-0.100822141
3	4	3	15	14	2.9	2.66	0.82	0	0.022159721	-0.150538706
4	5	1	2	0	17.3	1.79	3.06	1	0.781808095	0.528286162

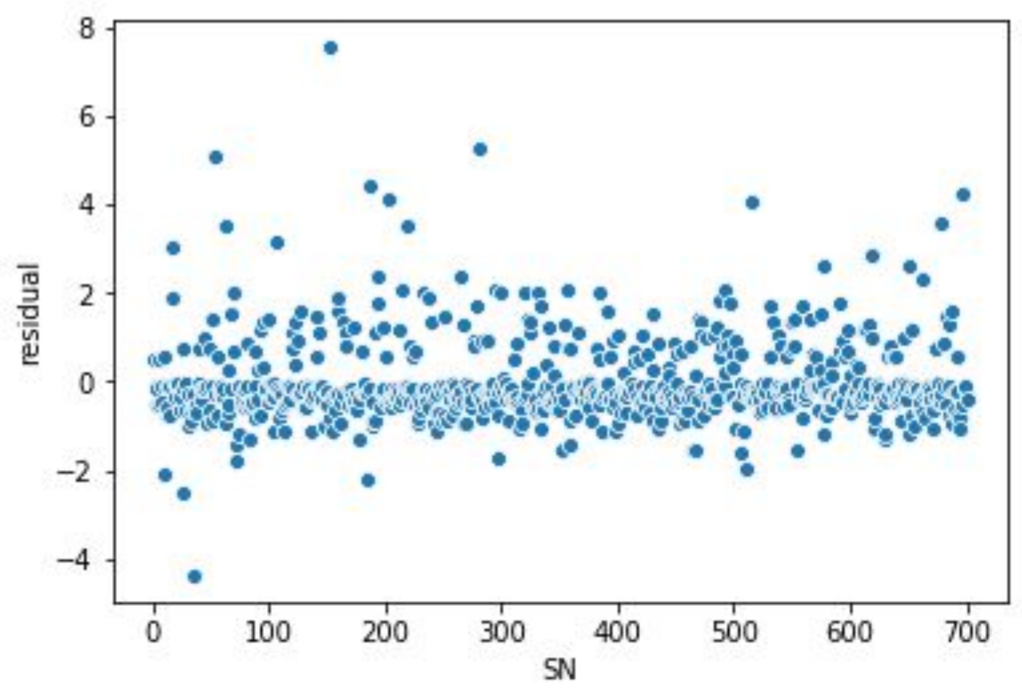
Residuals

Pearson Residuals Plot in Python

```
# Residuals Plot
```

```
import seaborn as sns
sns.scatterplot('SN','resid',data=bankloan); plt.xlabel('SN');
plt.ylabel('residual')
```

```
# Output:
```



Clearly one case has very high residual value.

Multicolinearity

- Multicollinearity exists if there is a strong linear relationship among the continuous independent variables.
- Do not ignore multicollinearity in Binary Logistic Regression .
- Use variance inflation factors to detect multicolliearity.

* **Multicolinearity is explained in MLR module.**

Quick Recap

In this session, we learnt about **checking model performance :**

ROC	• Graphical representation of the trade off between the false positive (FPR) and true positive (TPR) rates for various cut off values.
Lift curve	• Lift Curve Compares model results with baseline without model
K-S statistic	• KS is the maximum difference between % cumulative Goods (event/Y=1) and cumulative Bads (non events/Y=0) distribution across probability groups.
Residual	• Pearson's residual is used for binary logistic regression
Multicollinearity	• Multicollinearity exists if there is a strong linear relationship among the continuous independent variables