



Welcome to the **Co**Grammar Decision Trees

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(Fundamental British Values: Mutual Respect and Tolerance)
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- Report a **safeguarding** incident:
www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

Skills Bootcamp

8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

Due Date: 24 March 2024

✓ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

Due Date: 28 April 2024

Skills Bootcamp Progression Overview

✓ Criterion 3: Course Progress

Completion: All mandatory tasks,
including Build Your Brand and
resubmissions by study period end
Interview Invitation: Within 4 weeks
post-course
Guided Learning Hours: Minimum of
112 hours by support end date
(10.5 hours average, each week)

✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship
Outcome: Document within 12
weeks post-graduation
Relevance: Progression to
employment or related
opportunity

CoGrammar

Decision Trees

May 2024

Learning Objectives

- ❖ Understand the concept of **decision trees**.
- ❖ Interpret the **structure** and **components** of decision trees and **optimise** decision tree models.
- ❖ Apply decision trees to solve **regression** and **classification** problems, differentiating between regression and classification trees.

Learning Objectives

- ❖ **Build** and **assess** decision tree models using real-world datasets.
- ❖ Assess **model performance** by identifying the effects of **overfitting** and **underfitting** techniques in decision trees using performance **metrics** like accuracy.

Decision Trees

Introduction

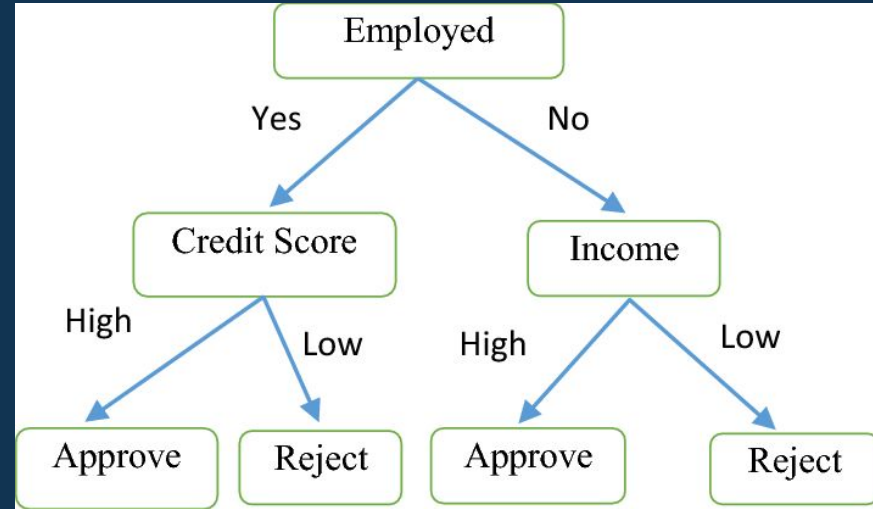


Introduction

Example: Getting a loan from a bank

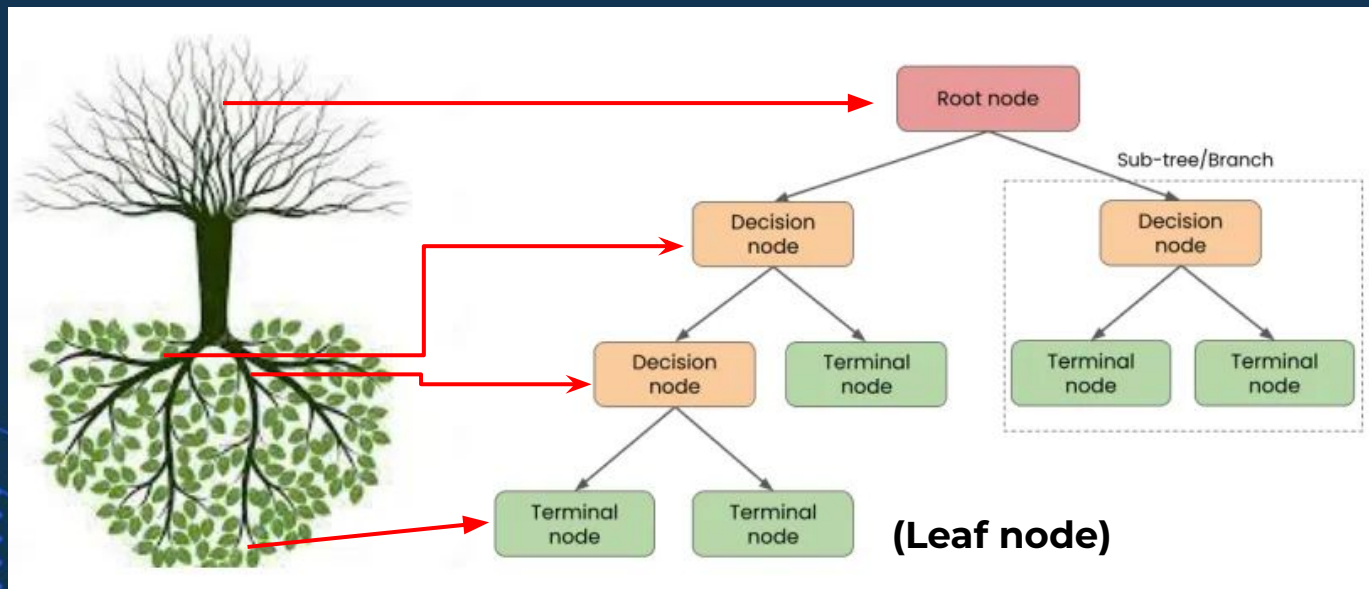
The bank considers **sequential list of questions** and **decides** if it is safe to give a loan to a person.

- ❖ Is the person employed? If yes (or no), move to the next question.
- ❖ What is their current income? If it is high (or low), move to next question.
- ❖ Do they make their credit card payments? If yes, they are offered the loan, and if no, they do not get the loan.



Decision Trees

- ❖ **Supervised** learning algorithm for **regression** and **classification**.
- ❖ **Decision Trees:** (*upside down*) **tree-like** machine learning models.



Decision Trees

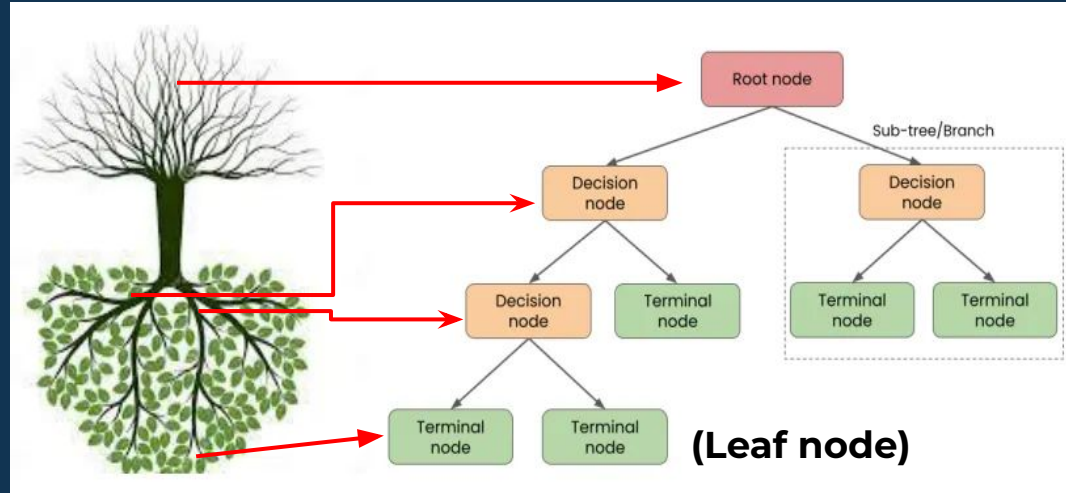
- ❖ Represent data by **partitioning** it into different **smaller subsets** based on questions asked of predictive variable in the data.
- ❖ **Hierarchical**: model is defined by a **sequential questions** that lead to a class label or a value when applied to any observation; model acts like a protocol in a series of “if this occurs then this occurs” conditions that produce a specific result from input data.
- ❖ **Non-parametric**: model is constructed based on the observed data; there are no underlying assumptions about the distribution of the errors or the data.

Components of Decision Trees



Components of Decision Trees

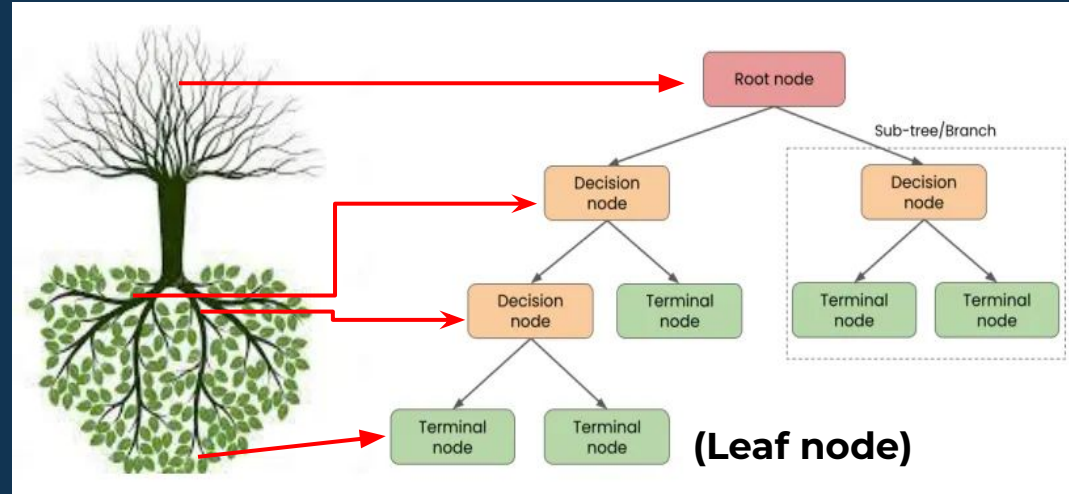
- ❖ **Root Node:** initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.
- ❖ **Decision Nodes:** nodes resulting from the splitting of root nodes; represent intermediate decisions or conditions within the tree.



- ❖ **Leaf (Terminal) Nodes:** nodes where further splitting is not possible, often indicating the final classification or outcome.

Components of Decision Trees

- ❖ **Branch / Sub-Tree:**
subsection of entire decision tree; represents specific path of decisions and outcomes within the tree.
- ❖ **Parent and Child Node:**
Parent node is divided into sub-nodes or **child nodes**.
Parent represents a decision or condition. Child nodes represent outcomes or further decisions.



- ❖ **Pruning:** The process of removing or cutting down specific nodes in a decision tree to prevent overfitting and simplify the model.

Classification and Regression Trees



Classification Trees

Decision tree models where the **target variable** uses a **discrete set of values**, **classification** problems, determine whether an event happened or didn't happen, involving a “yes” or “no” outcome. Each **node**, or **leaf**, represent **class labels** while **branches** represent conjunctions of **features** leading to class labels.

- ❖ The **root node (Outlook)** has two or more **decision nodes (Sunny, Overcast and Rainy)** with other **predictors (Windy, Humidity)**.
- ❖ The **leaf node (Play golf)** is the **target**, and represents a classification of decision.

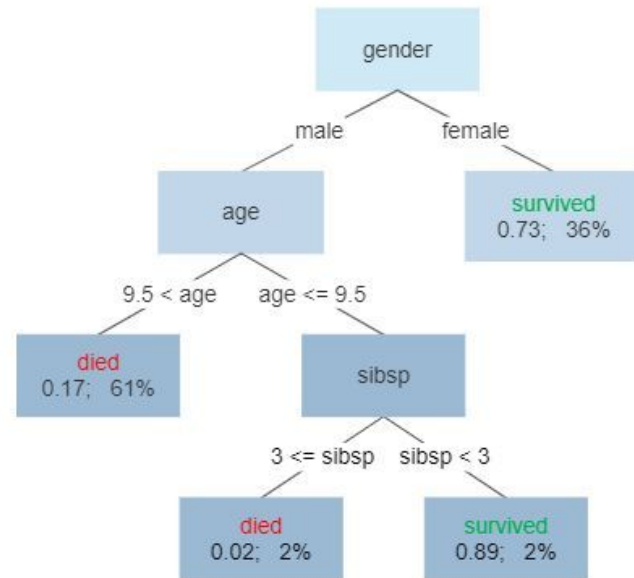


Regression Trees

Decision trees which **predict continuous values** as **targets** based on previous data or information sources. **Predicts** what is likely to happen, given previous behavior/trends.

- ❖ Survival of passengers on the Titanic. Figures under the leaves show the **probability of survival** and the **percentage of observations** in the leaf.
- ❖ "sibsp" is the number of spouses or siblings aboard.

Survival of passengers on the Titanic



CART algorithm

Classification and Regression Trees (CART) algorithm

- ❖ **Tree structure:** CART builds a tree-like structure with nodes and branches.
- ❖ **Nodes:** represent different decision points.
- ❖ **Branches:** represent possible outcomes.
- ❖ **Leaf nodes:** contain a predicted class label or value for the target variable.
- ❖ **Splitting criteria:** CART evaluates all possible splits and selects the one that best reduces the impurity of the resulting subsets.
 - **Gini impurity** (for **classification**, lower means purer subset) and **residual reduction** (for **regression**, lower means better model's fit to the data).
- ❖ **Pruning:** done to prevent overfitting of the data, removes the nodes that contribute little to the model accuracy.

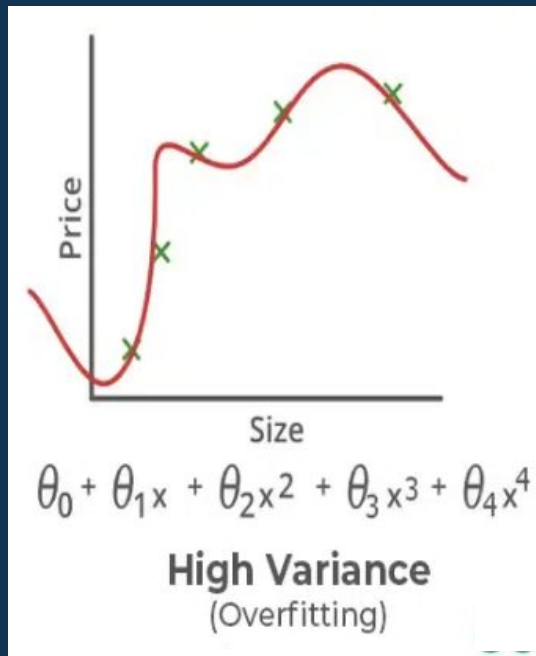
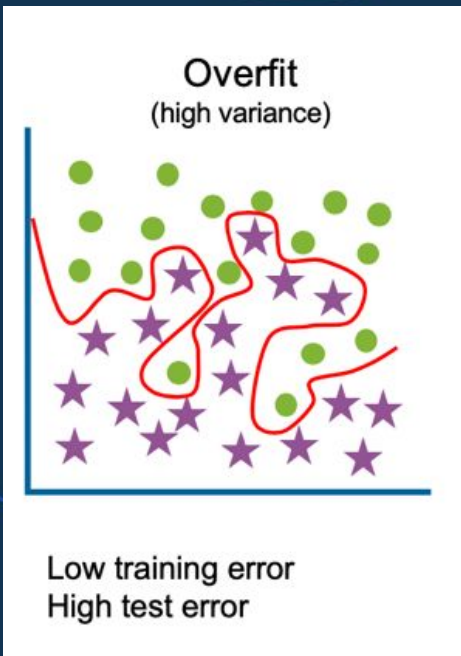
Overfitting and Underfitting



Terminologies

- ❖ **Bias:** prediction error due to **overly simplistic assumptions** in the learning algorithm, not capturing underlying complexities of the data.
- ❖ **Variance:** error due to the **model's sensitivity to fluctuations** in the training data; the variability of the model's predictions for different instances of training data.
- ❖ **Noise:** irrelevant data present in the dataset, affects model performance if it is not removed.
- ❖ **Generalisation:** how well model is trained to predict unseen data.

Overfitting



- Model fits more data than required, and tries to capture each and every data point.
- Starts capturing noise and inaccurate data from dataset.
- **High variance** occurs when a model learns training data's noise and random fluctuations rather than underlying pattern.
- Model gives **accurate predictions for training data** but performs **poorly** for **new test data**.

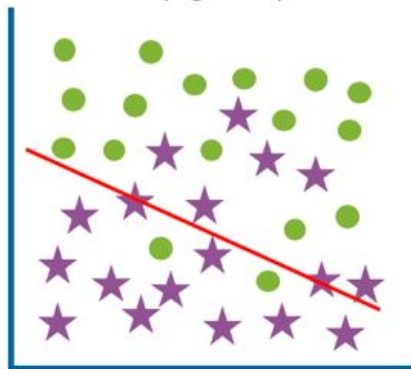
Detecting Overfitting

Mitigating Overfitting

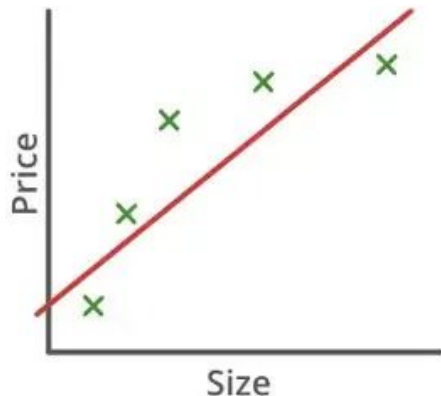
1. **Early Stopping:** training is paused before the model starts learning the noise within the model.
2. **Train with more data:** more chances to discover the relationship between input and output variables.
3. **Feature Selection:** identify the most important features within training data, and other features are removed.
4. **K-fold Cross-Validation:** divide dataset into k-equal-sized subsets (folds)
5. **Data Augmentation:** slightly modified copies of already existing data are added to the dataset.
6. **Regularisation:** group of methods that forces the learning algorithms to make a model simpler, apply a penalty value to features with minimal impact.
7. **Ensembling:** combines predictions from several weak ML algorithms, e.g. **bagging** (trains models in serial) and **boosting** (trains in parallel).

Underfitting

Underfit
(high bias)



High training error
High test error

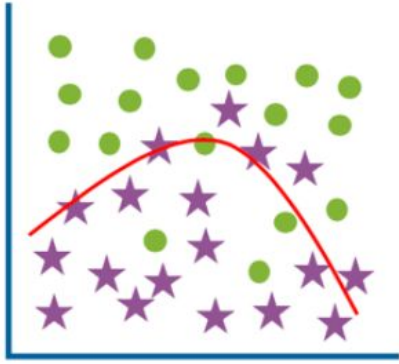


$\theta_0 + \theta_1 x$
High Bias
(Underfitting)

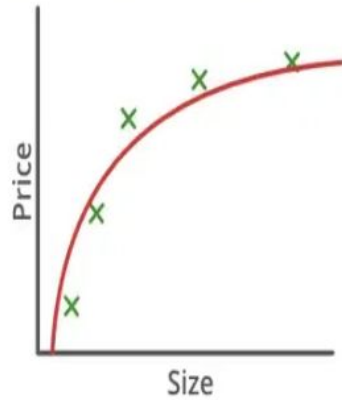
- Model too straightforward to capture data complexities.
- Represents inability of model to learn the training data effectively result in **poor performance** both on the **training** and **testing data**
- **High bias** as the model is unable to represent the true relationship between input and output accurately.

Optimum Fitting

Optimum

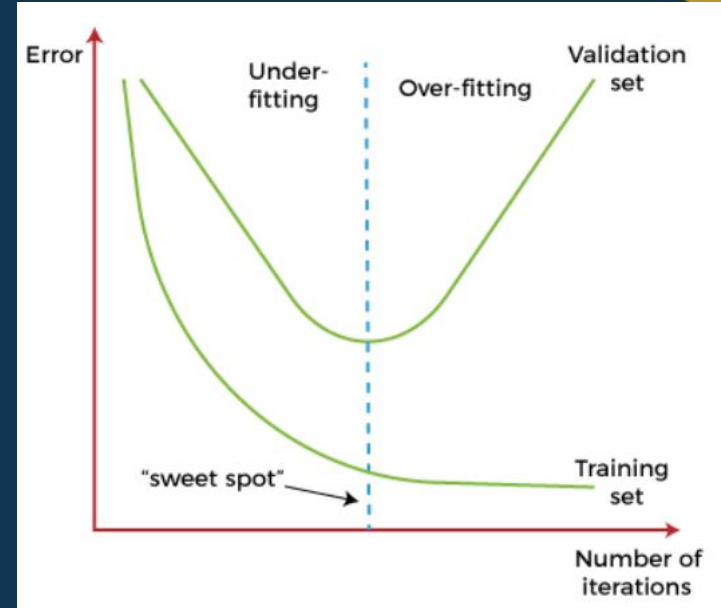


Low training error
Low test error



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Low Bias, Low Variance
(Goodfitting)



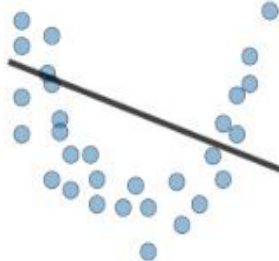

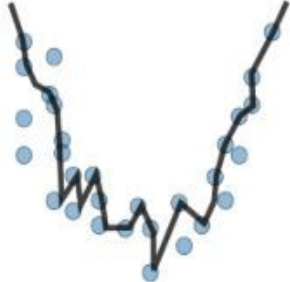
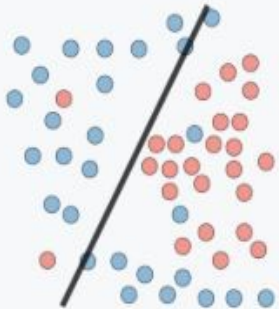
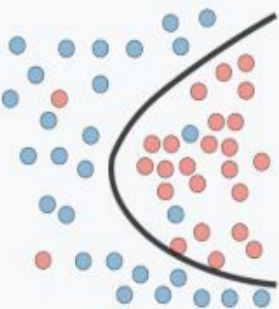
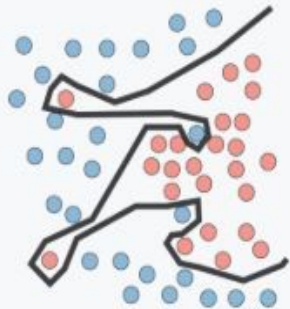
The goal of the machine learning models to **achieve the goodness of fit**, how closely the result or predicted values match the true values of the dataset.

Aim is to find the **sweet spot** between underfitting and overfitting when fitting a model, use **validation dataset**.

Examples

Bias-Variance Tradeoff

Finding the sweet-spot or the balance between underfitting and overfitting

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			
Classification illustration			
Possible remedies	<ul style="list-style-type: none">• Complexify model• Add more features• Train longer		<ul style="list-style-type: none">• Perform regularization• Get more data

Implementing Decision Trees



Decision Trees with Diabetes dataset

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Pregnancies	768 non-null	int64
1	Glucose	768 non-null	int64
2	BloodPressure	768 non-null	int64
3	SkinThickness	768 non-null	int64
4	Insulin	768 non-null	int64
5	BMI	768 non-null	float64
6	DiabetesPedigreeFunction	768 non-null	float64
7	Age	768 non-null	int64
8	Outcome	768 non-null	int64

Classify and predict diabetes based on **features**.

Target is Outcome = 0 for not diabetic and 1 for diabetic.

```
df = pd.read_csv('diabetes.csv')
df.info()
```

#Features and Target

```
X = df.drop(columns=['Outcome'])
y = df['Outcome']
```

#70% training and 30% test

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=42)
```

Implementing Decision Tree

```
# Import Decision Tree Classifier  
from sklearn import tree  
from sklearn.tree import DecisionTreeClassifier
```

DecisionTreeRegressor
for regression tasks
(will see an example in
Tutorial)

```
# Create Decision Tree classifier object  
# training a model without pruning  
unpruned = DecisionTreeClassifier(random_state=42)
```

```
# Train Decision Tree Classifier  
unpruned.fit(X_train,y_train)
```

Implementing Decision Tree

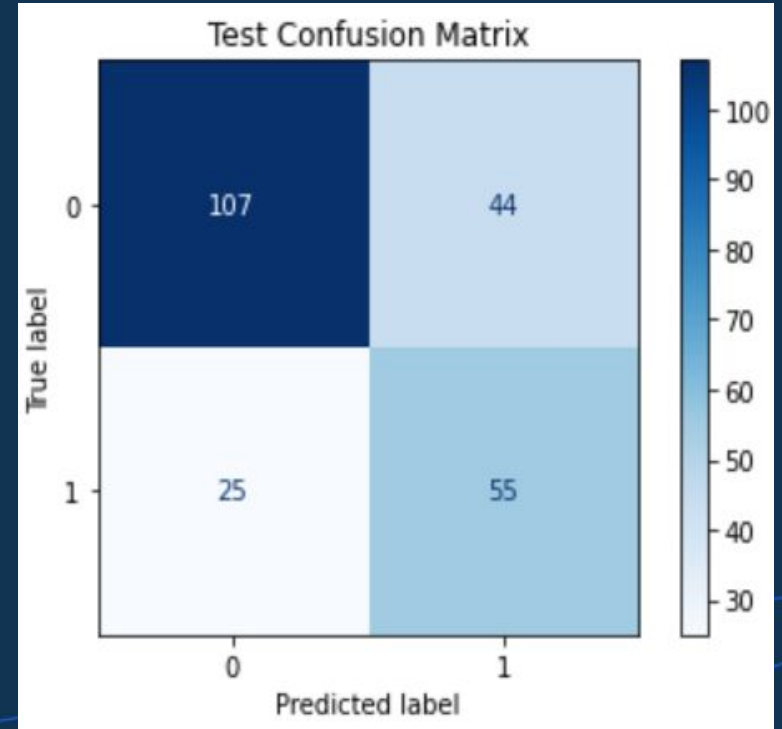
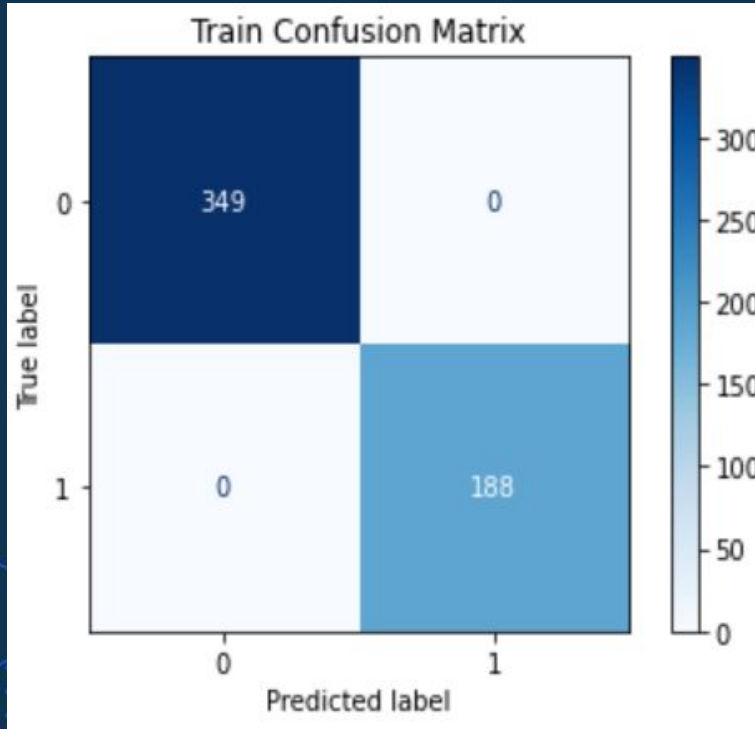
```
#Predict the response for test dataset  
y_train_pred = unpruned.predict(X_train)  
y_test_pred = unpruned.predict(X_test)
```

```
#Model Accuracy and Confusion Matrix, how often is the classifier correct  
print(f'Training Accuracy: {accuracy_score(y_train, y_train_pred)}')  
print(f'Testing Accuracy without pruning: {accuracy_score(y_test, y_test_pred)}')  
print("At depth:", unpruned.tree_.max_depth)
```

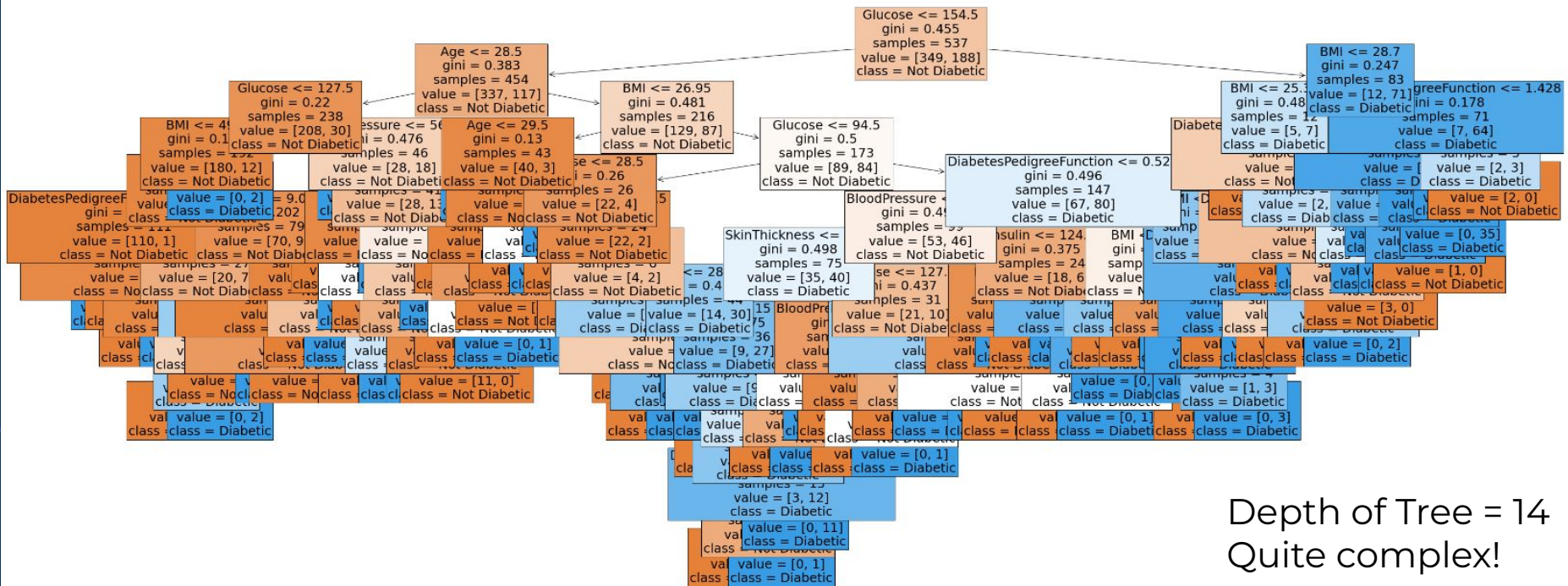
Depth of the tree = 14

```
Training Accuracy: 1.0  
Testing Accuracy without pruning: 0.7012987012987013  
At depth: 14
```


Implementing Decision Tree



Implementing Decision Tree



Gini impurity: measures the level of impurity or randomness in the subsets. Goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.

Optimising Decision Trees

Pruning and max_depth parameter





Optimising Decision Trees: Pruning




A **Decision tree** that is trained to its **full depth** will highly likely lead to **overfitting** the training data.

Pruning avoids **overfitting**, **removes** parts of the Decision Tree that have little or **no significance** in the **decision-making process** and **prevent** it from growing to its **full depth**.

Construct an algorithm that will perform worse on training data but will **generalise better on test data** by tuning the model hyperparameters.

Pre-pruning is done while growing the tree while **post-pruning** prunes nodes after it is built to depth.



Pre- and Post-Pruning

Pre-pruning

- ❖ Tunes hyperparameters (**max_depth**, **min_samples_leaf**, **min_samples_split**) prior to the training pipeline, get a robust model.
- ❖ **'Early stopping'** - **stops** the growth of the **decision tree** to reach its **full depth**, avoid producing leaves with small samples.
- ❖ Cross-validation error monitored at each step, if it is constant, stops growth.

Post-pruning

- ❖ **Decision Tree** model grows to its **full depth**, **tree branches** are **removed** to prevent the model from overfitting.
- ❖ Tune hyperparameter (**ccp_alpha** - **Cost Complexity Pruning**) to control the size of a tree, higher value leads to an increase in the number of nodes pruned.

Other Optimisations

max_depth: maximum depth of the tree.

If None, then nodes are expanded until all the leaves contain less than **min_samples_split** samples.

Too high values cause overfitting, and lower values cause underfitting.

Some other hyperparameters: **min_weight_fraction_leaf**,
max_leaf_nodes, **max_features**

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

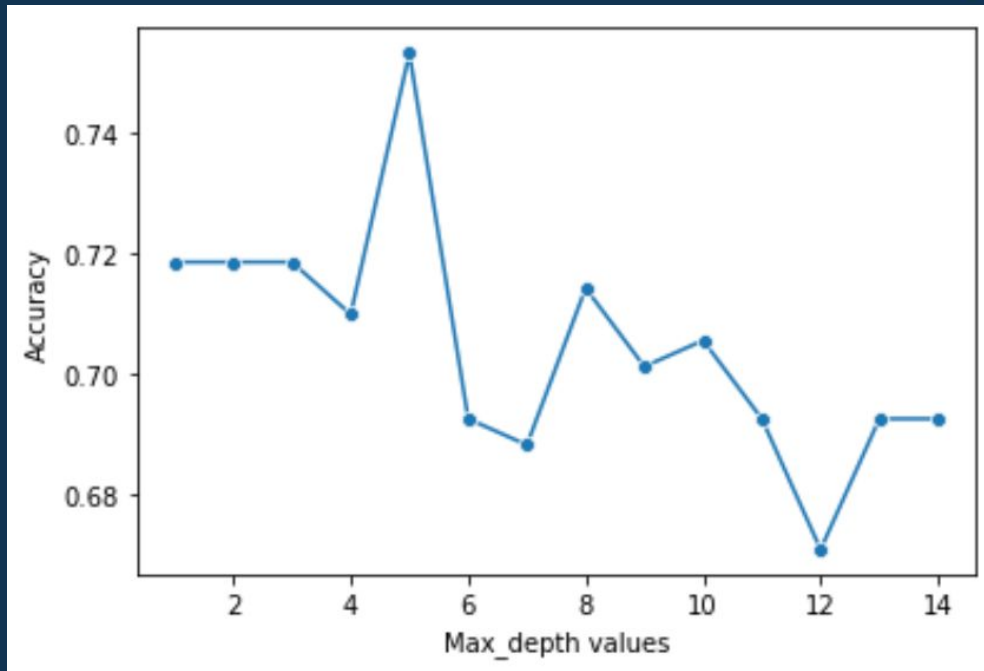
Increasing min hyperparameters or **reducing max** hyperparameters will regularize the model.

Pre-Pruned model

We will choose **max_depth = 5**
(default = None)

We will also choose **min_samples_leaf = 2**, (default=1),
the minimum number of samples required to be at a leaf node.

We will also change the **splitting criterion** (the function to measure the quality of a split) to '**entropy**' instead of the default 'gini'.



How **accuracy** changes with the **max_depth** values. (The tree has a maximum depth of 14).

Pre-Pruned model

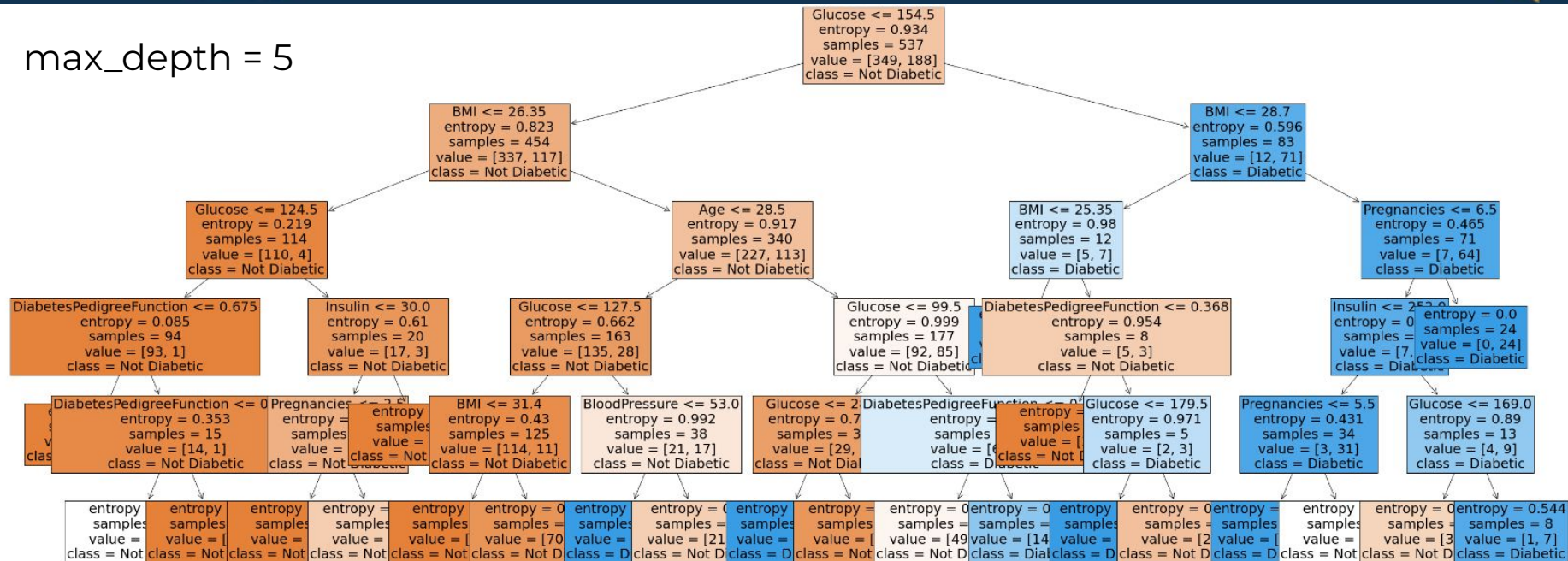
```
pruned = DecisionTreeClassifier(criterion="entropy", max_depth=5,  
random_state=42,min_samples_leaf=2)  
  
# Train Decision Tree Classifier  
pruned.fit(X_train,y_train)  
  
#Predict the response for test dataset  
y_test_pred = pruned.predict(X_test)  
  
# Model Accuracy, how often is the classifier correct  
print("Training Accuracy:",accuracy_score(y_train, y_train_pred))  
print("Testing Accuracy:",accuracy_score(y_test, y_test_pred))
```

Training Accuracy: 1.0

Testing Accuracy: 0.7792207792207793

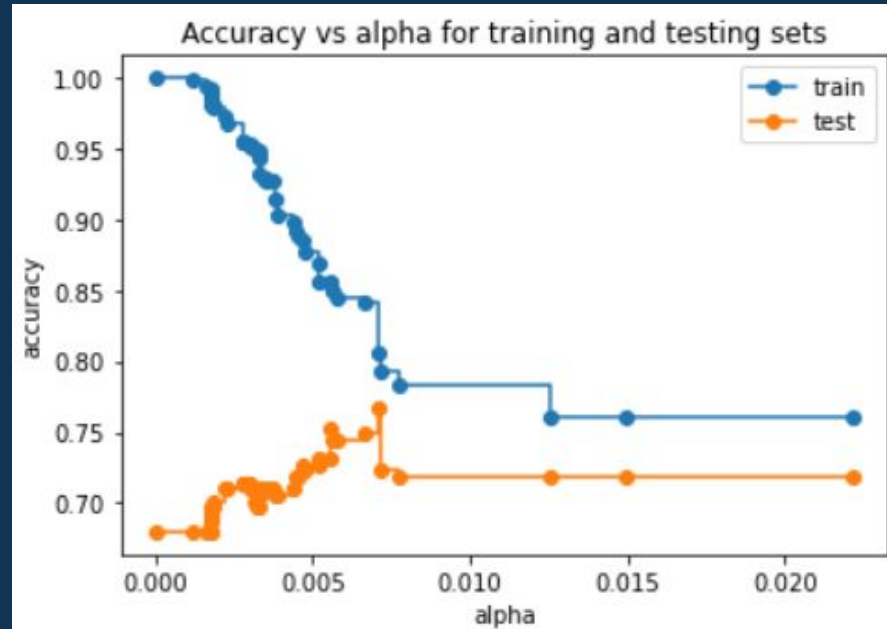
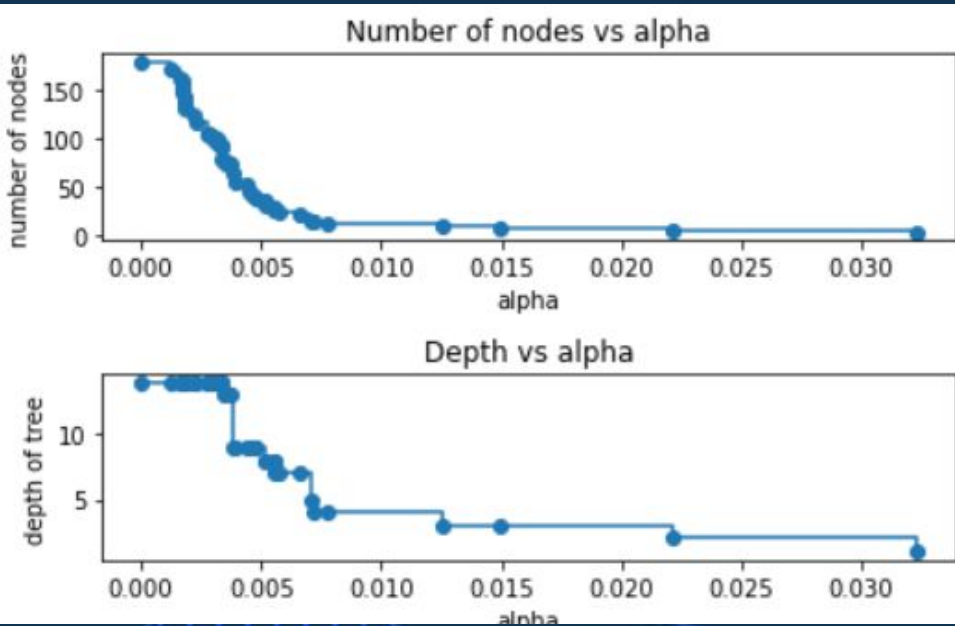
Testing accuracy increased
from **70%** to **78%**

max_depth = 5



Post-pruning

Cost Complexity Pruning: find right parameter for alpha, improve test accuracy, get better model.



Post-pruning

Choose `ccp_alpha = 0.020`

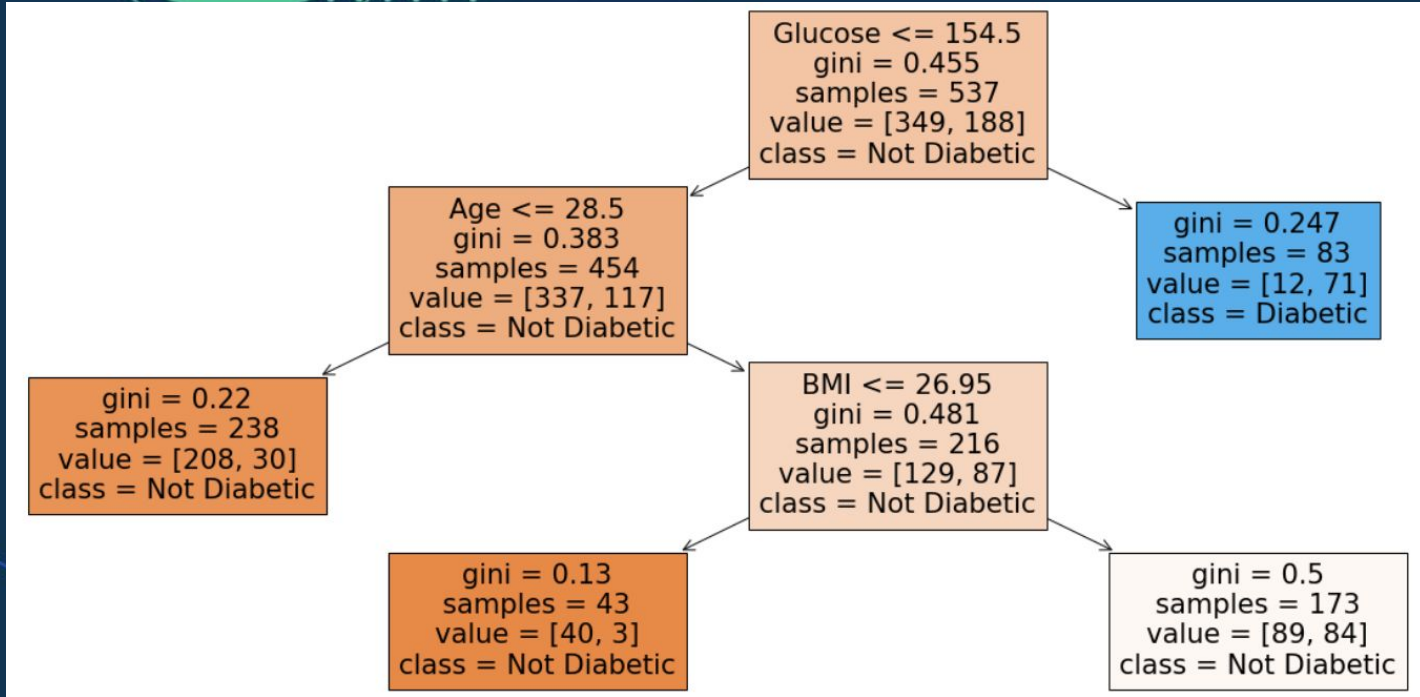
```
clf_ = tree.DecisionTreeClassifier(random_state=42, ccp_alpha=0.020)
clf_.fit(X_train, y_train)
y_train_pred = clf_.predict(X_train)
y_test_pred = clf_.predict(X_test)

# Model Accuracy, how often is the classifier correct
print("Training Accuracy:", accuracy_score(y_train, y_train_pred))
print("Testing Accuracy:", accuracy_score(y_test, y_test_pred))
```

```
Training Accuracy: 0.7597765363128491
Testing Accuracy: 0.7186147186147186
```

	precision	recall	f1-score
Not Diabetic	0.72	0.92	0.81
Diabetic	0.69	0.34	0.45

Post-pruning



Pruned model accuracy does not improve that much, but the benefit from a reduction in complexity could easily outweigh the reduction in accuracy. Making a model less complex reduces the chances that it overfits and as a result does not generalise to unseen data.

Summary



Key Takeaways from Decision Trees

Advantages of the Decision Tree:

1. Simple to understand, follows human decision-making.
2. Handy for solving decision-related problems and exploring all the possible outcomes for a problem.
3. Less data cleaning compared to other algorithms.

Disadvantages of the Decision Tree:

1. Contains lots of layers, which can make it complex.
2. Overfitting issue, can be resolved. Also use **Random Forest algorithm**.
3. For more class labels, computational complexity increases.

Next
Lecture



Further Resources

- ❖ <https://scikit-learn.org/stable/modules/tree.html>
- ❖ <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- ❖ <https://www.geeksforgeeks.org/decision-tree/>
- ❖ <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>
- ❖ <https://www.kdnuggets.com/2022/09/decision-tree-pruning-hows-why.html>

Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

