3.2 - Decision Trees

Giovanni Della Lunga
giovanni.dellalunga@unibo.it

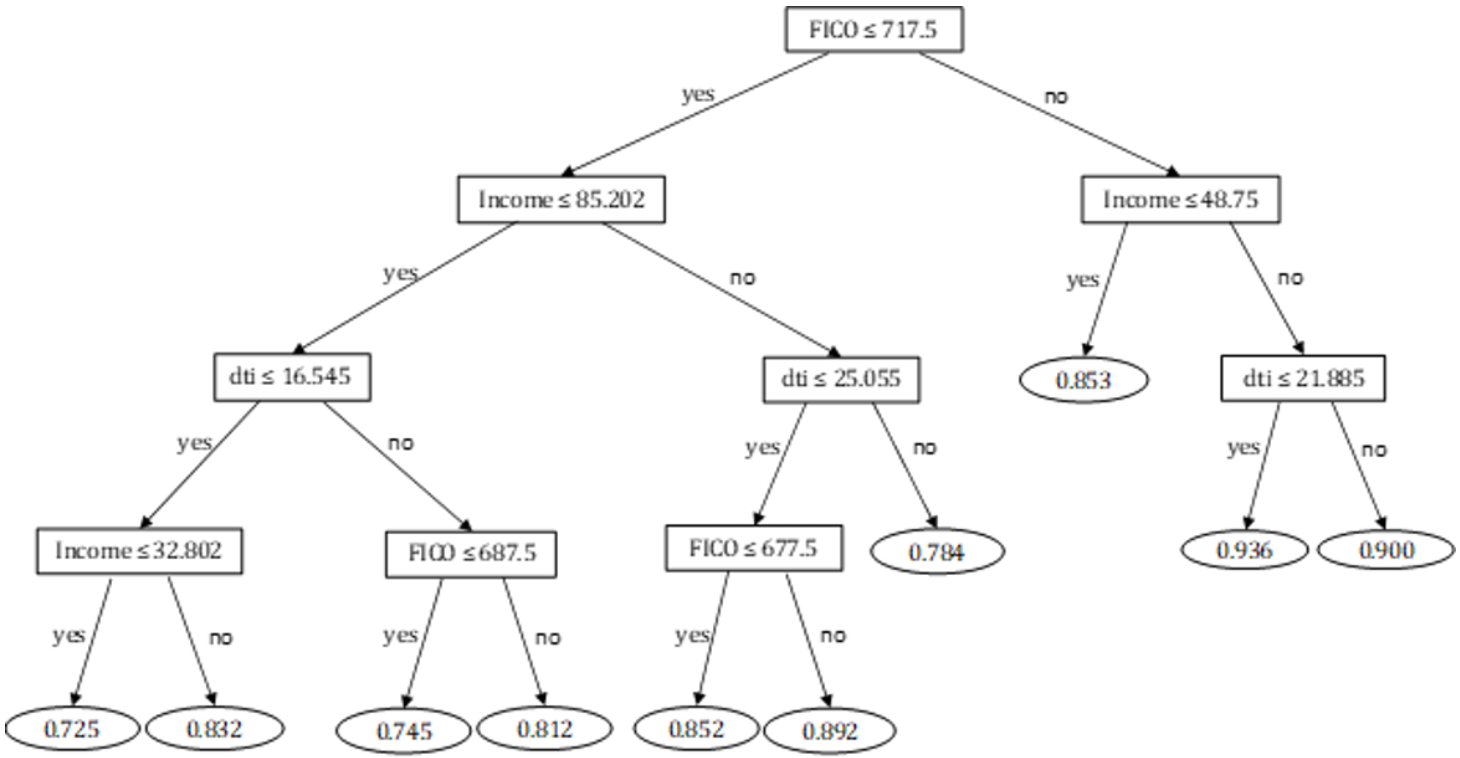
Introduction to Machine Learning for Finance

Bologna - February-March, 2025

What is a decision tree

What is a Decision Tree

- A **Decision Tree** shows a step-by-step process for making predictions;
- Example : if we are classifying bank loan application for a customer, the decision tree may look like this Here we can see the logic how it is making the decision.



What is a Decision Tree

- The decision is made by **considering features one at a time rather than all at once**;
- You will have different kind of splitting depending on the nature of the feature;
- you can have a very simple decision tree in which the decision is a **boolean condition** (true, false) or you can have a **numeric data**
- In the last case we have to define a desired **threshold** value of the feature. For instance, during each split of the parent node, we go to left node (with the corresponding subset of data) if a feature is less than the threshold, and right node otherwise.

How to Build a Decision Tree

- But **how do we decide on the split**?
- **What is the best feature to select at the root node**?
- and how we decide the threshold for numerical data?
- You can answer to these questions using the so called **information gain**
- But what is **information gain**?

How to Build a Decision Tree

Impurity and Information Gain

- Decision trees use the concept of **impurity** to describe how homogeneous or "pure" a node is.
- A node is pure if all its samples belong to the same class, while a node with many samples from many different classes is called impure.
- The difference between the impurity of a node and that of the child nodes is called **Information Gain**.

The goal of a decision tree is, at each layer, to try to split the data into two (or more) groups, so that data that fall into the same group are most similar to each other (homogeneity), and groups are as different as possible from each other (heterogeneity).

How to Build a Decision Tree

- In order to split the nodes at the most informative features, we need to define an objective function that we want to optimize via the tree learning algorithm.
- Here, our objective function is to maximize the **IG** at each split, which we define as follows:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \tag{1}$$

Where: f is the feature to perform the split, D_p and D_j are the dataset of the parent and *jth* child node, I is our impurity measure, N_p is the total number of training examples at the parent node and N_j is the number of examples in the jth child node.

How to Build a Decision Tree

- As we can see, the information gain is simply the **difference between the impurity of the parent node and the sum of the child node impurities** — the lower the impurities of the child nodes, the larger the information gain.
- However, for simplicity and to reduce the combinatorial search space, most libraries (including scikit-learn) implement binary decision trees.
- This means that each parent node is split into two child nodes, D_{left} and D_{right}:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right}) \qquad (2)$$

Measures of Information Gain

How to Build a Decision Tree

The three impurity measures or splitting criteria that are commonly used in binary decision trees are:

- **Entropy**
- **Gini Impurity**
- **Classification Error**

In the following we will focus only on the first two.

Shannon Entropy

- Before taking on Shannon's definition of entropy, it might help to break down his definition of information.
- The basic idea of Shannon's theory is that the informative value of a communicated message **depends on the degree to which the content of the message is surprising**.
- Shannon considered a source generating some inputs x characterised by a probability distribution $p(x)$.
- Then he introduced the quantity

$$I_p(x) = log_2 \frac{1}{p(x)} = -log_2\, p(x) \tag{3}$$

Information is uncertainty reduction which in turn is just the inverse of the event?s probability.

Shannon Entropy: Why the log?

- Say, for example, that you have a simple binomial model completely random, with a 50/50 chance for a stock of being either up or down in the next interval Δt.
- If someone tells you that the stock it?s going to be up tomorrow then they have actually reduced your uncertainty by a factor of two.
- There were two equally likely options, now there is just one.
- In this case the "channel" (whatever it is) actually sent you a **single bit** of useful information:

$$I_p(x) = -log_2 \frac{1}{2} = log_2 2 = 1 \tag{4}$$

Shannon Entropy: Why the log?

- Now suppose the future asset price has actually 8 possible states, all equally likely.
- Now when your insider gives you tomorrow?s price, he is dividing your uncertainty by a factor of 8, which is 2 to the power of 3.
- So he sents you 3 bits of useful information.
- Again, the number of bits of information that were actually communicated by computing the binary logarithm of the uncertainty reduction factor, which in this example is 8.

Shannon Entropy: Why the log?

- But what if the possibilities are not equally likely?
- Say 75% chance up, and 25% down.
- If the insider tells you the asset it?s going to be down tomorrow, then your uncertainty has dropped by a factor of 4, which is 2 bits of information.
- Again the uncertainty reduction is just the inverse of the event?s probability, in this case the inverse of 25% is 4 and the *log* is 2.

Shannon Entropy: Why the log?

- Now if the insider tells you the asset it?s going to be up tomorrow then your uncertainty hasn?t dropped much.
- In fact, you get just about 0.41 bits of information.
- So how much information are you actually going to get from the insider, on average?
- Well, there?s a 75% chance that the asset will be up tomorrow, and you get in this case 0.41 bits of information.
- Then there?s a 25% chance that it will be down, in which case and this will give you 2 bits of information.
- So on average, you will get 0.81 bits of information from your insider.

Shannon Entropy

- So what we just computed is called the **Entropy**.
- It is a measure of how uncertain the events are.

$$H(p) = -\sum_{i=1}^{n} p_i \log(p_i)$$

Entropy measures the **average amount of information** that you get when you receive a signal from a transmission channel, or more generally the average amount of information that you get from one sample drawn from a given probability distribution p. It tells you how unpredictable that probability distribution is.

Measures of Uncertainty

- Suppose that there are n possible outcomes and p_i is the probability of outcome i with $\sum_{i=1}^{n} p_i = 1$
- Entropy measure of uncertainty:

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \ln(p_i)$$

- Gini Measure of uncertainty:

$$\text{Gini} = 1 - \sum_{i=1}^{n} p_i^2$$

Information Gain

- The information gain is the expected decrease in uncertainty (as measured by either entropy or Gini).
- Suppose that there is a 20% chance that a person will receive a job offer
- Suppose further that there is a 50% chance the person has a relevant degree. If the person does have a relevant degree the probability of a job offer rises to 30%, otherwise it falls to 10%

Information Gain

$$\text{Initial Entropy} = -[0.2\ \log(0.2) + 0.8\ \log(0.8)] = 0.7219$$

$$\begin{aligned}\text{Expected Entropy} = &- 0.5 \cdot [0.1 \log(0.1) + 0.9 \log(0.9)] \\ &- 0.5 \cdot [0.3 \log(0.3) + 0.7 \log(0.7)] \\ &= 0.6751\end{aligned}$$

The Expected information gain from knowing whether there is a relevant degree is

$$\text{gain} = 0.7219 - 0.6751 = 0.0468$$

The Decision Tree Algorithm

- Algorithm **chooses the feature at the root of the tree that has the greatest expected information gain**;
- At subsequent nodes it choose the feature (not already chosen) that has the greatest expected information gain;
- When there is a threshold, it determines the optimal threshold for each feature (i.e., the threshold that maximizes the expected information gain for that feature) and bases calculations on that threshold

Application to credit decision

Introduction

- The Lending Club case illustrates how decision trees can be applied to credit risk assessment.
- The goal is to predict whether a loan will default or not based on four key financial variables:
  - Home Ownership (Categorical: 1 if owned, 0 if rented)
  - Income (Numerical: annual income in dollars)
  - Debt-to-Income Ratio (DTI) (Numerical: total monthly debt payments divided by gross monthly income)
  - Credit Score (FICO Score) (Numerical: a measure of creditworthiness)
- Training dataset: 8,695 observations (1,499 defaulted loans, 7,196 good loans)

Step 1: Initial Entropy Calculation

- Probability of a good loan:

$$P_{\text{good}} = \frac{7,196}{8,695} = 0.8276$$

- Initial entropy:

$$H_{\text{initial}} = -(0.8276 \log 0.8276 + 0.1724 \log 0.1724) = 0.6632$$

Feature	Threshold value	Expected entropy	Expected Information gain
Home Ownership	N.A.	0.6611	0.0020
Income	$85,202	0.6573	0.0058
Debt to income ratio	19.87	0.6601	0.0030
FICO credit score	717.5	0.6543	0.0088

Step 2: Selecting the Root Node

Feature	Threshold	Expected Entropy	Information Gain
Home Ownership	N.A.	0.6611	0.0020
Income ($000s)	85.202	0.6573	0.0058
Debt-to-Income (%)	19.87	0.6601	0.0030
Credit Score (FICO)	**717.5**	**0.6543**	**0.0088**

- Since FICO Score at 717.5 has the highest information gain, it is selected as the root node.

Step 3: Splitting the Tree

Choosing the next node if FICO > 717.5

Feature	Threshold value	Expected entropy	Information gain
Home ownership	N.A.	0.4400	0.0003
Income ($'000s)	48.75	0.4330	0.0072
Debt to income (%)	21.13	0.4379	0.0023
FICO score	789	0.4354	0.0048

Step 3: Splitting the Tree

Choosing the next node if FICO ≤ 717.5

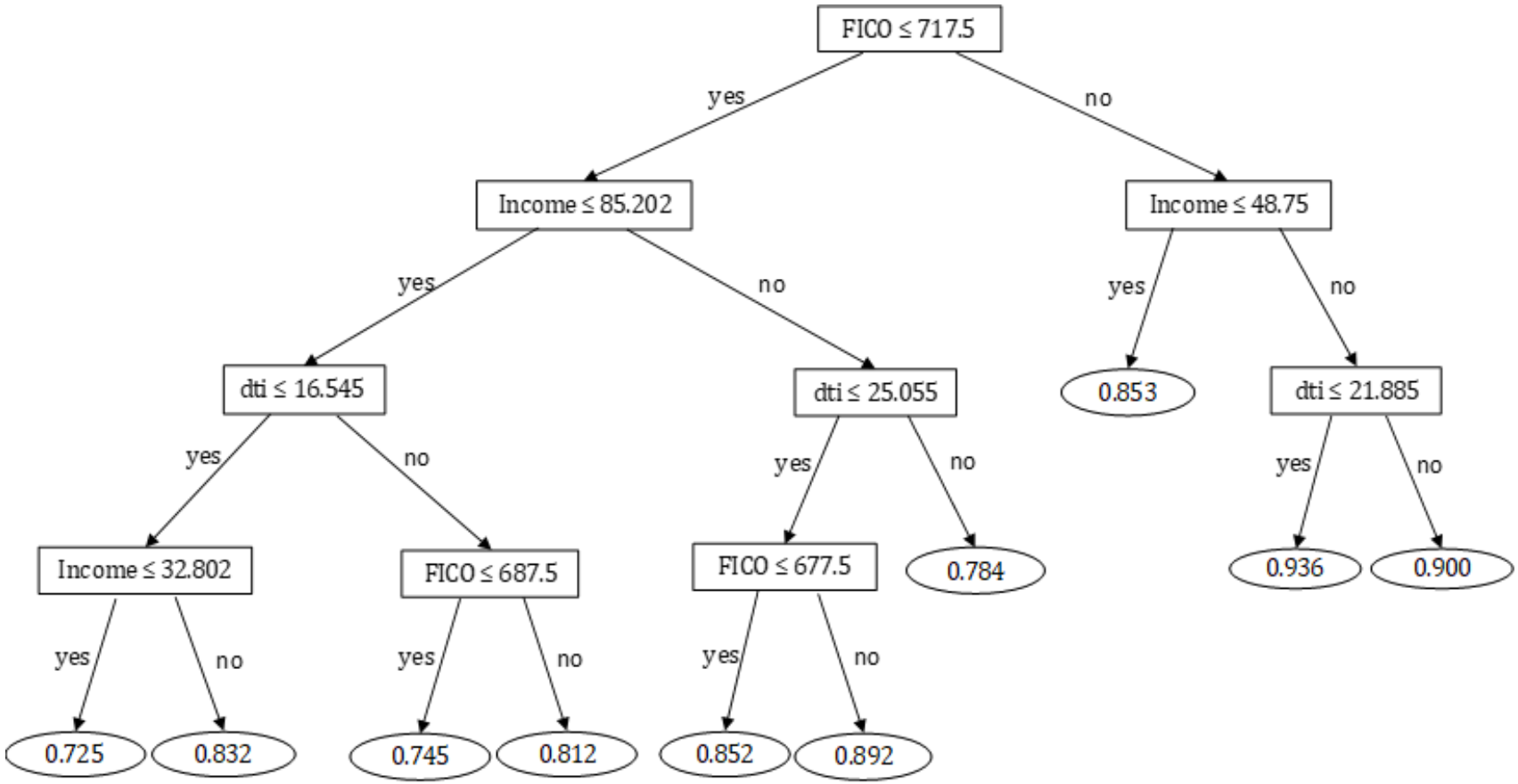
Feature	Threshold value	Expected entropy	Information gain
Home ownership	N.A.	0.7026	0.0017
Income ($'000s)	85.202	0.6989	0.0055
Debt to income (%)	16.80	0.7013	0.0030
FICO score	682	0.7019	0.0025

Step 3: Splitting the Tree

- Further splits:
  - If $FICO > 717.5 \rightarrow$ Further split by $Income > 48.75K$
  - If $FICO \leq 717.5 \rightarrow$ Further split by $Income > 85.202K$ and $DTI \leq 16.545\%$
- Examples:
  - If $FICO > 717.5$, $Income > 48.75K$, and $DTI > 21.885\%$, probability of no default $= 0.900$.
  - If $FICO \leq 717.5$, $Income \leq 32.802K$, and $DTI \leq 16.545\%$, probability of no default $= 0.725$.

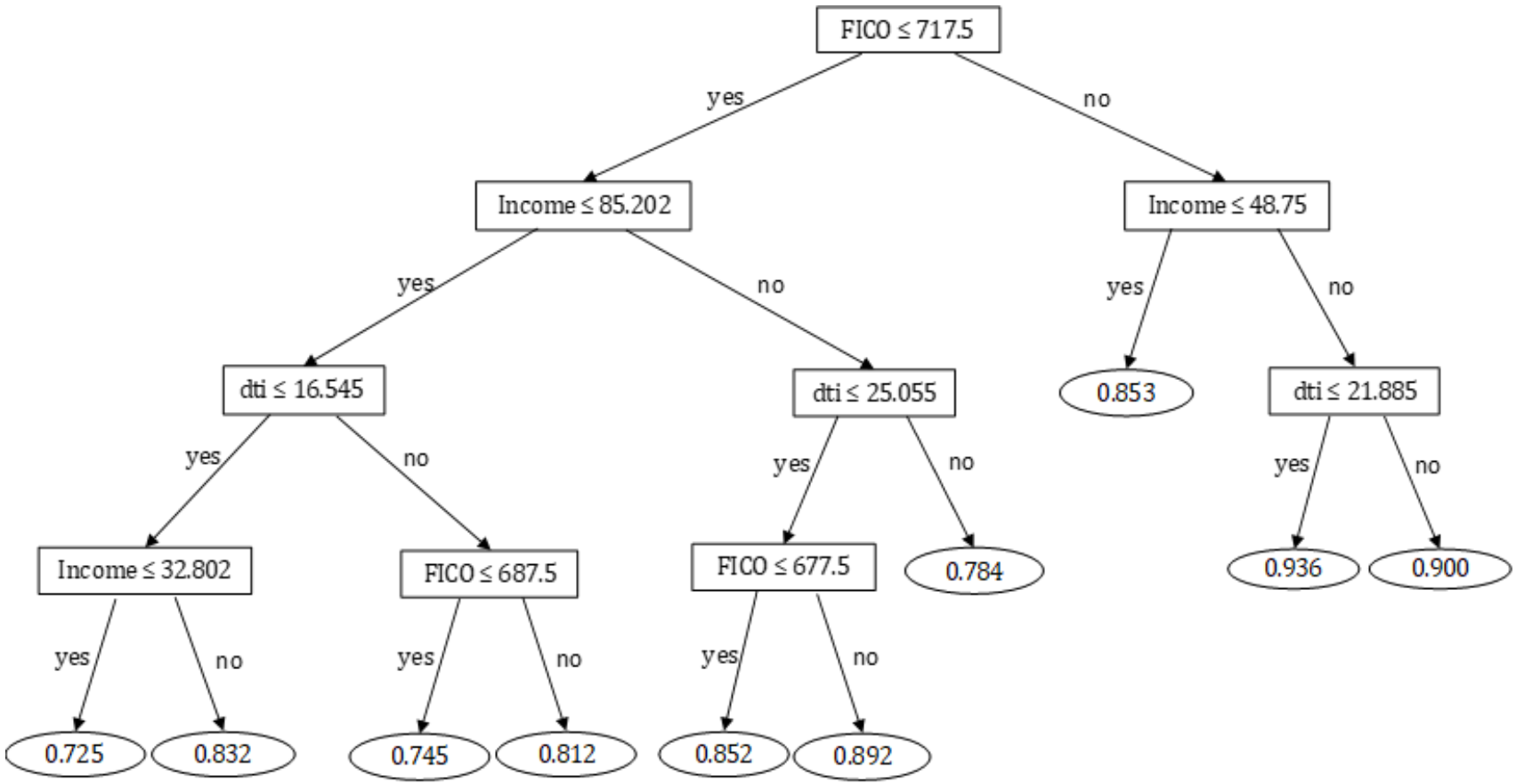
Step 3: Splitting the Tree

The construction of the tree continues in this way. The full tree produced by Sklearn's DecisionTreeClassifier is summarized in Figure.



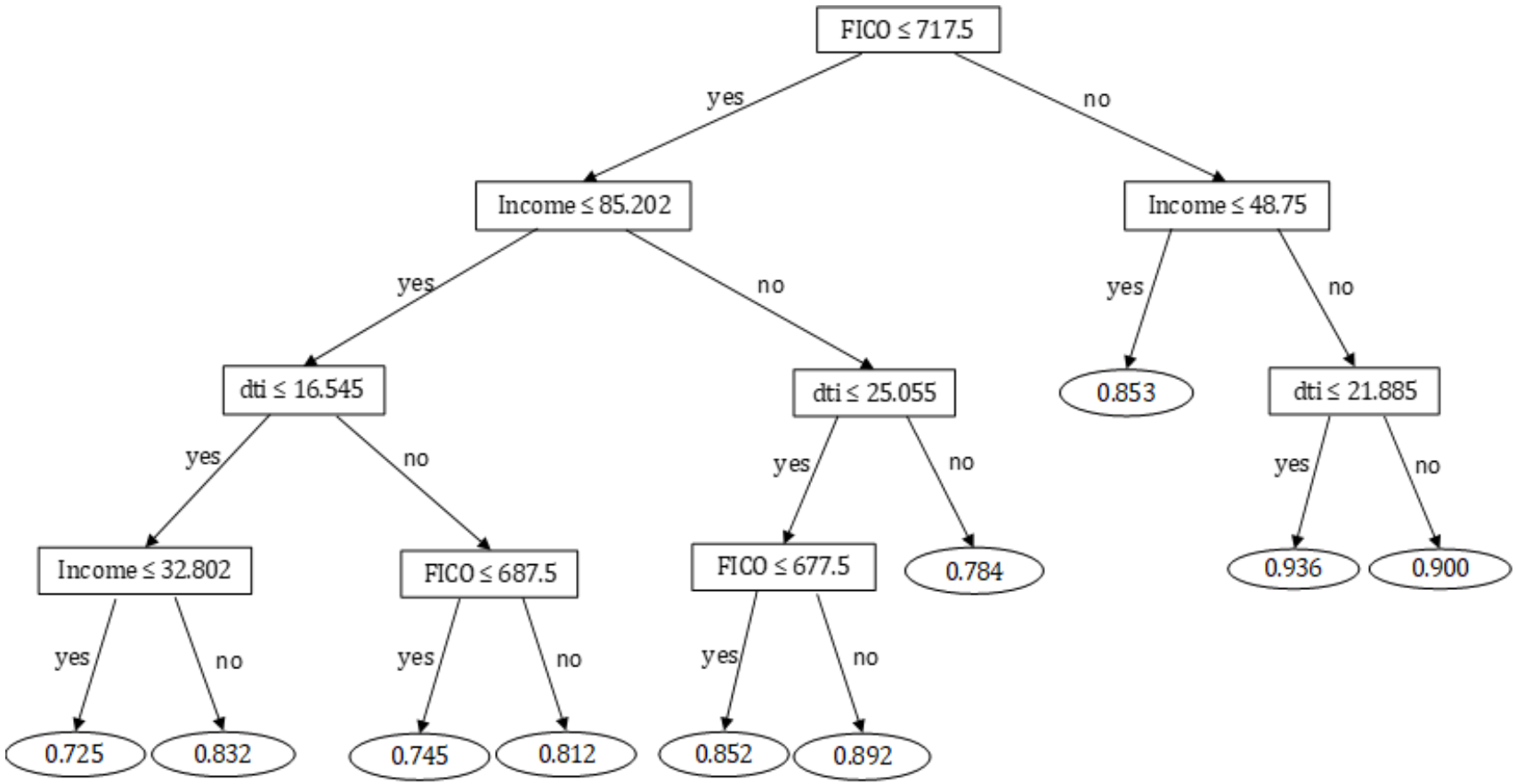
Step 3: Splitting the Tree

The final points reached by the tree (shown as ovals) are referred to as leaves. The numbers shown in the leaves in Figure are the predicted probabilities of default.



Step 3: Splitting the Tree

For example, if $FICO > 717.5$, $Income > 48.75$ and $dti > 21.885$, the estimated probability of no-default is 0.900. This is because there were 379 observations in the training set satisfying these conditions and 341 of them were good loans ($341/379 = 0.900$)

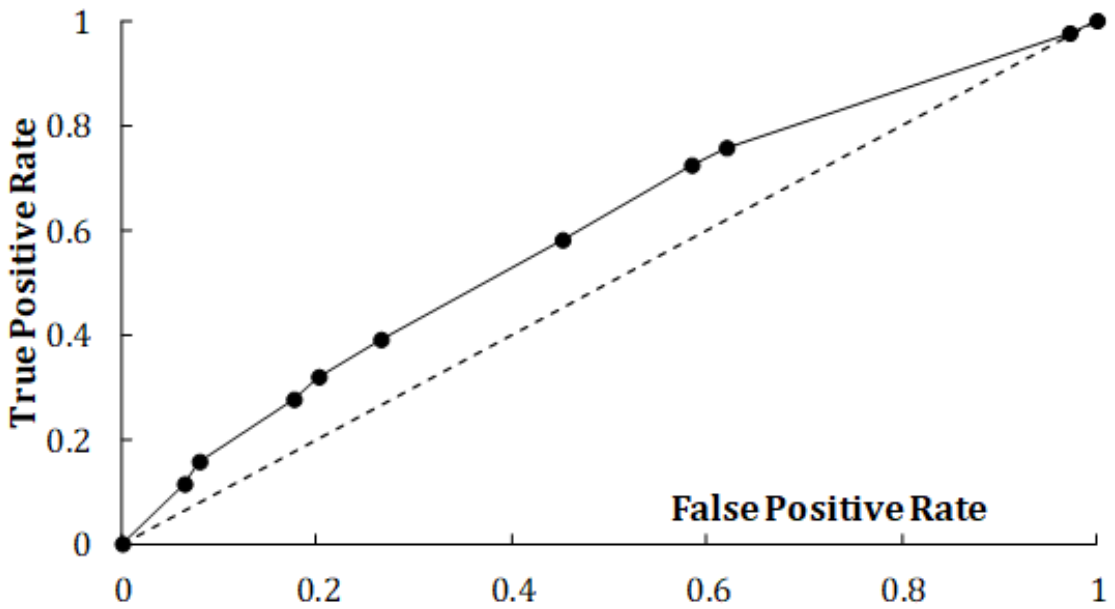


Step 4: Decision Thresholds and ROC Curve

- A Z-value determines the threshold for classifying a loan as good.
- Three thresholds analyzed:
  - $\mathbf{Z = 0.75}$: Predicts that all loans are good except:
    - $Income \leq 85,202$, $DTI > 16.545$, $FICO \leq 687.5$
    - $Income \leq 32,802$, $DTI \leq 16.545$, $FICO \leq 717.5$
  - $\mathbf{Z = 0.80}$: Also excludes $FICO \leq 717.5$, $Income > 85,202$, $DTI > 25.055$.
  - $\mathbf{Z = 0.85}$: Predicts loans as good only if:
    - $FICO > 717.5$, or
    - $FICO \leq 717.5$, $Income > 85,202$, $DTI \leq 25.055$.

Step 5: Performance Evaluation

- AUC (Area Under Curve) used to measure predictive performance:
  - Decision Tree **AUC = 0.5948**
  - Logistic Regression **AUC = 0.6020**
- While logistic regression performs slightly better, decision trees offer interpretability.



Conclusion

- The decision tree approach to Lending Club loan classification demonstrates how financial variables influence creditworthiness.
- Highlights:
  - The importance of the **FICO score** as the primary predictor.
  - The trade-off between **precision and recall** when selecting a **Z-threshold**.
  - The interpretability of decision trees compared to other models.
- Provides a structured, data-driven method to assess lending risk.

Continuous Target Variables

Introduction

- A **Decision Tree** can be used to predict continuous values using **Regression Trees**.
- Instead of making a **yes/no** decision like classification, a regression tree splits the data into smaller groups and assigns each group an **average value**.

Example: Predicting House Prices

Dataset:

Size (m^2)	Rooms	Price ($)
50	2	100,000
60	2	120,000
70	3	150,000
90	3	200,000
100	4	220,000

Finding the Best Split

- We evaluate **every possible split** and choose the one that minimizes the **Mean Squared Error (MSE)**.
- Possible split points: 55, 65, 75, 85, 95, etc.
- The algorithm finds the split that results in the **lowest total error**.

Mathematical Formulation

Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{5}$$

where:

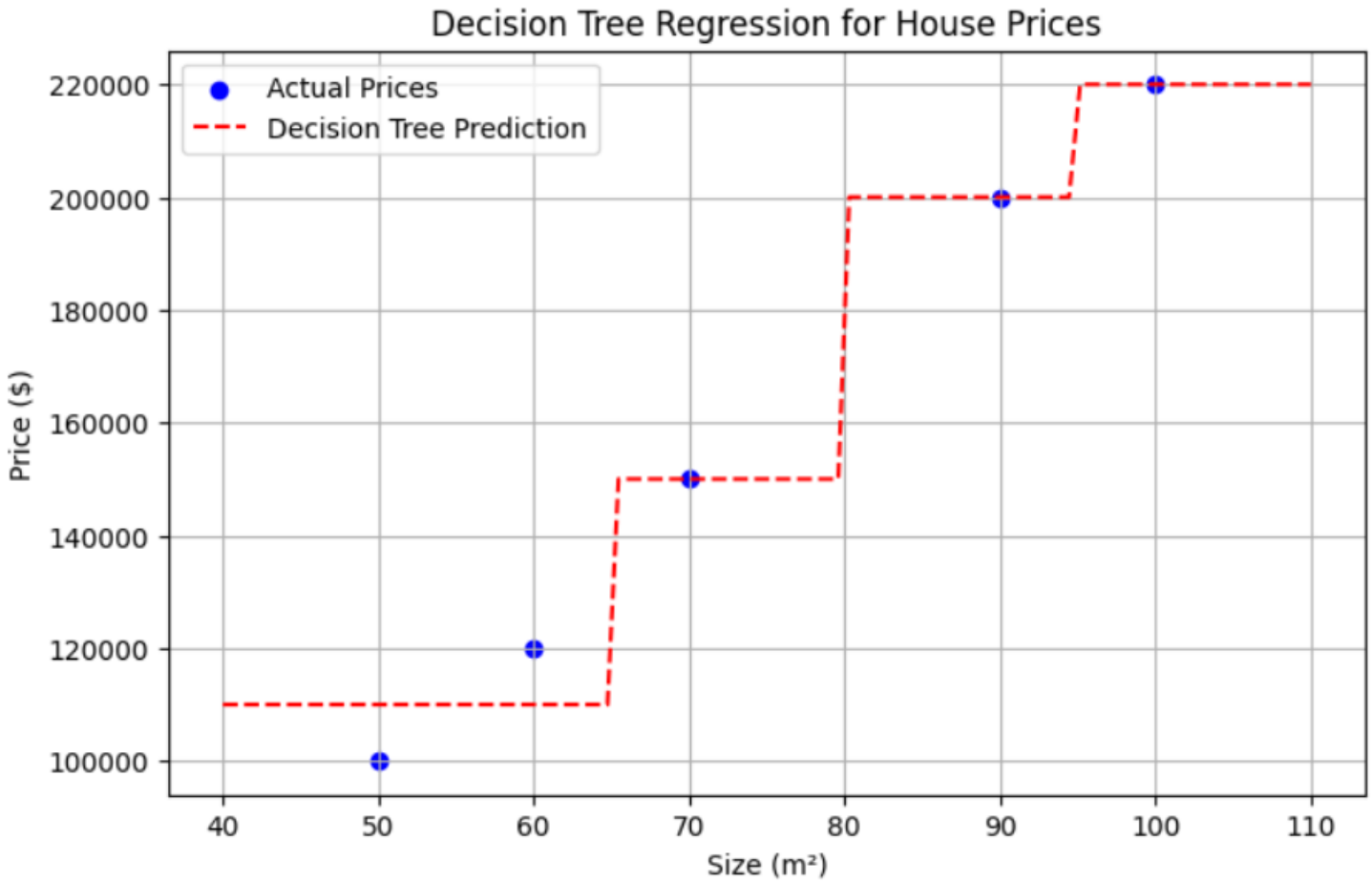
- y_i is the actual value.
- $\hat{y}$ is the predicted value (average of the group).
- n is the number of data points in the group.

Example: Testing Split at $75m^2$

- **Group 1 (Size $\leq$ 75m²)**
  - Houses: 50, 60, 70
  - Average Price: $123,333$
- **Group 2 (Size $>$ 75m²)**
  - Houses: 90, 100
  - Average Price: $210,000$

Automating the Process with Python

Decision tree algorithms like **CART (Classification and Regression Trees)** automatically find the best split. The optimal split point found was **80m²**. Below is a visualization of the Decision Tree Regression model.



Using the Tree for Prediction

- To predict a new value, we follow the tree's splits based on the input features.
- Example: If we want to predict the price of a house of $\mathbf{80m^2}$:
  - Start at the root node (e.g., check if Size $\leq 80m^2$).
  - Follow the correct branch until reaching a terminal leaf.
  - Assign the leaf node's average price as the predicted price.
- This method allows the tree to generalize predictions for new, unseen data.

Predicting House Prices in Iowa

Example: Predicting house prices in Iowa using two features:

- Overall Quality (Scale 1 to 10)
- Living Area (Square Feet)

Step 1: Optimal Root Node Selection

Predicting House Prices in Iowa

- Using an iterative search, the best root node is found:
  - **Overall Quality**: Optimal threshold $= 7.5$
  - **Living Area**: Alternative candidate, but results in higher MSE
- Since Overall Quality has the lowest expected MSE, it is chosen as the root node.

Step 2: Further Splitting

Predicting House Prices in Iowa

- After splitting at Overall Quality $= 7.5$:
  - If Overall Quality ≤ 7.5, the next optimal split is Overall Quality $= 6.5$.
  - If Overall Quality > 7.5, the next optimal split is Overall Quality $= 8.5$.
- After these splits, Living Area is used for further refinement.

Step 3: Final Predictions at Leaf Nodes

Predicting House Prices in Iowa

- Each leaf node contains multiple observations.
- The predicted value at a leaf node is the average of all values in that node.
- This results in a piecewise constant prediction function.

Step 4: Model Performance

Predicting House Prices in Iowa

- The dataset is divided into:
  - 1,800 observations in the training set
  - 600 observations in the validation set
  - 508 observations in the test set
- The mean and standard deviation of house prices in the training set:
  - Mean price: \$180.817K
  - Standard deviation: \$77.201K

Conclusion

- Decision Trees can effectively predict continuous values by splitting data into groups.
- The best split is chosen by minimizing **Mean Squared Error**.
- Libraries like `scikit-learn` make it easy to implement regression trees in Python.
- Once trained, the tree can quickly predict new values by following its decision structure.