Bankruptcy Prediction

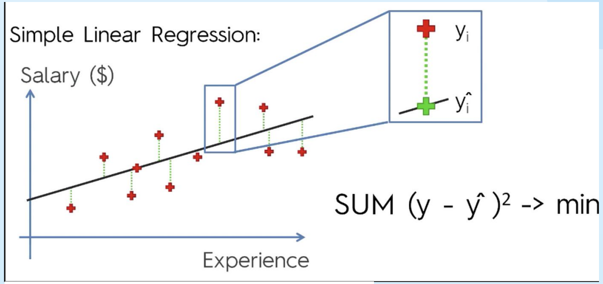
Machine Learning Models

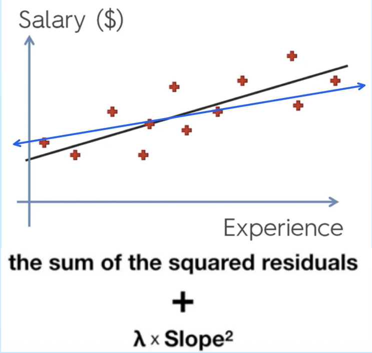
Team 1: Sirine and Salim

Contents

RidgeClassifier

- A Ridge Classifier is a linear classifier model used for classification tasks.
- Prevents overfitting: Especially beneficial for datasets with high dimensionality.
- The bias added to the model is also known as the Ridge Regression penalty.
- The parameter alpha (lambda) controls the strength of the penalty.





CategoricalNB

- Designed specifically to handle features that represent distinct categories, not numerical values. Examples: (red, green, blue) or text classifications (spam, not spam).
- CategoricalNB is specifically designed for features with unordered categories (ex: 1, 2, 3...)
- In practice, the normalization term P(B) is often ignored because it doesn't affect the class comparison.

Probability of B occurring given evidence A has already occurred

Probability of A occurring

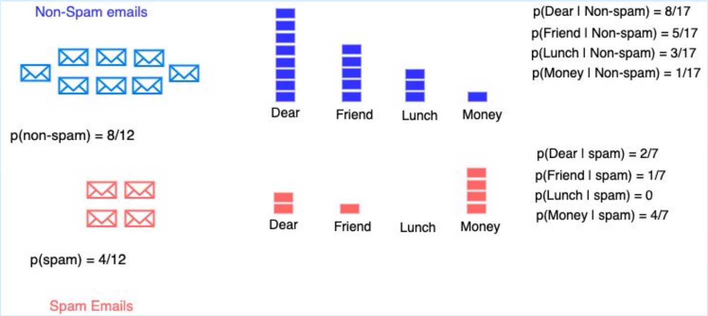
$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Probability of A occurring given evidence B has already occurred

Probability of B occurring

$$P(B|A) = P(b_1|A) \cdot P(b_2|A) \cdot \ldots \cdot P(b_n|A)$$

CategoricalNB



Dear Friend

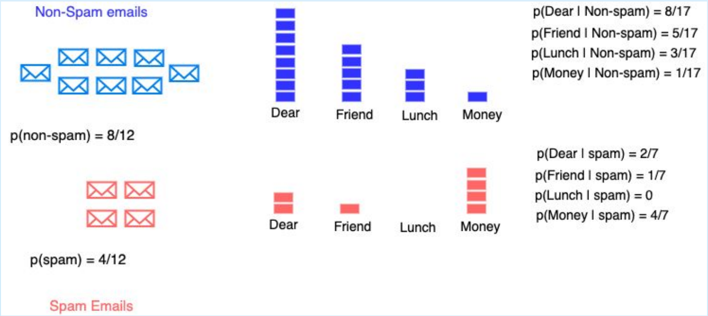
Not Spam

$$0.67 \times 0.47 \times 0.29 = 0.09$$

$$0.33 \times 0.29 \times 0.14 = 0.01$$

Spam

CategoricalNB



NOT SPAM

Not Spam

$$0.67 \times 0.47 \times 0.29 = 0.09$$

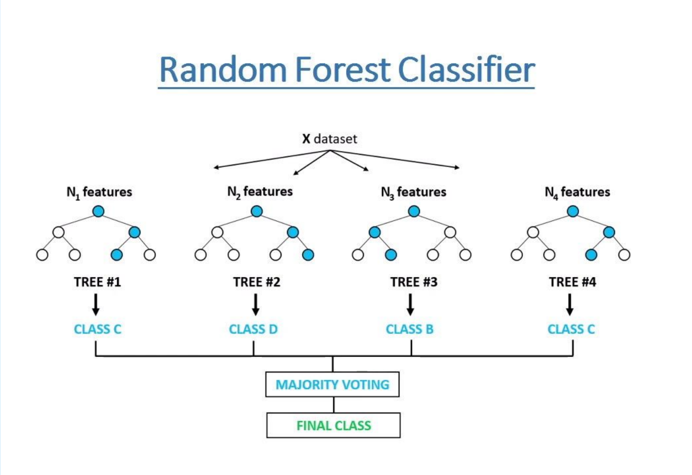
$$0.33 \times 0.29 \times 0.14 = 0.01$$

Spam

ExtraTreeClassifier

RandomForest

- Selects a random subset of features at each split
- Split thresholds are determined based on the best split among the randomly selected features.

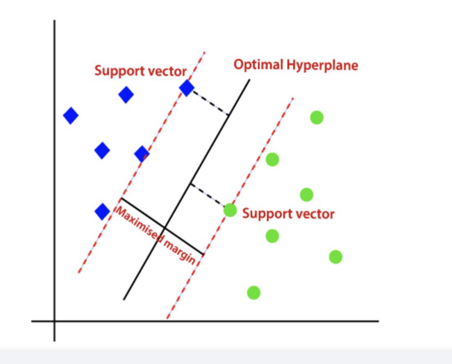


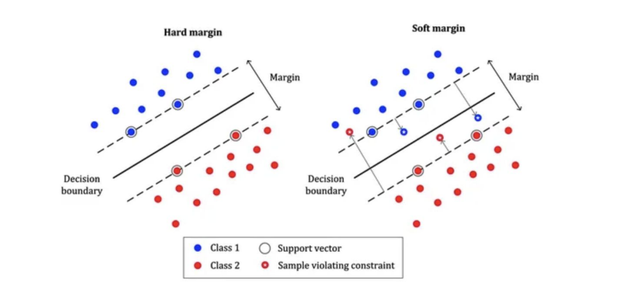
ExtraTreeClassifier

- Creates many decision trees
- Sampling for each tree is random
- Creates a dataset for each tree with unique samples
- Select k features for each split
- Choosing the split thresholds randomly

Support Vector Classifier

- Supervised machine learning problem where we try to find a **hyperplane** that best separates the two classes.
- SVM works best when the dataset is small and complex
- **Linear** SVM vs **Non-Linear** SVM (**kernel tricks**)
- **Support Vectors:** points that are closest to the hyperplane. Separating line will be defined with the help of these data points.
- **Margin:** distance between the hyperplane and the support vectors. In SVM large margin is considered a good margin.
- Two types of margins: **hard margin** and **soft margin**



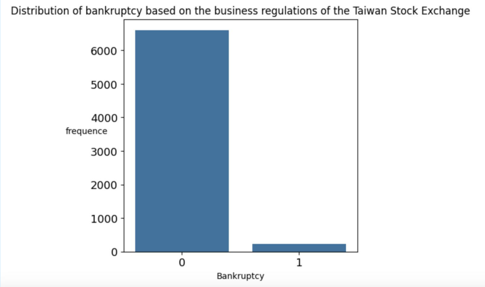


Support Vector Classifier: pre work

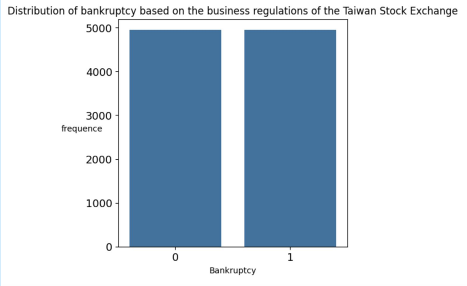
Select Feature : Kbest

	score	features
15	2.901545e+11	Cash/Current Liability
34	2.495621e+11	Fixed Assets to Assets
50	2.421293e+11	Net Value Growth Rate
33	2.200229e+11	Fixed Assets Turnover Frequency
83	8.633138e+10	Revenue per person
88	6.391313e+10	Total assets to GNP price
73	3.059892e+10	Quick Ratio
70	2.391564e+10	Quick Asset Turnover Rate
85	2.051825e+10	Total Asset Growth Rate
80	1.385736e+10	Research and development expense rate
13	7.747258e+09	Cash Turnover Rate
89	6.638710e+09	Total debt/Total net worth
20	6.456970e+09	Current Asset Turnover Rate
38	2.584786e+09	Interest-bearing debt interest rate
4	1.959675e+09	Average Collection Days

Balancing DataSet : Resample



oversampling



(5114, 16)

(9896, 16)

Support Vector Classifier

- **Results:**

```
'Confusion matrix:'

array([[766, 211],
       [195, 809]])


Precission Score: 0.7931372549019607
Recall Score: 0.8057768924302788
F1 Score: 0.799407114624506
Accuracy Score: 0.7950530035335689
```

Logistic Regression

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	N cre Interv
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.6228
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.6236
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.6238
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.6229
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.6235

5 rows × 96 columns

- 96 columns
- No null values

Feature Selection

```
selected_features = ['_interest-bearing_debt_interest_rate',
          '_persistent_eps_in_the_last_four_seasons',
          '_per_share_net_profit_before_tax_(yuan_¥)',
          '_net_value_growth_rate',
          '_quick_ratio',
          '_interest_expense_ratio',
          '_net_worth/assets',
          '_borrowing_dependency',
          '_net_profit_before_tax/paid-in_capital',
          '_accounts_receivable_turnover',
          '_fixed_assets_turnover_frequency',
          '_cash/total_assets',
          '_cash/current_liability',
          '_working_capital/equity',
          '_net_income_to_total_assets',
          "_net_income_to_stockholder's_equity",
          '_degree_of_financial_leverage_(dfl)',
          '_equity_to_liability', 'bankrupt?']
```

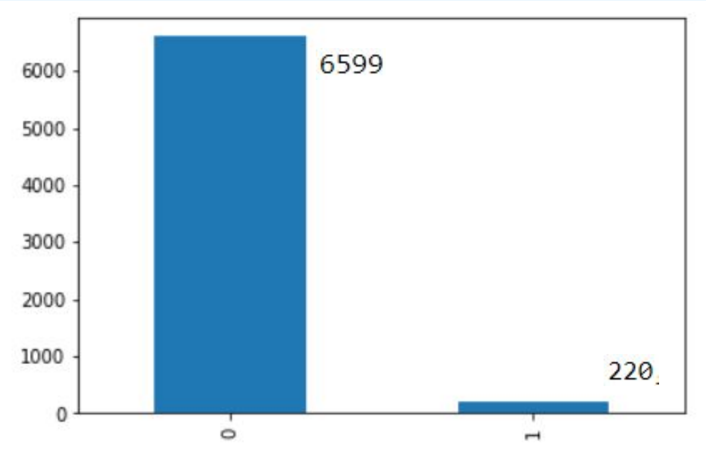
- RFECV
- The importance scores of each feature are determined based on the **fitted model**
- Drop the least important features
- On first try, i got 68 features

Data Imbalance

SMOTE

```
X_train_SMOTE.shape, y_train_SMOTE.shape
```

```
((9890, 18), (9890,))
```



Metrics and Confusion Matrix

```
Confusion Matrix:
[[1373  281]
 [  31   20]]
```

precision: 0.0664451827242525
recall: 0.39215686274509803
accuracy: 0.8170087976539589
f1: 0.11363636363636363

TN FP
FN TP

Precision = TP / (TP + FP)
low proportion of correctly predicted positive instances among all predicted positives

model captures around 39.22% of the actual positive instances.
Recall = TP / (TP + FN)

overall proportion of correct predictions.
Accuracy = (TP + TN) / (TP + TN + FP + FN)

F1 = 2 * (Precision * Recall) / (Precision + Recall)

Models Comparison

Using Kbest and Resample

SVC

```
'Confusion matrix:'

array([[766, 211],
       [195, 809]])


Precission Score: 0.7931372549019607
Recall Score: 0.8057768924302788
F1 Score: 0.799407114624506
Accuracy Score: 0.7950530035335689
```

Logistic Regression

```
'Confusion matrix:'

array([[721, 256],
       [521, 483]])


Precission Score: 0.6535859269282814
Recall Score: 0.4810756972111554
F1 Score: 0.5542168674698794
Accuracy Score: 0.607773851590106
```

Using RFECV and SMOTE

Logistic Regression

```
Confusion Matrix:
[[1373  281]
 [  31   20]]
precision:  0.0664451827242525
recall:  0.39215686274509803
f1:  0.11363636363636363
accuracy:  0.817008797653959
```

TN FP
FN TP

Thank You !