3.2 - Linear and Logistic Regression

Giovanni Della Lunga
giovanni.dellalunga@unibo.it

Introduction to Machine Learning for Finance

Bologna - February-March, 2026

Lesson 3.1 – Linear and Logistic Regression

Positioning of the lecture

This lecture introduces two fundamental supervised learning models used across *banking and quantitative finance*:

- **Linear regression** (continuous targets)
- **Logistic regression** (binary outcomes / probabilities)

They are not presented as "simple baselines" only, but as **core tools** for:

- model interpretability and governance,
- robust decision support under uncertainty,
- benchmarking more complex ML models.

Why Linear and Logistic Models Matter in Finance

In many financial applications, the key requirements are:

- **Interpretability**: explain decisions to stakeholders and regulators
- **Stability**: avoid fragile models under noise and data shifts
- **Calibrated outputs**: probabilities and scores that can drive policies
- **Robust validation**: reliable out-of-sample evaluation

Typical domains (non-exhaustive):

- **Credit risk** and scoring
- **Fraud detection** and AML screening
- **Risk management** (early warning indicators, stress testing)
- **Pricing / forecasting** in retail and corporate banking
- **Trading** (as a special case with additional challenges)

Learning Objectives

By the end of this lecture, you should be able to:

1. Formulate a supervised learning problem as **regression** or **classification**
2. Fit and interpret a **linear regression** model, and understand the role of **MSE**
3. Explain why **regularization** (Ridge, Lasso, Elastic Net) improves stability in high-dimensional settings
4. Derive the **logistic regression** model as a probability model:

$$\hat{p}(y = 1 \mid x) = \sigma(w^\top x + b)$$

5. Explain the meaning of the **classification threshold** and why it is a **policy choice**
6. Evaluate classification models beyond accuracy using: **confusion matrix, precision, recall, F1-score, ROC curve, AUC**
7. Connect model outputs to **decision-making** via asymmetric costs (e.g., false positives vs. false negatives in credit and fraud)

How This Lecture Fits the Course

This lecture builds directly on concepts already introduced:

- **Data preprocessing** and feature representation
- **Training / validation / test** and out-of-sample evaluation
- **Bias–variance trade-off** and overfitting
- **Curse of dimensionality** and the need for regularization

and prepares the ground for upcoming supervised models:

- **Decision Trees** (interpretability with higher variance)
- **Ensembles** (variance reduction / bias reduction)
- **Support Vector Machines** (margin-based generalization)

Back to Linear Regression

Linear Regression

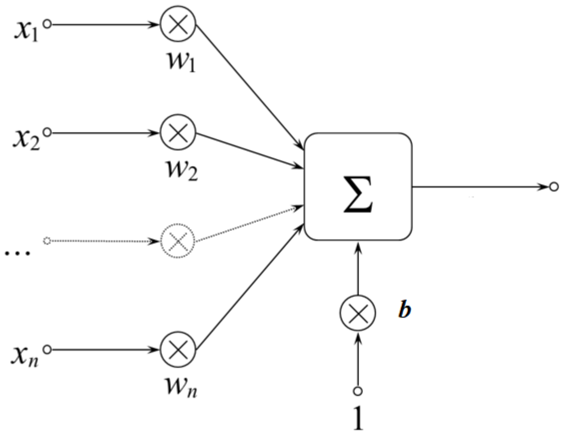
A linear model makes a prediction by simply computing a weighted sum of the input features, plus a constant called the **bias** term (also called the **intercept** term):

$$y = b + w_1 X_1 + w_2 X_2 + \cdots + w_m X_m + \epsilon \quad (1)$$

where:

- y is the predicted value (the value of the target);
- m is the number of features;
- X_i is the i^{th} feature value that are used to predict y;
- b and w_j are the j^{th} model parameters (b being the **bias** term and w_i the **weights**)
- ϵ is the predicton error.

Linear Regression



$$y = \sum_{i=1}^{n} w_i x_i + b$$

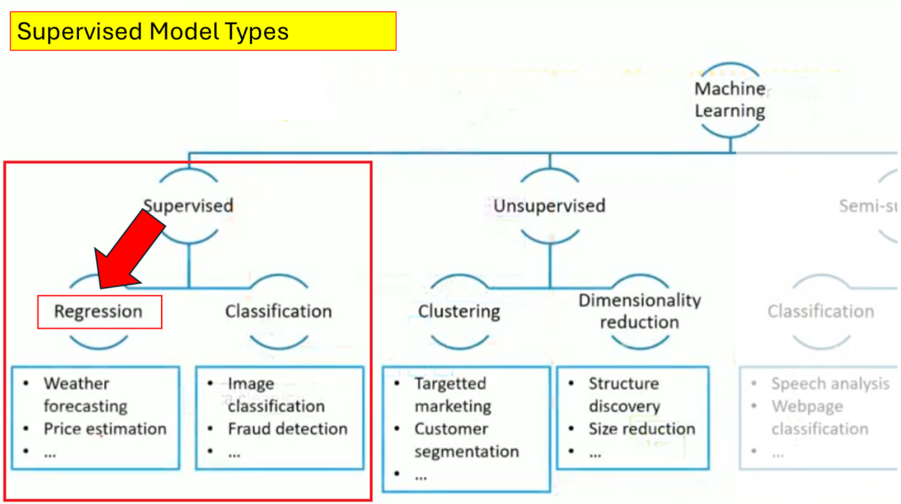
Linear Regression

- The parameters b and w_i are chosen to minimize the mean squared error over the training data set.
- This means that the task in linear regression is to find values for b and w_i that minimize

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y - b - w_1 X_{i1} - w_2 X_{i2} - \cdots - w_m X_{im})^2 \tag{2}$$

where n is the size of the training set.

Linear Regression



Iowa House Price Case Study

Linear Regression

The **Iowa House Pricing Dataset** from Kaggle is a popular dataset used for regression problems, particularly in the context of machine learning. It contains comprehensive information on houses in Ames, Iowa, and their sale prices, and it was designed to serve as an alternative to the Boston Housing Dataset. Here's a brief overview: **Dataset Overview**

- **Target Variable**: 'SalePrice' – the final price of the house in USD.
- **Number of Rows (Observations)**: 2,930 (combined training and test datasets).
- **Number of Features (Columns)**: 79 explanatory variables plus the target variable.

Iowa House Price Case Study

Linear Regression

- The objective is to predict the prices of house in Iowa from features
- 800 observations in training set, 600 in validation set, and 508 in test set
- Here the original competition description: **https://www.kaggle.com/c/house-prices-advanced-regression-techniques**

Types of Features The dataset includes a mix of:

- 1. **Numerical Features**:
  - Continuous (e.g., ‘LotArea‘, ‘GrLivArea‘, ‘SalePrice‘)
  - Discrete (e.g., ‘GarageCars‘, ‘TotRmsAbvGrd‘)
- 2. **Categorical Features**:
  - Nominal (e.g., ‘Neighborhood‘, ‘HouseStyle‘)
  - Ordinal (e.g., ‘ExterQual‘, ‘KitchenQual‘)
- 3. **Temporal Features**:
  - Year-based (e.g., ‘YearBuilt‘, ‘YrSold‘)
- 4. **Location Features**:
  - Specific location details (e.g., ‘Neighborhood‘, ‘MSSubClass‘).

Iowa House Price Case Study

Linear Regression

Categorical Features

In this problem one of the categorical features is concerned with the basement quality as indicated by the ceiling height. The categories are:

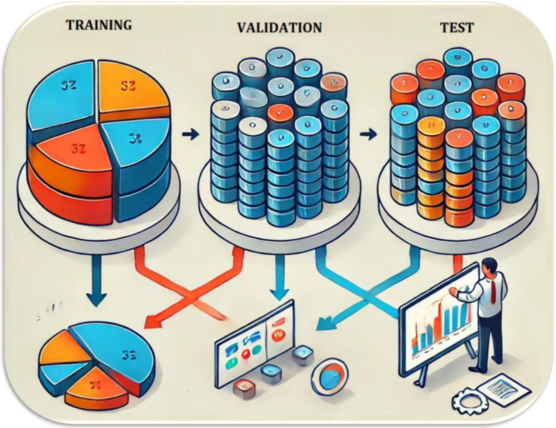
- Excellent (¡ 100 inches)
- Good (90-99 inches)
- Typical (80-89 inches)
- Fair (70-79 inches)
- Poor (¡ 70 inches)
- No Basement

This is an example of a categorical variable where **there is a natural ordering**. We created a new variable that had a values of 5, 4, 3, 2, 1 and 0 for the above six categories respectively.

Iowa House Price Case Study

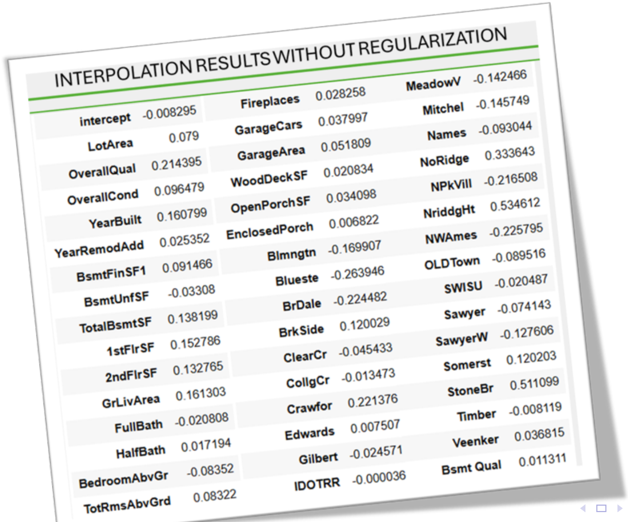
Linear Regression

Dataset splitting: Training, Validation and Test



Iowa House Price Results

Linear Regression



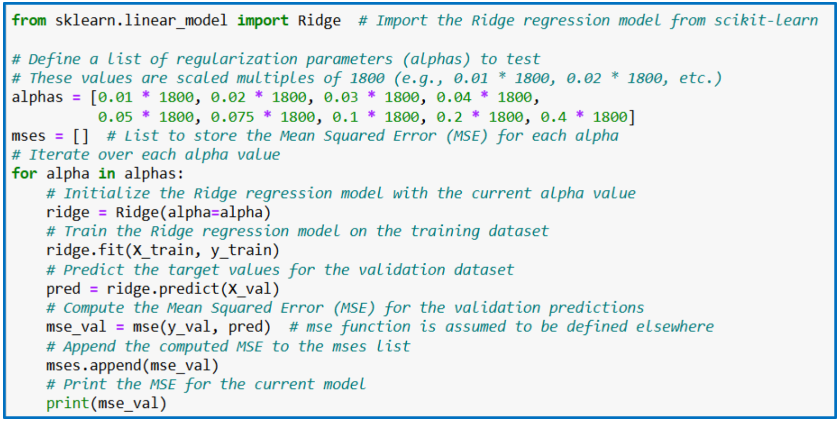
INTERPOLATION RESULTS WITHOUT REGULARIZATION

intercept	-0.008295	Fireplaces	0.028258	MeadowV	-0.142466
LotArea	0.079	GarageCars	0.037997	Mitchel	-0.145749
OverallQual	0.214395	GarageArea	0.051809	Names	-0.093044
OverallCond	0.096479	WoodDeckSF	0.020834	NoRidge	0.333643
YearBuilt	0.160799	OpenPorchSF	0.034098	NPkVill	-0.216508
YearRemodAdd	0.025352	EnclosedPorch	0.006822	NriddgHt	0.534612
BsmtFinSF1	0.091466	Blmngtn	-0.169907	NWAmes	-0.225795
BsmtUnfSF	-0.03308	Blueste	-0.263946	OLDTown	-0.089516
TotalBsmtSF	0.138199	BrDale	-0.224482	SWISU	-0.020487
1stFlrSF	0.152786	BrkSide	0.120029	Sawyer	-0.074143
2ndFlrSF	0.132765	ClearCr	-0.045433	SawyerW	-0.127606
GrLivArea	0.161303	CollgCr	-0.013473	Somerst	0.120203
FullBath	-0.020808	Crawfor	0.221376	StoneBr	0.511099
HalfBath	0.017194	Edwards	0.007507	Timber	-0.008119
BedroomAbvGr	-0.08352	Gilbert	-0.024571	Veenker	0.036815
TotRmsAbvGrd	0.08322	IDOTRR	-0.000036	Bsmt Qual	0.011311

Ridge Regression

Linear Regression

We try using Ridge regression with different values of the hyperparameter λ. The following code shows the effect of this parameter on the prediction error.



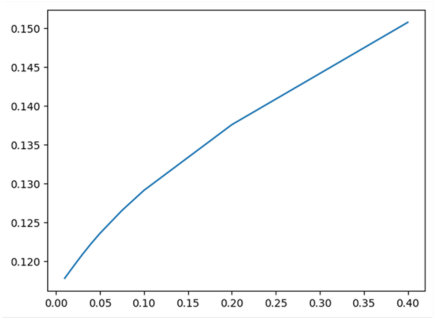
```
from sklearn.linear_model import Ridge  # Import the Ridge regression model from scikit-learn

# Define a list of regularization parameters (alphas) to test
# These values are scaled multiples of 1800 (e.g., 0.01 * 1800, 0.02 * 1800, etc.)
alphas = [0.01 * 1800, 0.02 * 1800, 0.03 * 1800, 0.04 * 1800,
          0.05 * 1800, 0.075 * 1800, 0.1 * 1800, 0.2 * 1800, 0.4 * 1800]
mses = []  # List to store the Mean Squared Error (MSE) for each alpha
# Iterate over each alpha value
for alpha in alphas:
    # Initialize the Ridge regression model with the current alpha value
    ridge = Ridge(alpha=alpha)
    # Train the Ridge regression model on the training dataset
    ridge.fit(X_train, y_train)
    # Predict the target values for the validation dataset
    pred = ridge.predict(X_val)
    # Compute the Mean Squared Error (MSE) for the validation predictions
    mse_val = mse(y_val, pred)  # mse function is assumed to be defined elsewhere
    # Append the computed MSE to the mses list
    mses.append(mse_val)
    # Print the MSE for the current model
    print(mse_val)
```

Ridge and Lasso Regression

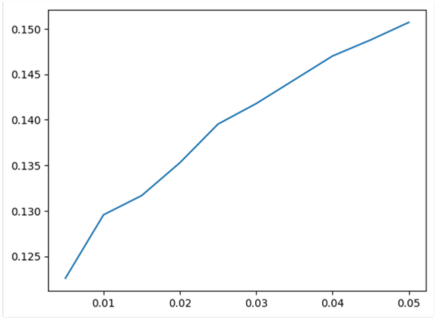
Linear Regression

Error analysis using Ridge and Lasso regression with different values of the hyperparameter λ

Ridge Regression



Lasso Regression

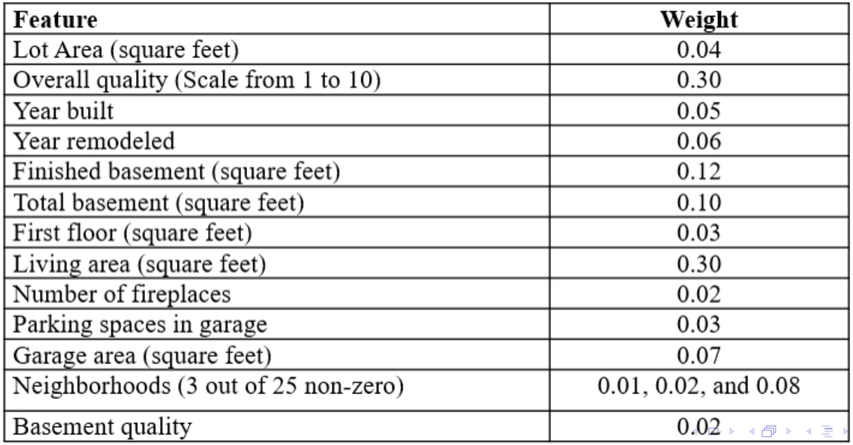


As expected the prediction error increases as λ increases. Values of λ in the range 0 to 0.1 might be reasonably be considered because prediction errors increases only slightly when λ is in this range. However it turns out that the improvement in the model is quite small for these values of λ.

Iowa House Price Results

Linear Regression

Non-zero weights for Lasso when $\lambda = 0.1$ (overall quality and total living area were most important)



Feature	Weight
Lot Area (square feet)	0.04
Overall quality (Scale from 1 to 10)	0.30
Year built	0.05
Year remodeled	0.06
Finished basement (square feet)	0.12
Total basement (square feet)	0.10
First floor (square feet)	0.03
Living area (square feet)	0.30
Number of fireplaces	0.02
Parking spaces in garage	0.03
Garage area (square feet)	0.07
Neighborhoods (3 out of 25 non-zero)	0.01, 0.02, and 0.08
Basement quality	0.02

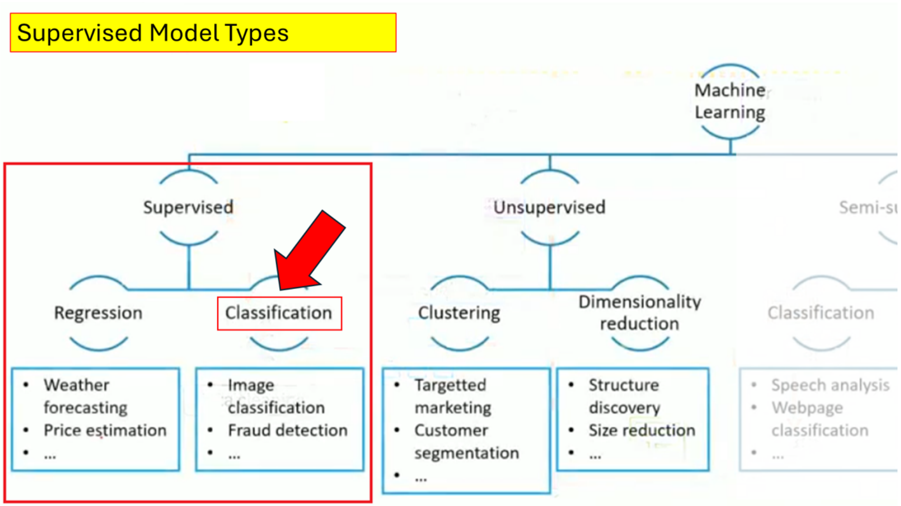
Summary of Iowa House Price Results

Linear Regression

- With no regularization correlation between features leads to some negative weights which we would expect to be positive
- Improvements from Ridge is modest
- Lasso leads to a much bigger improvement in this case
- Elastic net similar to Lasso in this case
- Mean squared error for test set for Lasso with $\lambda = 0.1$ is 14.7

Logistic Regression

Logistic Regression



Logistic Regression

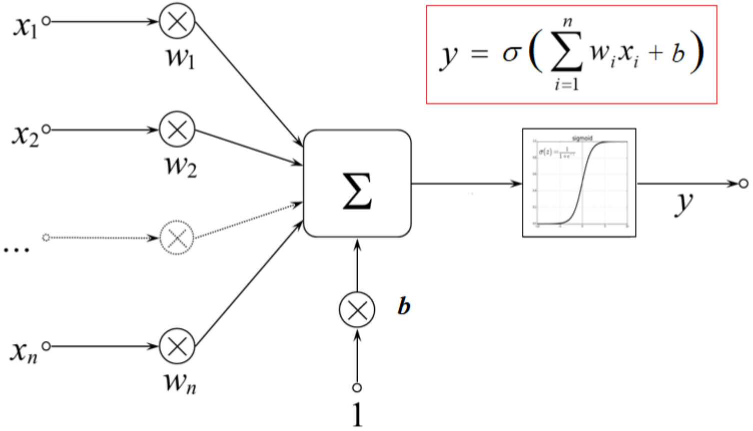
- The objective is to classify observations into a **positive outcome** and **negative outcome** using data on features
- Probability of a positive outcome is assumed to be a sigmoid function:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{3}$$

- where Y is related linearly to the values of the features:

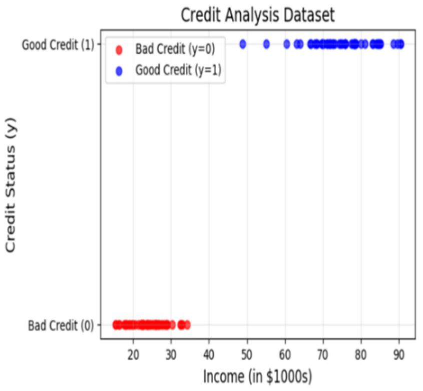
$$y = b + w_1 X_1 + w_2 X_2 + \cdots + w_m X_m \tag{4}$$

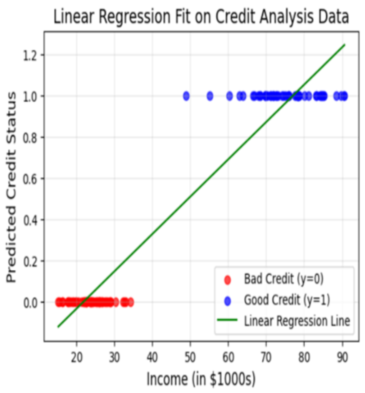
Logistic Regression



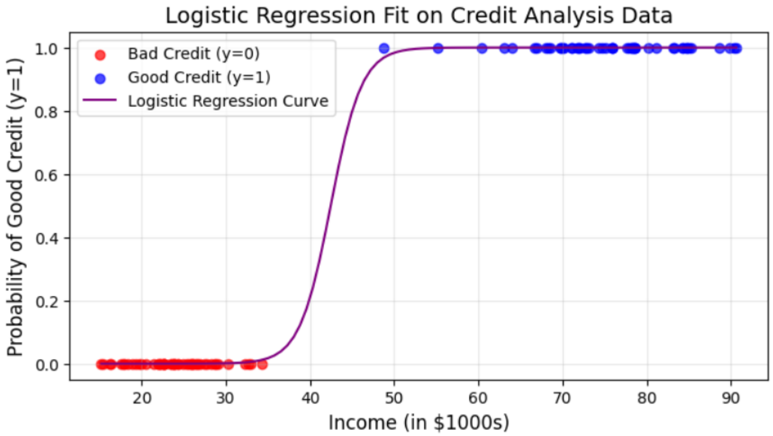
$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Logistic Regression





Logistic Regression



Logistic Regression

- The output of the sigmoid function is then interpreted as the probability of a particular example belonging to class 1, $\sigma(z) = P(y = 1|\mathbf{x}; \mathbf{w})$, given its features, x, parameterized by the weights, w.
- The predicted probability can then simply be converted into a binary outcome via a **threshold function**:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Cost Function

Logistic Regression

- For example, for a simple binary classification problem we can use the logistic loss function

$$L(y, \hat{y}) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})] \tag{5}$$

- y is the true label (0 or 1).
- $\hat{y}$ is the predicted probability of class 1.
- The function penalizes incorrect predictions more severely.

This cannot be maximized analytically but we can use a gradient ascent algorithm

Cost Function

Logistic Regression

$$L(y, \hat{y}) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})] \tag{6}$$

When $y = 1$:

$$L(1, \hat{y}) = -\log \hat{y}$$

- If $\hat{y}$ is close to 1, the loss is small.
- If $\hat{y}$ is close to 0, the loss is large.

When $y = 0$:

$$L(0, \hat{y}) = -\log (1 - \hat{y})$$

- If $\hat{y}$ is close to 0, the loss is small.
- If $\hat{y}$ is close to 1, the loss is large.

Why Accuracy Can Be Misleading

Logistic Regression

Definition of Accuracy:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{7}$$

Example:

- Suppose 95% of loans are **good**, and only 5% are **bad**.
- A naive model that **always predicts "good"** achieves **95% accuracy**.
- However, this model is completely **useless** for detecting bad loans.

Conclusion: Accuracy does not capture the impact of misclassification.

Why Accuracy Can Be Misleading

Why Accuracy can be Misleading in Credit Risk Models

- In **credit risk modeling**, we often predict whether a loan will **default** (bad loan) or not (good loan).
- The dataset is often **highly unbalanced**: good loans are much more frequent than bad loans.
- A simple **accuracy metric** can be misleading in such cases.

Accuracy Metrics

Threshold

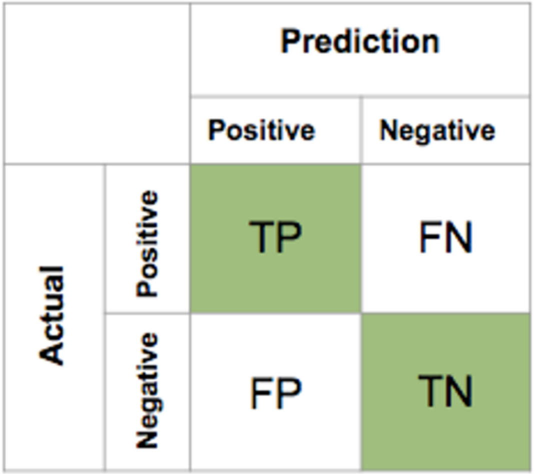
- The threshold is a value between 0.0 and 1.0 that serves as your "cutoff" for which predicted probabilities you want to consider a True or a False, a Yes or a No, a 1 or a 0.
- Who decides it? The threshold is externally determined, based on business, risk, and regulatory constraints.
- You may have assumed this threshold is naturally located right in the middle, at 0.5, but you can move that threshold.

Threshold

- Why would you want to do that?
- Reasons include:
- a) You are conservative about your guesses, so you set the threshold for a "Yes" to 0.7 (or 70%, if you will). Anything predicted to have less than a 70% probability is just too risky for you.
- b) Alternatively, a risk-taker may want to call anything over 0.35 probability a "Yes", so that they don‘t miss any opportunities.
- c) Lastly, perhaps you want to use the threshold that gives the highest performance, for whatever metric you choose.

Confusion Matrix

- A confusion matrix is $N * N$ dimension matrix wherein one axis represents **Actual** label while the other axis represents **Predicted** label.
- Confusion Matrix is the most intuitive and basic metric from which we can obtain various other metrics like precision, recall, accuracy, F1 score, AUC - ROC.



		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Confusion Matrix

- For a better understanding of what TP, FP, TN, and FN are, we will consider an example of: **If received mail is spam or ham.**
- **Positive** - Mail received is ham
- **Negative** - Mail received is spam
- $\Rightarrow$ **True Positive (TP)**: It represents the predicted label is positive and also actual label is positive - correctly predicted. Credit Example: Risky client correctly rejected.
- $\Rightarrow$ **True Negative (TN)**: It represents the predicted label is negative and also actual label is negative - correctly predicted. Credit Example: Good client correctly approved.
- $\Rightarrow$ **False Negative (FN)**: It represents the predicted label is negative but the actual label is positive - wrongly predicted. Credit Example: good loan rejected $\rightarrow$ opportunity cost.
- $\Rightarrow$ **False Positive (FP)**: It represents the predicted label is positive but the actual label is negative - wrongly predicted. Credit Example: Loan approved that defaults $\rightarrow$ direct loss

Precision

- **General Definition**: Precision measures what proportion of predicted positive label is actually positive.
- **Precision** can be expressed in terms of True Positive and False Positive:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- As ‘False Positive‘ decreases, our precision increases and vice-versa
- When to use Precision?
- Precision is used when we want to mostly focus on false-positive i.e to decrease false-positive value thereby increase precision value.
- A question might arise why we want to mostly focus on false-positive and not false-negative. The answer to this question depends on the context.

Recall/Sensitivity

- **General Definition**: Recall measures what proportion of actual positive label is correctly predicted as positive.
- **Precision vs Recall in Banking**:

  High Precision
  - few false alarms
  - important in fraud investigation to reduce operational costs

  High Recall
  - few missed risky cases
  - crucial in credit risk and AML screening

Recall/Sensitivity

- Recall in terms of True Positive and False Negative:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- From the above formula in the image, we can analyze that as ‘False Negative‘ decreases, our recall increases and vice-versa.
- Recall is used when we want to mostly focus on false-negative i.e to decrease false negative value thereby increase recall value.

F1-Score

- F1-score is another one of the good performance metrics which leverages both precision and recall metrics.
- F1-score can be obtained by simply taking 'Harmonic Mean' of precision and recall.
- Unlike precision which mostly focuses on false-positive and recall which mostly focuses on false-negative, **F1-score focuses on both false positive and false negative**.

F1-Score

- F1-score in terms of Precision and Recall;
- The F-Score is the Harmonic Mean of Precision and Recall:

$$F = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

- Alternatively:

$$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

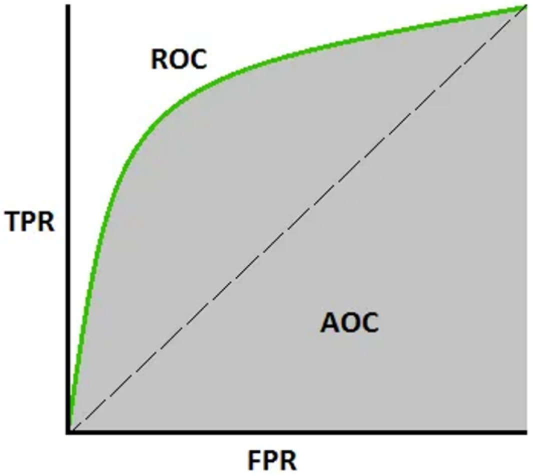
- When to use F1-score:
- As mentioned above, F1-score focuses on both false positive and false negative and try to decrease both false positive and false negative thereby increase F1-score.

AUC - ROC curve

- AUC - ROC is one of the most important performance metric used to check model performance.
- AUC - ROC is used for binary and also multi-class classification but mostly used in binary classification problems.
- In this lesson, we will consider a binary class classification.
- AUC-ROC is a graphical representation of model performance. ROC is a probability curve and AUC is the measure of separability.
- Depending on the threshold set, we can analyze how well our model has performed in separating two classes.
- Higher the AUC better is our model in separating two classes.

AUC - ROC curve

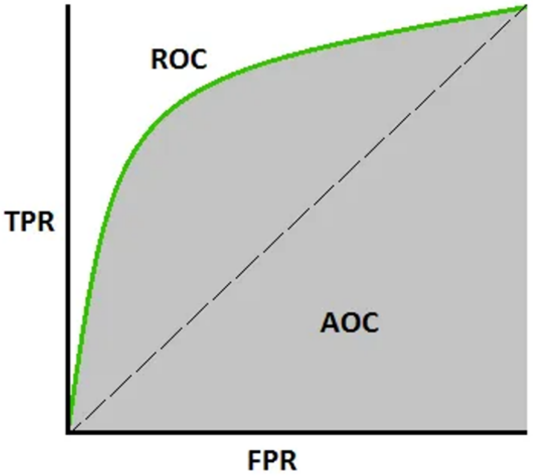
- Graphical representation of AUC - ROC
- Referring the image, we can see that AUC - ROC curve is plotted with FPR against TPR where FPR (False Positive Rate) is on X-axis while TPR (True Positive Rate) is on Y-axis.
- The green curve represents ROC curve while the area/region under ROC curve (green curve) represents AUC.



AUC - ROC curve

- **True Positive Rate (TPR)**: TPR is nothing but Recall / Sensitivity.
- The formula for TPR as follows:

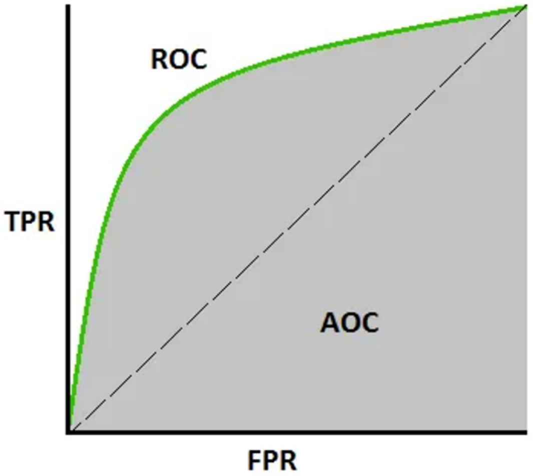
$$TPR = \frac{TP}{TP + FN}$$



AUC - ROC curve

- **False Positive Rate (FPR)**
- The formula for TPR as follows:

$$FPR = \frac{FP}{TN + FP}$$

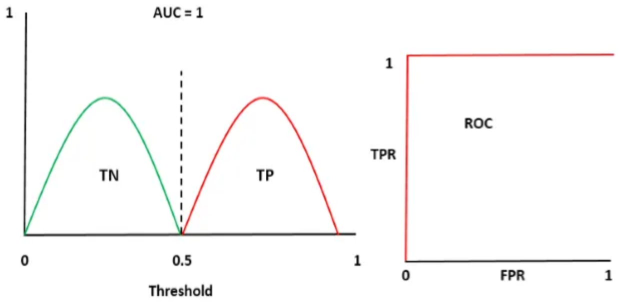


AUC - ROC curve

- Interpretation of AUC-ROC curve
- Let's now look into the analysis of binary class classification based on the AUC score and ROC curve...

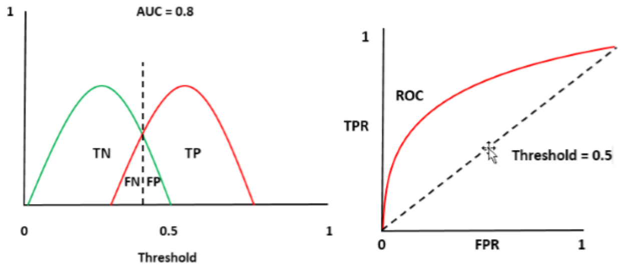
AUC - ROC curve

Threshold set to 0.5. There is no overlap between the two curves (green and red). This is the best model with AUC score of 1.0. This indicates that the probability of a model to separate positive and negative class is 1.0. In other words, we can say that there is 100% chance model can separate positive and negative class.



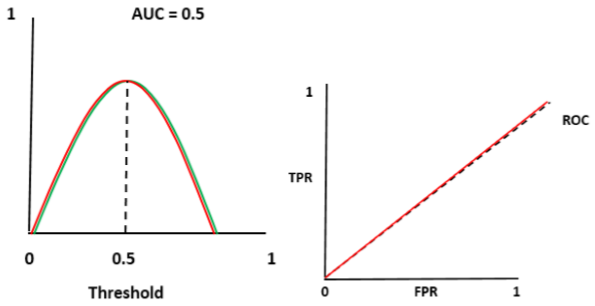
AUC - ROC curve

Threshold set to 0.5. There is a little bit of overlap between the two curves (green and red). This is a good model with AUC score of 0.8. This indicates that the probability of a model to separate positive and negative class is 0.8. In other words, we can say that there is 80% chance model can separate positive and negative class.



AUC - ROC curve

Threshold set to 0.5. We can see the full overlap between the two curves (green and red). This is a bad model with AUC score of 0.5. This indicates that the probability of a model to separate positive and negative class is 0.5. In other words, we can say that there is 50% chance model can separate positive and negative class.



Summary

- **Precision**: Precision measures what proportion of predicted positive label is actually positive. We mostly focus on false-positive value and try to decrease it to the least possible value thereby increase in precision value.
- **Recall**: Recall measures what proportion of actual positive label is correctly predicted as positive. We mostly focus on false-negative value and try to decrease it to the least possible value thereby increase in recall value.
- **F1-Score**: F1 Score is the 'Harmonic Mean' of precision and recall. We focus on both false-positive and false-negative and try to decrease both false-positive and false-negative thereby increase F1 Score.
- **AUC- ROC curve**: It is a graphical representation of ROC curve and region/area under curve i.e AUC. It is mostly used in binary class classification. It interprets the probability or percentage of separability of positive and negative classes. Higher the AUC - ROC, better is our model in separating positive and negative classes.

Key Takeaways

- **Accuracy alone is misleading** in unbalanced datasets.
- **Confusion matrix** provides deeper insights into model performance.
- **Precision, Recall, F1-score, and Specificity** are better suited for evaluating credit risk models.
- A strong credit risk model should minimize **False Positive**, as these represent loans that default but were wrongly classified as good.

Credit Risk Example

Lending Club Case Study

- Data consists of loans made and whether they proved to be good or defaulted.
- We use only four features
  - Home ownership (rent vs. own)
  - Income
  - Debt to income
  - Credit score
- Training set has 8,695 observations (7,196 good loans and 1,499 defaulting loans). Test set has 5,196 observations (4,858 good loans and 1,058 defaulting loans)

Interpretation of FICO Scores

- The **FICO** credit score is a three-digit number used by lenders to assess an individual's creditworthiness.
- It ranges from **300 to 850**, with higher scores indicating lower credit risk.
- Developed by **Fair Isaac Corporation (FICO)**.
- Used in the U.S. for lending decisions such as **mortgages, car loans, and credit cards**.

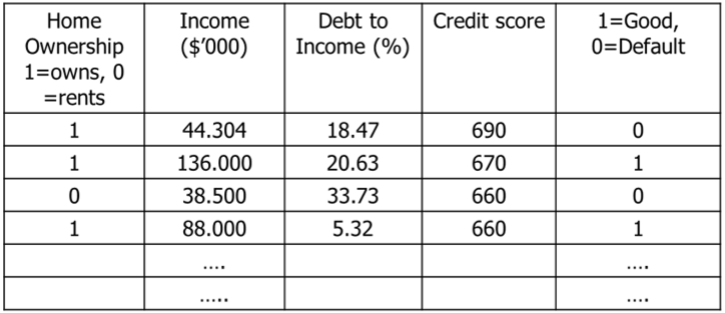
FICO Score Range Breakdown

- **300-579 (Poor)**: High interest rates, difficult loan approvals.
- **580-669 (Fair)**: Some lenders approve loans, but with high interest.
- **670-739 (Good)**: Average U.S. credit score, decent interest rates.
- **740-799 (Very Good)**: Better loan terms and lower interest rates.
- **800-850 (Exceptional)**: Best rates and highest credit limits.

Use in LendingClub

- In the **LendingClub dataset**, FICO scores are given as a **range** (e.g., 700-724).
- **fico_low**: The lower bound of the range.
- **fico_high**: The upper bound of the range.
- This range helps investors assess **borrower risk** when making lending decisions.

The Data



Home Ownership 1=owns, 0 =rents	Income ($'000)	Debt to Income (%)	Credit score	1=Good, 0=Default
1	44.304	18.47	690	0
1	136.000	20.63	670	1
0	38.500	33.73	660	0
1	88.000	5.32	660	1
	....			....
	.....			....

Results for Lending Club Training Set

- X_1= Home Ownership
- X_2= Income
- X_3= Debt to income ratio
- X_4 = Credit score

$$Y = -6.5645 + 0.1395 \cdot X_1 + 0.0041 \cdot X_2 - 0.0011 \cdot X_3 + 0.0113 \cdot X_4$$

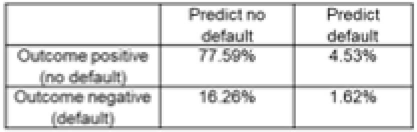
Decision Criterion

- The data set is imbalanced with more good loans than defaulting loans
- There are procedures for creating a balanced data set
- With a balanced data set we could classify an observation as positive if $Q > 0.5$ and negative otherwise
- However this does not consider the cost of misclassifying a bad loan and the lost profit from misclassifying a good loan
- A better approach is to investigate different thresholds, Z
  - If $Q > Z$ we accept a loan
  - If $Q \leq Z$ we reject the loan

Test Results

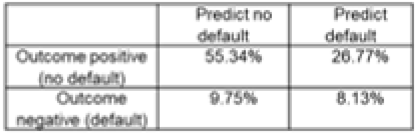
See Hull, Tables 3.10, 3.11, and 3.12

Z = 0.75:



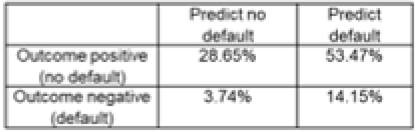
	Predict no default	Predict default
Outcome positive (no default)	77.59%	4.53%
Outcome negative (default)	16.26%	1.62%

Z=0.80:



	Predict no default	Predict default
Outcome positive (no default)	55.34%	26.77%
Outcome negative (default)	9.75%	8.13%

Z=0.85:



	Predict no default	Predict default
Outcome positive (no default)	28.65%	53.47%
Outcome negative (default)	3.74%	14.15%

The Confusion matrix and common ratios

	Predict positive outcome	Predict negative outcome
Outcome positive	TP	FN
Outcome negative	FP	TN

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

$$\text{True Positive Rate (TPR also called sensitivity or recall)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

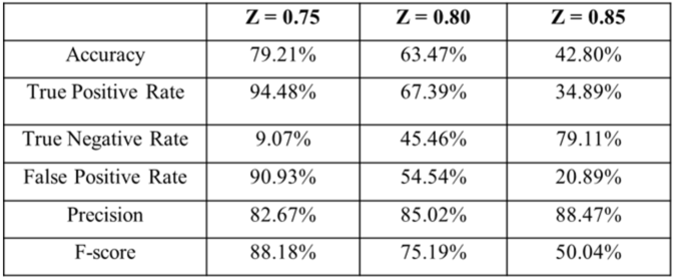
$$\text{The True Negative rate(also called specificity)} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{The False Positive Rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

$$\text{Precision, P} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

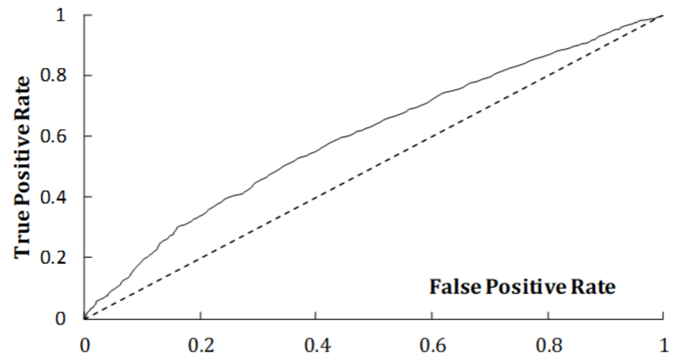
$$\text{F score} = 2 \times \frac{\text{P} \times \text{TPR}}{\text{P} + \text{TPR}}$$

Test Set Ratios for different Z values



	Z = 0.75	Z = 0.80	Z = 0.85
Accuracy	79.21%	63.47%	42.80%
True Positive Rate	94.48%	67.39%	34.89%
True Negative Rate	9.07%	45.46%	79.11%
False Positive Rate	90.93%	54.54%	20.89%
Precision	82.67%	85.02%	88.47%
F-score	88.18%	75.19%	50.04%

As we change the Z criterion we get an ROC



Area Under Curve (AUC)

- The area under the curve is a popular way of summarizing the predictive ability of a model to estimate a binary variable
- When $AUC = 1$ the model is perfect.
- When $AUC = 0.5$ the model has no predictive ability
- When $AUC < 0.5$ the model is worse than random
- In this case $AUC = 0.6020$

Choosing Z

- The value of Z can be based on
- The expected profit from a loan that is good, P
- The expected loss from a loan that defaults, L
- We need to maximize: $P \times \mathrm{TP} - L \times \mathrm{FP}$

Conclusions

Key Takeaways (1/3): Linear Models in Finance

- **Linear regression** provides a transparent baseline for continuous targets and a valuable **diagnostic tool** for understanding data and model risk.
- In high-dimensional or correlated feature spaces, **regularization is essential**:
  - **Ridge** stabilizes estimates by shrinking coefficients
  - **Lasso** promotes sparsity (but can be unstable under collinearity)
  - **Elastic Net** balances shrinkage and sparsity
- Regularization **does not create information**: it controls complexity to improve **out-of-sample robustness**.

Key Takeaways (2/3): Logistic Regression and Decisions

- **Logistic regression** outputs a **probability model**:

$$\hat{p}(y = 1 \mid x) = \sigma(w^\top x + b).$$

- Turning probabilities into actions requires a **decision rule**:

$$\hat{y} = \mathbb{I}(\hat{p} \geq z),$$

  where the threshold z is **policy-driven** (business, risk, regulatory constraints), not a universal constant.
- In banking and finance, misclassification costs are typically **asymmetric**:
  - **False positives**: risky cases classified as safe (direct losses)
  - **False negatives**: safe cases classified as risky (opportunity costs)

Key Takeaways (3/3): Performance Metrics that Matter

- **Accuracy can be misleading** in imbalanced datasets (common in credit, fraud, AML).
- The **confusion matrix** is the foundation for meaningful metrics:
  - **Precision**: control false positives (operational burden / direct losses)
  - **Recall**: control false negatives (missed risky cases)
  - **F1-score**: trade-off between precision and recall (model comparison)
- **ROC and AUC** evaluate **ranking/separation ability** across thresholds:
  - AUC summarizes discrimination, but **does not select the optimal threshold**
  - Threshold selection requires **costs, constraints, and objectives**

From Linear Models to Decision Trees: What Changes?

Linear and logistic models are powerful, but they impose a strong assumption:

$f(x)$ is approximately linear in the features.

In many financial problems, relevant patterns may involve:

- **Non-linear effects** (e.g. thresholds, saturation, regime-like behavior)
- **Feature interactions** (e.g. high DTI *and* low FICO)
- **Heterogeneous segments** (e.g. different borrower profiles)

Decision trees address these points by learning **rules** and **splits**, but introduce new challenges (especially **variance** and **overfitting**).

Bridge to Lesson 3.2: Decision Trees (Preview)

In the next lecture we will study **decision trees** as supervised models that:

- learn **non-linear decision boundaries** via recursive partitioning,
- provide **human-readable rules** (high interpretability),
- naturally capture **interactions** and **segmentations**.

We will also focus on why decision trees can be **fragile**:

- small data changes can produce very different trees,
- deep trees overfit quickly (**high variance**),
- careful validation and pruning are essential.

Key question: how can we keep interpretability while controlling variance?