



Ενότητα 6 – Supervised Learning *Logistic Regression*

Μέθοδοι Μηχανικής Μάθησης στα Χρηματοοικονομικά

Αθανάσιος Σάκκας, Επ. Καθηγητής, ΟΠΑ

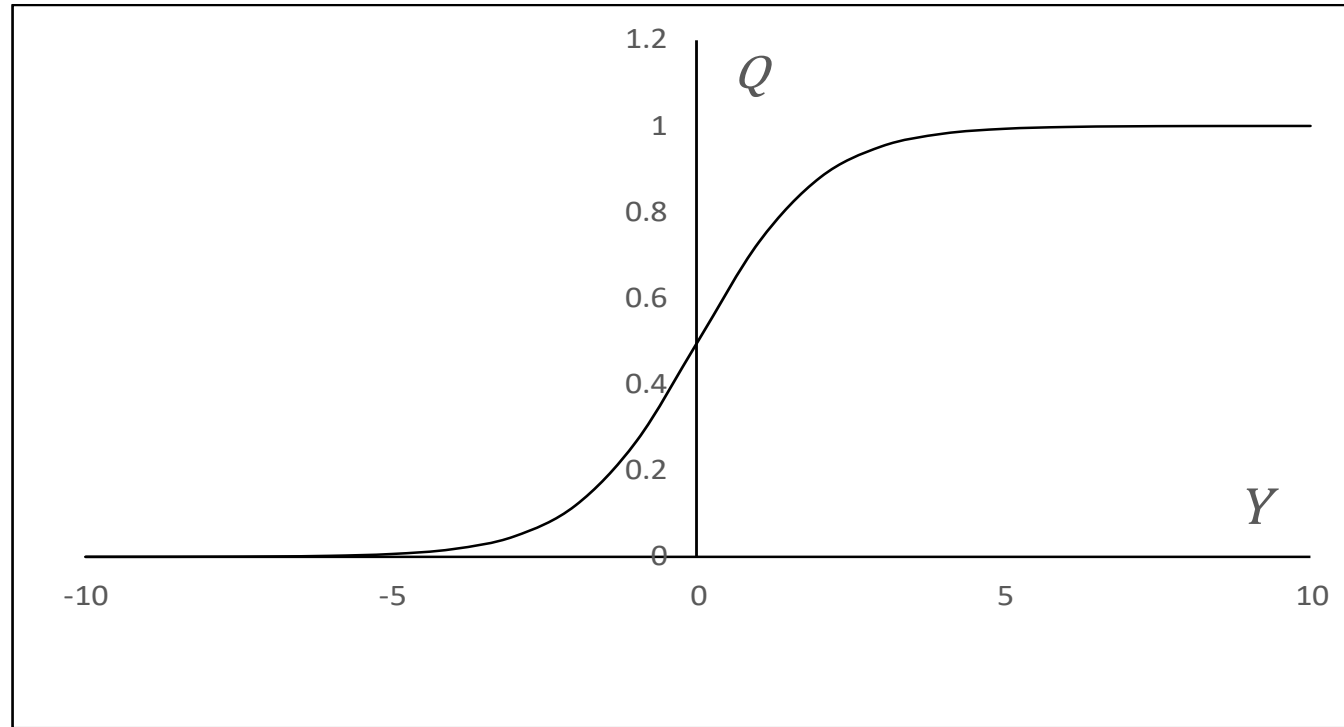
Logistic (ή Logit) Regression

- Ο στόχος είναι να ταξινομηθούν οι παρατηρήσεις σε "θετικό αποτέλεσμα" και "αρνητικό αποτέλεσμα" χρησιμοποιώντας ως δεδομένα χαρακτηριστικά.
- Η πιθανότητα θετικού αποτελέσματος θεωρείται ότι είναι μια σιγμοειδής συνάρτηση (sigmoid ή logistic function): $Q = \frac{1}{1+e^{-Y}}$

όπου το Y σχετίζεται γραμμικά με τις τιμές των χαρακτηριστικών:

$$Y = a + b_1X_1 + b_2X_2 + \dots + X_m$$

Σιγμοειδής συνάρτηση (sigmoid ή logistic function)



Maximum Likelihood Estimation

- Χρησιμοποιούμε το training set για να μεγιστοποιήσουμε

$$\sum_{\text{Positive Outcomes}} \ln(Q) + \sum_{\text{Negative Outcomes}} \ln(1 - Q)$$

Ισοδύναμα μπορούμε να ελαχιστοποιήσουμε μια συνάρτηση κόστους

$$\frac{1}{n} \left[- \sum_{\text{Positive Outcomes}} \ln(Q) - \sum_{\text{Negative Outcomes}} \ln(1 - Q) \right]$$

- Αυτή η μεγιστοποίηση (ελαχιστοποίηση) δεν μπορεί να γίνει αναλυτικά, αλλά μπορούμε να χρησιμοποιήσουμε έναν αλγόριθμο gradient ascent
- Για να συμπεριλάβουμε το regularization μπορούμε να προσθέσουμε $\lambda \sum b_j^2$ ή $\lambda \sum |b_j|$ στη συνάρτηση που ελαχιστοποιείται

Εφαρμογή LendingClub

- Τα δεδομένα αποτελούνται από δάνεια, καλά (good) ή αθετημένα (defaulted).
- Εμείς θα χρησιμοποιήσουμε μόνο τέσσερα χαρακτηριστικά
 1. Ιδιοκτησία σπιτιού (ενοικίαση έναντι ιδιοκτησία) - Home ownership (rent vs. own)
 2. Εισόδημα – Income
 3. Χρέος προς εισόδημα - Debt to income
 4. Πιστωτική βαθμολογία - Credit score
- Το σύνολο δεδομένων έχει 12.290 παρατηρήσεις (9.733 καλά δάνεια “Fully Paid” και 2.557 δάνεια αθέτησης “Charged Off”). 7.000 δάνεια τέθηκαν σε training set, 3.000 σε validation set, και 2.290 σε test set.

Κάποια δεδομένα (unscaled) από το training, validation και test set

```
#print first five instances for each data set - unscaled
print("Unscaled data")
print("Training set")
print(X_train_unscaled.head())
print("Validation set")
print(X_val_unscaled.head())
print("Test set")
print(X_test_unscaled.head())
```

Unscaled data				
Training set				
	home_ownership	income	dti	fico
0	1	44304.0	18.47	690
1	0	50000.0	29.62	735
2	0	64400.0	16.68	675
3	0	38500.0	33.73	660
4	1	118000.0	26.66	665
Validation set				
	home_ownership	income	dti	fico
0	0	25000.0	27.60	660
1	0	50000.0	21.51	715
2	1	100000.0	8.14	770
3	0	75000.0	1.76	685
4	1	78000.0	16.11	680
Test set				
	home_ownership	income	dti	fico
0	1	52400.0	24.64	665
1	1	150000.0	17.04	785
2	1	100000.0	20.92	710
3	0	97000.0	13.11	705
4	1	100000.0	24.08	685

Αποτελέσματα σε Training Set (Unscaled data)

```
!|: #Create an instance of Logistic regression named lgstc_reg on unscaled data
lgstc_reg = LogisticRegression(penalty="none", solver="newton-cg")
# Fit Logistic regression to training set
lgstc_reg.fit(X_train_unscaled, y_train) # fit training data on logistic regression
print(lgstc_reg.intercept_, lgstc_reg.coef_) # get the coefficients of each features
[-5.98082741] [[ 2.97659316e-01  6.17979317e-07 -3.65499148e-02  1.12603137e-02]]
```

$X_1 = \text{Home Ownership}$

$X_2 = \text{Income}$

$X_3 = \text{Debt to income ratio}$

$X_4 = \text{Credit score}$

$$Y = -5.981 + 0.297X_1 + 0.0000006X_2 - 0.0365X_3 + 0.0113X_4$$

$$\text{Probability of a good loan} = \frac{1}{1+e^{-Y}}$$

π.χ. Για το πρώτο δάνειο η πιθανότητα να είναι καλό είναι $\frac{1}{1+e^{-Y}}$, όπου $Y = -5.981 + 0.297 * 1 + 0.0006 * 44304 - 0.0365 * 18.47 + 0.0113 * 690 = 1.439$, άρα $\text{Probability of a good loan} = \frac{1}{1+e^{-Y}} = 0.808$.

Κάποια δεδομένα (scaled) από το training, validation και test set

```
#print first five instances for each data set - scaled
print("Scaled data")
print("Training set")
print(X_train.head())
print("Validation set")
print(X_val.head())
print("Test set")
print(X_test.head())
```

```
Scaled data
Training set
  home_ownership  income      dti      fico
0      0.809651 -0.556232  0.053102 -0.163701
1     -1.234923 -0.451393  1.307386  1.262539
2     -1.234923 -0.186349 -0.148259 -0.639114
3     -1.234923 -0.663060  1.769728 -1.114527
4      0.809651  0.800204  0.974410 -0.956056
Validation set
  home_ownership  income      dti      fico
0     -1.234923 -0.911538  1.080153 -1.114527
1     -1.234923 -0.451393  0.395077  0.628655
2      0.809651  0.468899 -1.108940  2.371837
3     -1.234923  0.008753 -1.826638 -0.322172
4      0.809651  0.063971 -0.212379 -0.480643
Test set
  home_ownership  income      dti      fico
0      0.809651 -0.407219  0.747177 -0.956056
1      0.809651  1.389190 -0.107762  2.847250
2      0.809651  0.468899  0.328707  0.470184
3     -1.234923  0.413681 -0.549855  0.311713
4      0.809651  0.468899  0.684181 -0.322172
```

Αποτελέσματα σε Training Set (scaled data)

```
#Create an instance of Logistic regression named lgstc_reg on scaled data
lgstc_reg = LogisticRegression(penalty="none", solver="newton-cg")

# Fit logistic regression to training set
lgstc_reg.fit(X_train, y_train) # fit training data on logistic regression

print(lgstc_reg.intercept_, lgstc_reg.coef_) # get the coefficients of each features

[1.4162429] [[ 0.14531037  0.03366005 -0.32404502  0.36315462]]
```

$X_1 = \text{Home Ownership}$

$X_2 = \text{Income}$

$X_3 = \text{Debt to income ratio}$

$X_4 = \text{Credit score}$

$$Y = 1.416 + 0.145X_1 + 0.033X_2 - 0.324X_3 + 0.363X_4$$

$$\text{Probability of a good loan} = \frac{1}{1+e^{-Y}}$$

π.χ. Για το πρώτο δάνειο η πιθανότητα να είναι καλό είναι $\frac{1}{1+e^{-Y}}$, όπου $Y = 1.416 + 0.145 * 0.810 + 0.033 * (-0.556) - 0.324 * 0.053 + 0.363 * (-0.164) = 1.439$, άρα $\text{Probability of a good loan} = \frac{1}{1+e^{-Y}} = 0.808$.

Υπολογίζουμε τη συνάρτηση κόστους για το training, validation και test sets.

$$\frac{1}{n} \left[- \sum_{\text{Positive Outcomes}} \ln(Q) - \sum_{\text{Negative Outcomes}} \ln(1 - Q) \right]$$

Unscaled data

```
# y_train_pred, y_val_pred, and y_test_pred are the predicted probabilities for the training set
# validation set and test set using the fitted Logistic regression model

y_train_pred=lgstc_reg.predict_proba(X_train_unscaled)
y_val_pred=lgstc_reg.predict_proba(X_val_unscaled)
y_test_pred=lgstc_reg.predict_proba(X_test_unscaled)

# Calculate maximum likelihood for training set, validation set, and test set

mle_vector_train = np.log(np.where(y_train == 1, y_train_pred[:,1], y_train_pred[:,0])) # if y_train == 1 set y_tr
mle_vector_val = np.log(np.where(y_val == 1, y_val_pred[:,1], y_val_pred[:,0]))
mle_vector_test = np.log(np.where(y_test == 1, y_test_pred[:,1], y_test_pred[:,0]))

# Calculate cost functions from maximum Likelihoods

cost_function_training=np.negative(np.sum(mle_vector_train)/len(y_train))
cost_function_val=np.negative(np.sum(mle_vector_val)/len(y_val))
cost_function_test=np.negative(np.sum(mle_vector_test)/len(y_test))

print('cost function training set =', cost_function_training)
print('cost function validation set =', cost_function_val)
print('cost function test set =', cost_function_test)

cost function training set = 0.49111475922103454
cost function validation set = 0.4861711930624396
cost function test set = 0.4847008607351808
```

Scaled data

```
]# y_train_pred, y_val_pred, and y_test_pred are the predicted probabilities for the training set
# validation set and test set using the fitted Logistic regression model

y_train_pred=lgstc_reg.predict_proba(X_train)
y_val_pred=lgstc_reg.predict_proba(X_val)
y_test_pred=lgstc_reg.predict_proba(X_test)

# Calculate maximum likelihood for training set, validation set, and test set

mle_vector_train = np.log(np.where(y_train == 1, y_train_pred[:,1], y_train_pred[:,0]))
mle_vector_val = np.log(np.where(y_val == 1, y_val_pred[:,1], y_val_pred[:,0]))
mle_vector_test = np.log(np.where(y_test == 1, y_test_pred[:,1], y_test_pred[:,0]))

# Calculate cost functions from maximum Likelihoods

cost_function_training=np.negative(np.sum(mle_vector_train)/len(y_train))
cost_function_val=np.negative(np.sum(mle_vector_val)/len(y_val))
cost_function_test=np.negative(np.sum(mle_vector_test)/len(y_test))

print('cost function training set =', cost_function_training)
print('cost function validation set =', cost_function_val)
print('cost function test set =', cost_function_test)

cost function training set = 4.72569039145501
cost function validation set = 4.7284436817084305
cost function test set = 4.729486296578716
```

π.χ. Οι συναρτήσεις κόστους για τα training και validation sets δείχνουν ότι το μοντέλο γενικεύεται καλά. Αυτό δε σημαίνει απαραίτητα ότι είναι το καλύτερο μοντέλο. Με την προσθήκη νέων χαρακτηριστικών μπορείτε να καταλήξετε σε καλύτερο μοντέλο.

Decision Criterion

- Το dataset είναι imbalanced με περισσότερα καλά δάνεια (good loans) από δάνεια που δεν πληρώνουν (defaulting loans).
- Υπάρχουν διαδικασίες για τη δημιουργία ενός balanced dataset.
- Με ένα balanced data set θα μπορούσαμε να ταξινομήσουμε μια παρατήρηση ως θετική αν $Q > 0.5$ και αρνητική αν $Q \leq 0.5$.
- Ωστόσο, αυτό δεν λαμβάνει υπόψη το κόστος της εσφαλμένης ταξινόμησης ενός επισφαλούς δανείου και το διαφυγόν κέρδος από την εσφαλμένη ταξινόμηση ενός καλού δανείου
- Μια καλύτερη προσέγγιση είναι η διερεύνηση διαφορετικών thresholds Z
 - Αν $Q > Z$ τότε δεχόμαστε το δάνειο.
 - Αν $Q \leq Z$ τότε απορρίπτουμε το δάνειο.

The Confusion matrix and common ratios (αφορούν το test set)

Confusion matrix :

Δείχνει τη σχέση Predictions - outcomes

	Predict positive outcome	Predict negative outcome
Outcome positive	TP (True Positive)	FN (False Negative)
Outcome negative	FP (False Positive)	TN (True Negative)

Common ratios:

- **Accuracy** = $\frac{TP+TN}{TP+FN+FP+TN}$. Δείχνει το ποσοστό των παρατηρήσεων που ταξινομούνται σωστά.
- **True Positive Rate** (sensitivity or recall) = $\frac{TP}{TP+FN}$. Δείχνει το ποσοστό των θετικών αποτελεσμάτων που προβλέπονται σωστά.
- **True Negative rate** (specificity) = $\frac{TN}{TN+FP}$. Δείχνει το ποσοστό των αρνητικών αποτελεσμάτων που προβλέπονται ως αρνητικά.
- **False Positive Rate** = $\frac{FP}{TN+FP}$. Δείχνει το ποσοστό των αρνητικών αποτελεσμάτων που ταξινομήθηκαν λανθασμένα.
- **Precision, P** = $\frac{TP}{TP+FP}$. Δείχνει το ποσοστό των θετικών αποτελεσμάτων που αποδείχθηκαν σωστά.
- **F score** = $2 \times \frac{P \times TPR}{P + TPR}$. Είναι ένα accuracy μέτρο για imbalanced dataset και δείχνει πόσο καλά τα θετικά αποτελέσματα έχουν αναγνωρισθεί.

Αποτελέσματα στο test set

Z = 0.00:

	Predict no default	Predict default
Outcome positive (no default)	79.17%	0.00%
Outcome negative (default)	20.83%	0.00%

Z = 0.75:

	Predict no default	Predict default
Outcome positive (no default)	60.83%	18.34%
Outcome negative (default)	11.70%	9.13%

Z=0.80:

	Predict no default	Predict default
Outcome positive (no default)	42.71%	36.46%
Outcome negative (default)	6.46%	14.37%

Z=0.85:

	Predict no default	Predict default
Outcome positive (no default)	22.75%	56.42%
Outcome negative (default)	3.01%	17.82%

Z = 1.00:

	Predict no default	Predict default
Outcome positive (no default)	0.00%	79.17%
Outcome negative (default)	0.00%	20.83%

```

: THRESHOLD = [0.00, .75, .80, .85, 1.00]
# Create dataframe to store results
results = pd.DataFrame(columns=["THRESHOLD", "accuracy", "true pos rate", "true neg rate", "false pos rate", "p

# Create threshold row
results['THRESHOLD'] = THRESHOLD

j = 0

# Iterate over the 3 thresholds

for i in THRESHOLD:

    #lgstc_reg.fit(X_train, y_train)

    # If prob for test set > threshold predict 1
    preds = np.where(lgstc_reg.predict_proba(X_test)[: ,1] > i, 1, 0)

    # create confusion matrix
    cm = (confusion_matrix(y_test, preds, labels=[1, 0], sample_weight=None) / len(y_test))*100

    print('Confusion matrix for threshold =',i)
    print(cm)
    print(' ')
    
```

```

Confusion matrix for threshold = 0.0
[[79.17030568  0.         ]
 [20.82969432  0.         ]]
    
```

```

Confusion matrix for threshold = 0.75
[[60.82969432 18.34061135]
 [11.70305677  9.12663755]]
    
```

```

Confusion matrix for threshold = 0.8
[[42.70742358 36.4628821 ]
 [ 6.4628821  14.36681223]]
    
```

```

Confusion matrix for threshold = 0.85
[[22.7510917 56.41921397]
 [ 3.01310044 17.81659389]]
    
```

```

Confusion matrix for threshold = 1.0
[[ 0.         79.17030568]
 [ 0.         20.82969432]]
    
```

Test Set Ratios για διαφορετικές τιμές του Z

```

for i in THRESHOLD:

    #lgstc_reg.fit(X_train, y_train)

    # If prob for test set > threshold predict 1
    preds = np.where(lgstc_reg.predict_proba(X_test)[:,:1] > i, 1, 0)

    # create confusion matrix
    cm = (confusion_matrix(y_test, preds, labels=[1, 0], sample_weight=None) / len(y_test))*100

    print('Confusion matrix for threshold =',i)
    print(cm)
    print(' ')

    TP = cm[0][0]
    FN = cm[0][1]
    FP = cm[1][0]
    TN = cm[1][1]

    results.iloc[j,1] = accuracy_score(y_test, preds)
    results.iloc[j,2] = recall_score(y_test, preds)
    results.iloc[j,3] = TN/(FP+TN)
    results.iloc[j,4] = FP/(FP+TN)
    results.iloc[j,5] = precision_score(y_test, preds)
    results.iloc[j,6] = f1_score(y_test, preds)

    j += 1

print('ALL METRICS')
print( results.T)

```

	Z=0.00	Z = 0.75	Z = 0.80	Z = 0.85	Z=1.00
Accuracy	79.17%	69.96%	57.07%	40.57%	20.83%
True Positive Rate	100.00%	76.83%	53.94%	28.74%	0.00%
True Negative Rate	0.00%	43.82%	68.97%	85.53%	100.00%
False Positive Rate	100.00%	56.18%	31.03%	14.47%	0.00%
Precision	79.17%	83.87%	86.86%	88.31%	n.a.
F-score	88.37%	80.20%	66.55%	43.36%	n.a.

```

ALL METRICS
      0      1      2      3      4
THRESHOLD  0.0  0.75  0.8  0.85  1.0
accuracy  0.791703 0.699563 0.570742 0.405677 0.208297
true pos rate  1.0  0.76834 0.539437 0.287369 0.0
true neg rate  0.0 0.438155 0.689727 0.855346 1.0
false pos rate  1.0 0.561845 0.310273 0.144654 0.0
precision  0.791703 0.838651 0.868561 0.883051 0.0
f-score    0.883744 0.801957 0.665532 0.433625 0.0

```

- Υπάρχει ένα **trade-off** μεταξύ των **True Positive Rate** - **False Positive Rate**:

Όταν μεγαλώνει το **True Positive Rate** μεγαλώνει και το **False Positive Rate**.

Στο παράδειγμά μας αυτό σημαίνει ότι μπορούμε να αναγνωρίσουμε ένα μεγαλύτερο ποσοστό καλών δανείων μόνο όταν ταξινομούμε λάθος ένα μεγαλύτερο ποσοστό κακών δανείων.

- Η **ROC (receiver operating characteristics)** curve δείχνει τη σχέση αυτή.
- Area Under Curve (AUC)** Η area under the curve είναι ένας δημοφιλής τρόπος περίληψης της προγνωστικής ικανότητας ενός μοντέλου να εκτιμά μια δυαδική μεταβλητή (binary variable).
 - Όταν $AUC = 1$ τότε το μοντέλο είναι τέλειο (perfect).
 - Όταν $AUC > 0.5$ τότε το μοντέλο έχει (κάποια) προγνωστική ικανότητα
 - Όταν $AUC = 0.5$ τότε το μοντέλο δεν έχει προγνωστική ικανότητα (no predictive ability).
 - Όταν $AUC < 0.5$ τότε το μοντέλο έχει αρνητική προγνωστική ικανότητα (negative predictive ability).

Για το LendingClub model με τα 4 χαρακτηριστικά (features), $AUC = 0.6578$.

```
# Calculate the receiver operating curve and the AUC measure
```

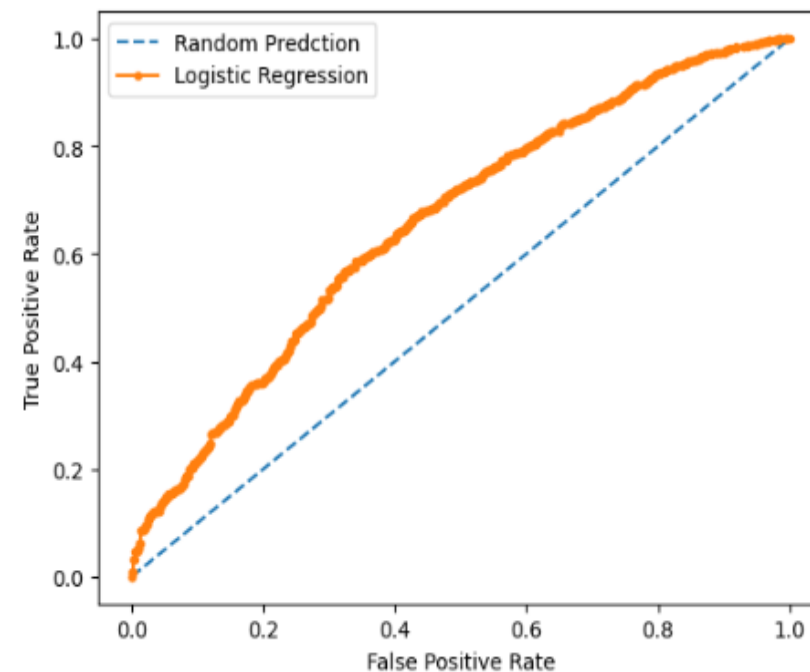
```
lr_prob=lgstc_reg.predict_proba(X_test)
lr_prob=lr_prob[:, 1]
ns_prob=[0 for _ in range(len(y_test))]
ns_auc=roc_auc_score(y_test, ns_prob)
lr_auc=roc_auc_score(y_test,lr_prob)
print("AUC random predictions =", ns_auc)
print("AUC predictions from logistic regression model =", lr_auc)
ns_fpr,ns_tpr,_=roc_curve(y_test,ns_prob)
lr_fpr,lr_tpr,_=roc_curve(y_test,lr_prob)

plt.plot(ns_fpr,ns_tpr,linestyle='--',label='Random Prediction')
plt.plot(lr_fpr,lr_tpr,marker='.',label='Logistic Regression')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

AUC random predictions = 0.5

AUC predictions from logistic regression model = 0.6577628841779786



Επιλέγοντας το Z

- Η τιμή του Z μπορεί να βασίζεται
 - Στο αναμενόμενο κέρδος από ένα δάνειο που είναι καλό, P
 - Η αναμενόμενη απώλεια από ένα δάνειο που αθετείται, L
 - Θέλουμε να μεγιστοποιήσουμε το $(P \times \text{True Positive}) - (L \times \text{False Positive})$
 - Αν θεωρήσουμε ότι το κόστος των δανείων που αθετούνται να είναι 4 φορές το κέρδος των καλών δανείων, τότε για
 - Για $Z = 0.75$ προκύπτει $0.1402P$ ($=P \times 60.83\% - 4P \times 11.70\%$)
 - Για $Z=0.80$ προκύπτει $0.1686P$ ($=P \times 42.71\% - 4P \times 6.46\%$)
 - Για $Z = 0.85$ προκύπτει $0.1070P$ ($=P \times 22.75\% - 4P \times 3.01\%$)
- Άρα επιλέγουμε το $Z=0.80$ επειδή είναι το πιο κερδοφόρο.