CoGrammar

Machine Learning

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- 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 Open Classes. You can submit these questions here: **Open Class Questions**

Data Science Lecture 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**

Lecture Objectives

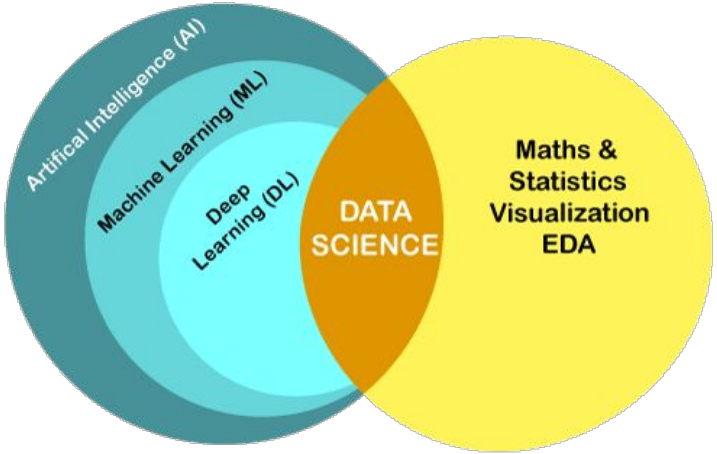
- Distinguish between **supervised, unsupervised, and semi-supervised learning** by identifying key characteristics and application scenarios, directly applying this knowledge to categorise real-world problems.

Lecture Objectives

- **Select and justify machine learning algorithm**s for various case studies, with an emphasis on understanding the **suitability of regression or classification approaches** based on dataset analysis.

Machine Learning

★ **Machine Learning (ML)** is a subset of artificial intelligence (AI) that enables systems to learn and improve from experience **without being explicitly programmed**.



Machine Learning

- ML powers many of today's advanced technologies, from **recommendation systems** to **self-driving cars**, enhancing decision-making and creating new opportunities for innovation.

Types of Machine Learning

- **Supervised Learning:** Learning a function that **maps an input to an output** based on example input-output pairs.
- **Unsupervised Learning:** Uncovering hidden patterns from data without any explicit instructions on what to look for.
- **Semi-supervised Learning:** Combines a small amount of labeled data with a large amount of unlabeled data during training.

Supervised vs. Unsupervised vs. Semi-Supervised Learning

★ **Key Characteristics:**

- Supervised learning uses **labeled datasets** to train algorithms.
- Unsupervised learning deals with data **without historical labels**.
- Semi-supervised learning uses **both labeled and unlabeled data** for training.

Supervised vs. Unsupervised vs. Semi-Supervised Learning

- **Application Scenarios:**
  - **Supervised learning:** Spam detection, image recognition.
  - **Unsupervised learning:** Market basket analysis, customer segmentation.
  - **Semi-supervised learning:** Large-scale image classification with limited labeled data.

Decision Trees with the Iris Dataset (Supervised Learning)

- ★ Decision trees classify data by learning **decision rules** inferred from features.
- ★ **Objective:** Learn how to classify species of Iris flowers based on sepal and petal measurements.

- Libraries Used:
  - **pandas and numpy** for data manipulation.
  - **sklearn.datasets** for accessing the Iris dataset.
  - **sklearn.model_selection** for data splitting.
  - **sklearn.tree** for decision tree algorithms and visualization.
  - **matplotlib.pyplot** for plotting.

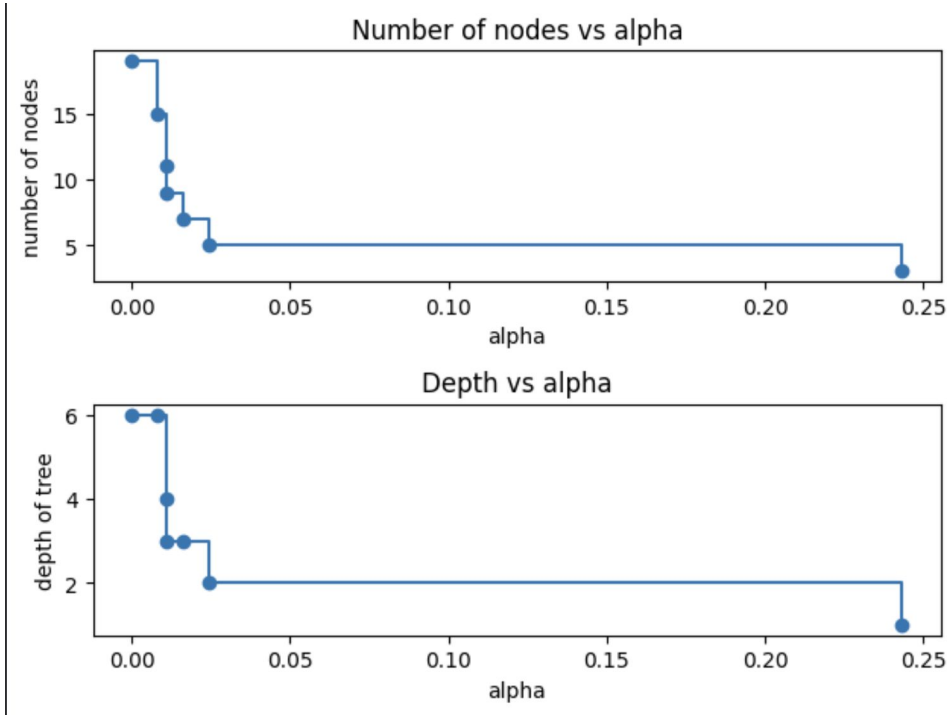
- Dataset: Iris dataset features and targets.
  - Features: Sepal length, sepal width, petal length, petal width.
  - Targets: Species of Iris (setosa, versicolor, virginica).

- Splitting Data
  - **Objective:** Prepare data for model training and evaluation.
  - **Method:** Use train_test_split to divide the dataset into training and testing sets.
  - **Split Ratio:** 80% training data, 20% testing data, with a consistent split using random_state=42.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Creating and Training the Model
  - **Model:** Decision Tree Classifier without a maximum depth to fully grow the tree.
  - **Training:** The model learns to classify Iris species based on the training dataset.

```
# Create and Train the Model with no max_depth limit to make the full tree
clf_full = DecisionTreeClassifier(random_state=42)
clf_full.fit(X_train, y_train)
```

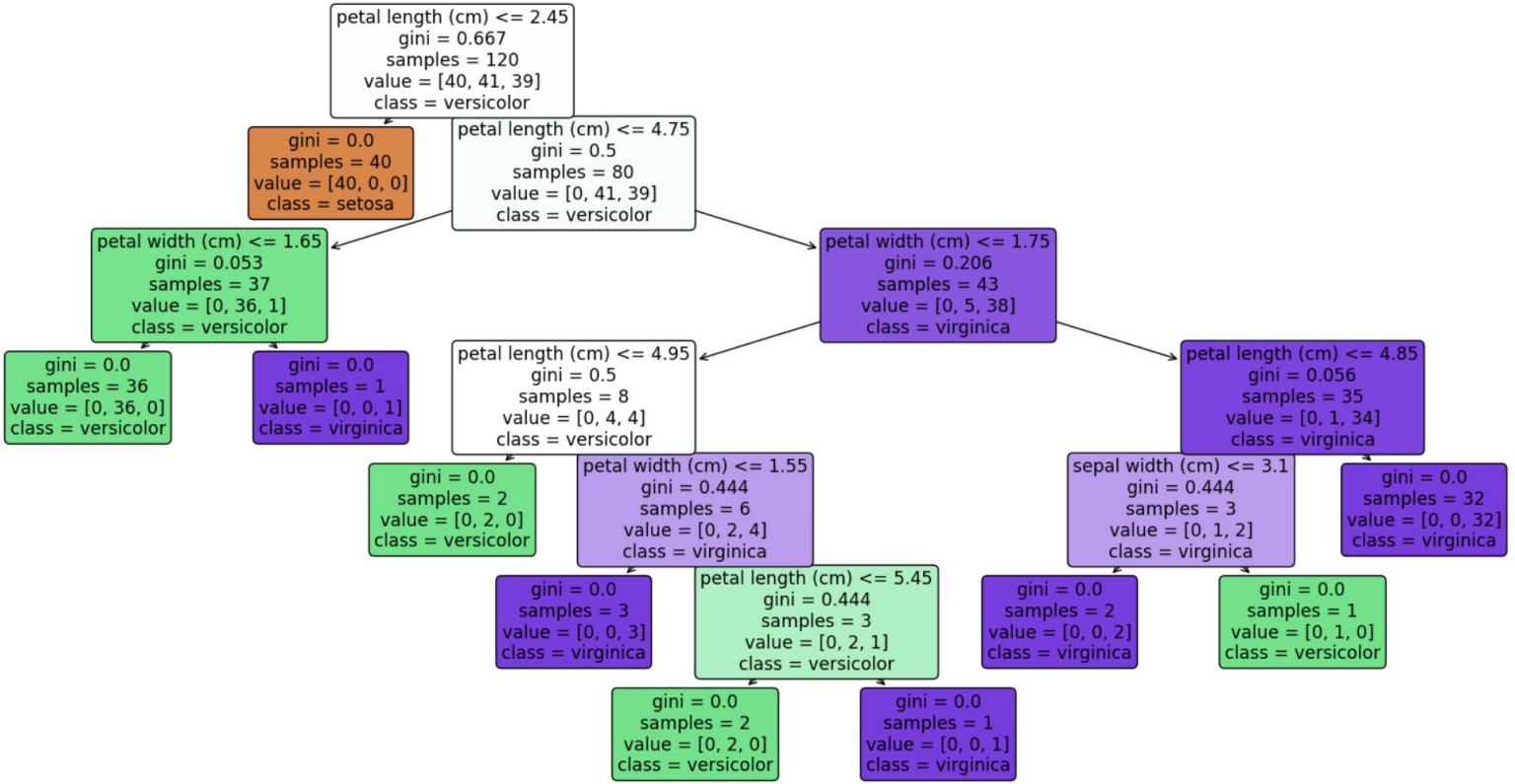


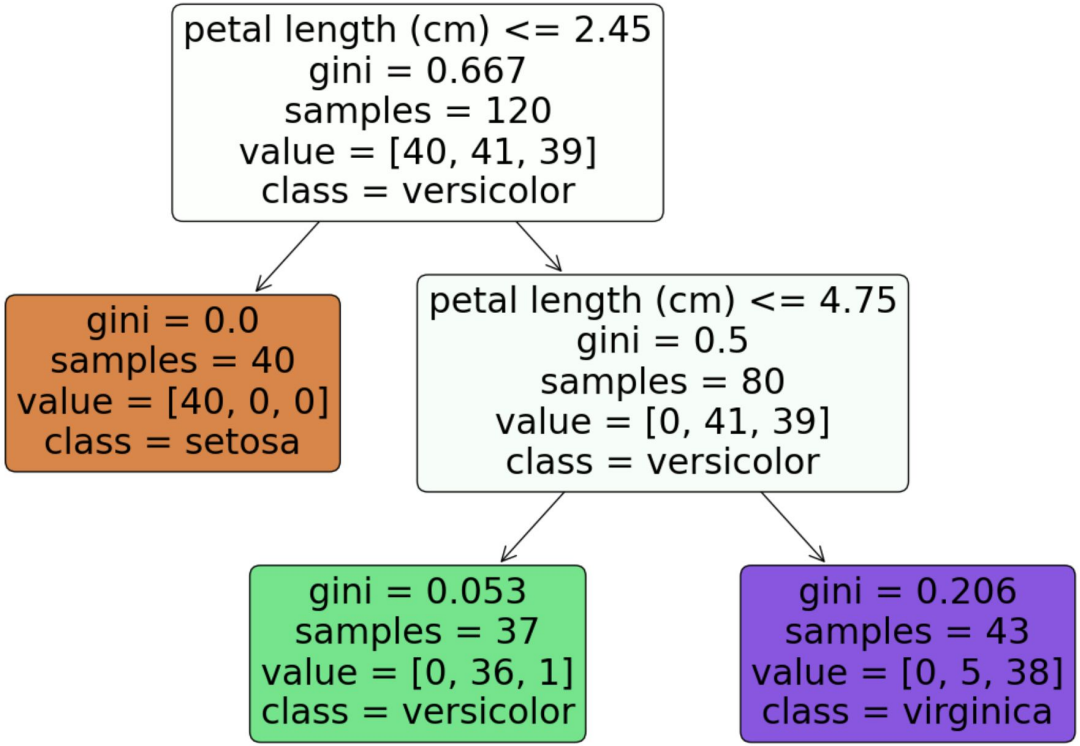
Pruning the Tree

- ★ **Purpose:** Reduce the complexity of the decision tree to prevent overfitting.
- ★ **Method:** Apply cost complexity pruning to find the optimal balance between tree depth and accuracy.

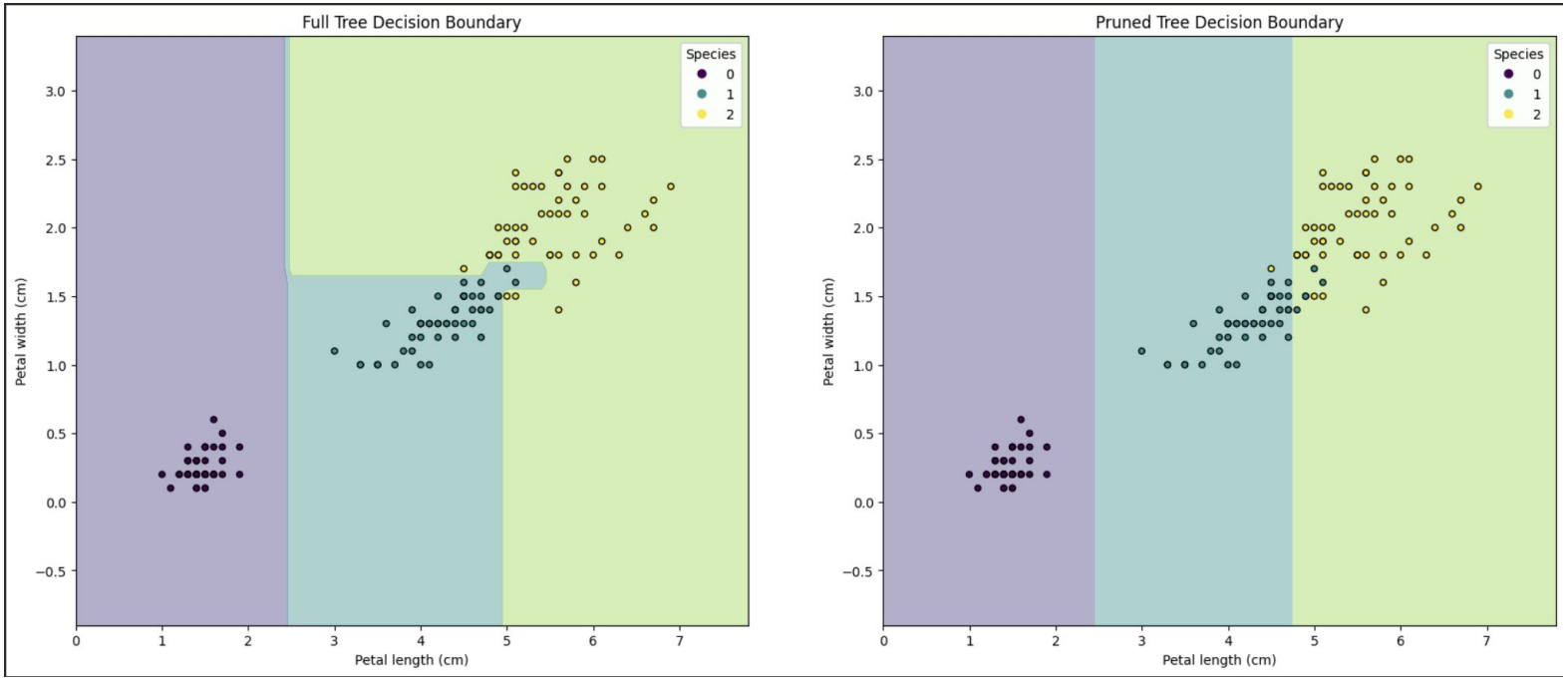
- Visualizing the Decision Tree
  - **Visualization:** Use tree.plot_tree to visually represent the decision tree's structure.
  - **Features:** Display how decisions are made based on feature values.

```
# Visualize the Decision Tree
plt.figure(figsize=(20,10))
tree.plot_tree(
    clf_full,
    filled=True,
    rounded=True,
    feature_names=iris.feature_names,
    class_names=iris.target_names
)
plt.show()
```


petal length (cm) <= 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal length (cm) <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
petal width (cm) <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
petal width (cm) <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
class = virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal length (cm) <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = versicolor
petal length (cm) <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) <= 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica


petal length (cm) <= 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal length (cm) <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
gini = 0.206
samples = 43
value = [0, 5, 38]
class = virginica

- Decision Boundaries Visualization
  - **Objective:** Show how the decision tree classifies Iris species based on petal length and width.
  - **Comparison:** Display decision boundaries for both the full and pruned trees to highlight the **effect of pruning on model complexity and generalization**.


Full Tree Decision Boundary
Species
0
1
2
Petal width (cm)
Petal length (cm)
Pruned Tree Decision Boundary
Species
0
1
2
Petal width (cm)
Petal length (cm)

★ Model Evaluation
- **Accuracy Measurement:** How well the model predicts Iris species on test data.
- **Evaluation Metrics:** Use accuracy score, confusion matrix, and classification report for detailed analysis.

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Accuracy and f1-scores are good for the full model. Too good perhaps?

This could indicate overfitting.

```
Accuracy: 0.9666666666666667
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

The accuracy of the pruned model might be slightly less, 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 generalize to unseen data.

Regression Analysis

- Regression analysis is a **statistical method** for **modelling the relationship between a dependent variable and one or more independent variables**.

- The main goal is to understand **how the typical value of the dependent variable changes when any one of the independent variables is varied** while the other independent variables are held fixed.

Let’s Breathe!

Let’s take a small break before moving on to the next topic.

CoGrammar

Linear Regression

- A linear approach to modelling the relationship between a scalar response and one or more explanatory variables.
- One use case is **predicting housing prices** based on various features such as size, location, and number of bedrooms.

Simple Linear Regression

- ★ Models the relationship between one independent variable and one dependent variable.

- ★ Equation: $y = \beta_0 + \beta_1 x + \varepsilon$
  - y: Dependent variable
  - x: Independent variable
  - β_0: Intercept
  - β_1: Slope (coefficient)
  - ε: Error term

- ★ Example: Predicting house prices based on the size of the house.

Multiple Linear Regression

★ Models the relationship between multiple independent variables and one dependent variable.

★ Equation: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$
- y: Dependent variable
- $x_1, x_2, \ldots, x_n$: Independent variables
- β_0: Intercept
- $\beta_1, \beta_2, \ldots, \beta_n$: Slopes (coefficients)
- ε: Error term

★ Example: Predicting house prices based on the size of the house, number of bedrooms, and location.

Key Differences

- Number of independent variables:
  - Simple linear regression has only one independent variable.
  - Multiple linear regression has two or more independent variables.

- Model complexity:
  - Simple linear regression is less complex and easier to interpret.
  - Multiple linear regression is more complex and may require more data and computational resources.

Key Differences

- Capturing relationships:
  - Simple linear regression can only capture the linear relationship between one independent variable and the dependent variable.
  - Multiple linear regression can capture the linear relationships between multiple independent variables and the dependent variable.

Housing Price Prediction Example

★ **Objective:** Use linear regression to predict housing prices based on various housing features.

★ **Key Concepts:**

- **Feature Selection:** Identifying which features most significantly impact the prediction.
- **Multiple Regression Model:** Utilizes multiple independent variables to predict an outcome.
- **Model Evaluation:** Employ metrics like R-squared and Mean Squared Error (MSE) to assess model performance.

Housing Price Prediction Example

★ **R-Squared - Coefficient of Determination**

- R^2 measures the **proportion of the variance** in the dependent variable that is predictable from the independent variables.
- A value of 0 indicates that the model **explains none of the variability of the response data around its mean**.
- A value of 1 indicates that the model **explains all the variability of the response data around its mean**.
- R^2 is **particularly useful in linear regression to assess the goodness of fit of the model**.

Housing Price Prediction Example

★ **Mean Squared Error (MSE)**

- MSE measures the average of the squares of the errors—that is, **the average squared difference between the estimated values and the actual value**.
- $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ where Y_i are the actual values and Y_i hat are the predicted values.
- **A lower MSE indicates a better fit of the model to the data**. It is always non-negative, and values closer to zero are ideal.

Housing Price Prediction Example

1. **Data Preparation:**
   a. Load the dataset.
   b. Preprocess data by dropping irrelevant features.
2. **Feature Selection:**
   a. Select features believed to influence housing prices.
3. **Splitting Dataset:**
   a. Divide the data into training and testing sets for model evaluation.
4. **Model Training:**
   a. Train a Linear Regression model on the training set.
5. **Evaluation:**
   a. Use MSE and R-squared values to evaluate model accuracy.

Housing Price Prediction Example

★ After following the process (which you could follow along with in the code alongside this lecture) we get:

```
Mean Squared Error: 10068422549.4956
R-squared: 0.9146818498916276
```

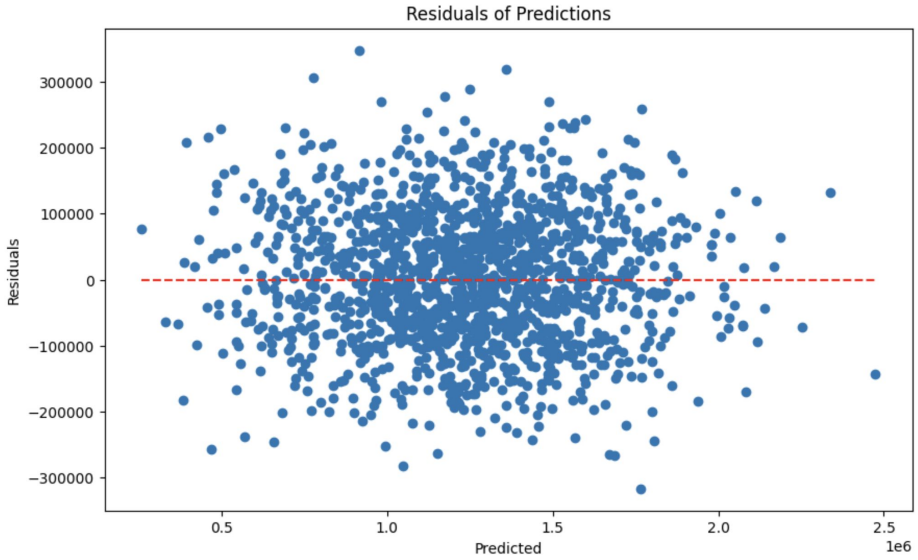
★ Here we see quite a **high MSE**, although it also has quite a **high R-squared** value, indicating that much of the variance can be explained by our current modeling - meaning it could potentially have a high goodness of fit.

Housing Price Prediction Example

★ One way to verify that our predictions are close to the actual values is plotting them against each other. If the line is straight it indicates a very good model.

Housing Price Prediction Example

★ Another valuable graph is looking at the residuals, which should be random for the model to be fitted well. In this case we can't see a clear pattern, suggesting a good model.

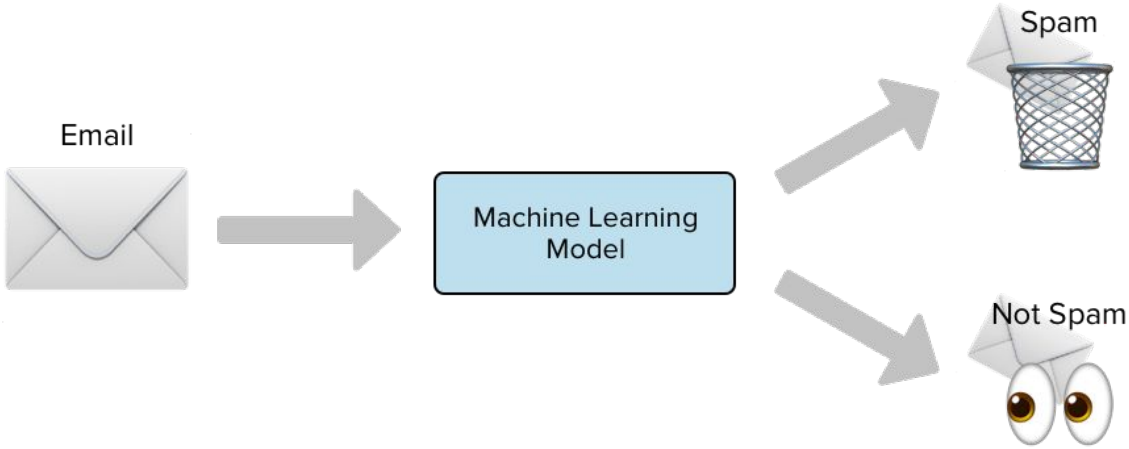


Key Takeaways from Regression

- ★ Linear regression provides a simple yet powerful tool for **predicting quantitative outcomes**.
- ★ **Applications:** From real estate to finance, linear regression plays a crucial role in predictive analytics.
- ★ **Limitations:** While useful, it **assumes a linear relationship between variables which may not always hold** in real-world scenarios.

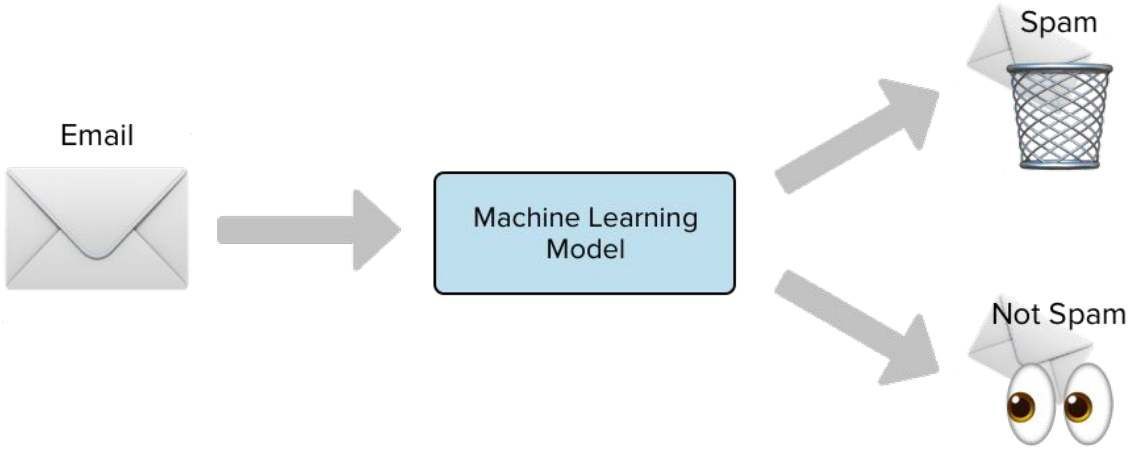
Binary Classification

★ Binary Classification involves **categorizing data into two distinct groups**. For example, distinguishing between spam and non-spam emails.



Logistic Regression

★ Used for binary classification to **predict the probability that a given data entry belongs to one of two categories**.



Spam Email Classification Example

- **Objective:** Utilize logistic regression to classify emails.
- **Key Concepts:**
  - **Sigmoid Function:** Maps predictions to probabilities.

Spam Email Classification Example

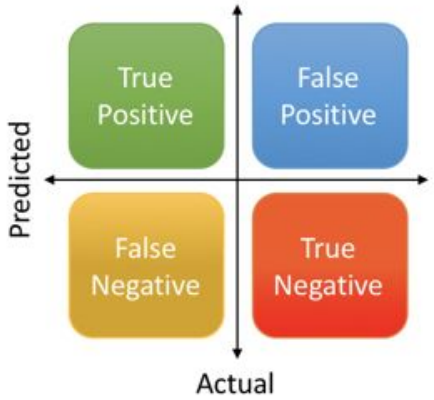
1. **Data Loading:**
   a. Import email dataset.
2. **Text Preprocessing:**
   a. Convert email texts into a numerical format using TF-IDF.
3. **Model Training:**
   a. Train a Logistic Regression model.
4. **Evaluation:**
   a. Assess model performance using accuracy, precision, recall, and ROC AUC.

Spam Email Classification Example

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



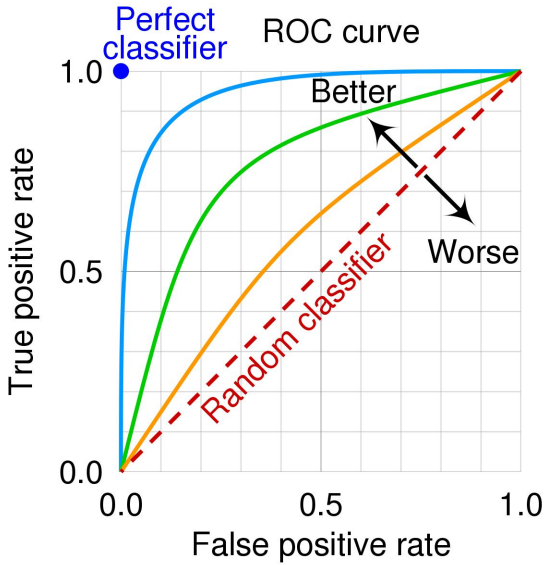
Spam Email Classification Example

- **Precision**
  - **Definition:** The ratio of correctly predicted positive observations to the **total predicted positive observations.** Also known as Positive Predictive Value.
  - **Interpretation:** High precision **indicates a low rate of false positives**. It's crucial when the cost of false positives is high.

Spam Email Classification Example

- **Recall (Sensitivity)**
  - **Definition:** The ratio of correctly predicted positive observations to all observations **in the actual class.** It measures the model's ability to find all the relevant cases.
  - **Interpretation:** High recall indicates a model that captures a large proportion of positive cases. It’s important when the cost of false negatives is high.

Spam Email Classification Example



Spam Email Classification Example

- **Receiver Operating Characteristic (ROC) Curve**
  - **Definition:** A plot that illustrates the diagnostic ability of a binary classifier system **as its discrimination threshold is varied.** It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
  - **Interpretation:** The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

Spam Email Classification Example

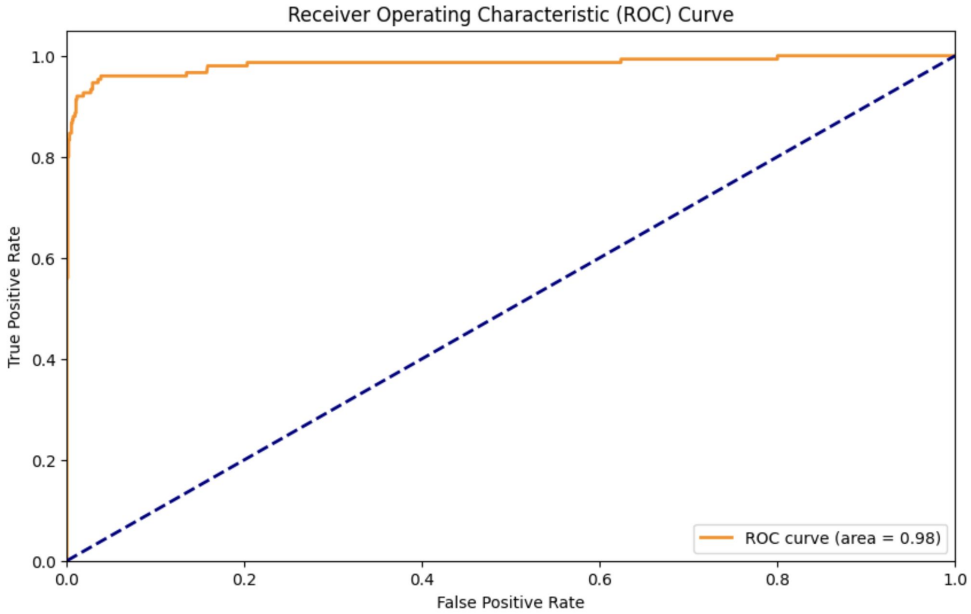
- **Area Under the Curve (AUC)**
  - **Definition:** A summary measure of the ROC curve. It represents the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative one.
  - **Interpretation:** An AUC of 1 represents a perfect model; an AUC of 0.5 represents a worthless model. **Higher values indicate better model performance**.

Spam Email Classification Example

★ After training our email classifier (which you could see in the code alongside this lecture) we get:

```
Accuracy: 0.967713004484305
Precision: 0.9913793103448276
Recall: 0.7666666666666667
ROC AUC: 0.984
```

Spam Email Classification Example

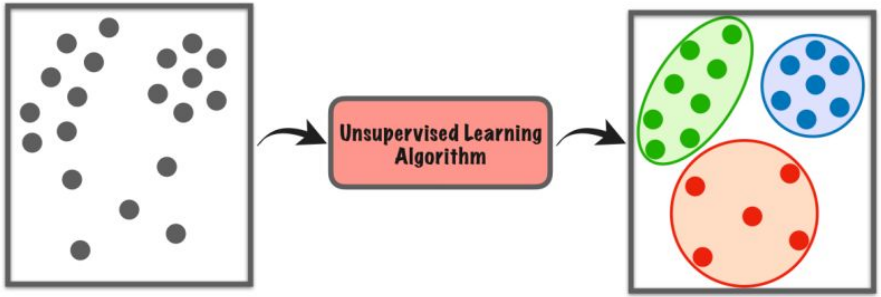


We also get the following ROC curve with an AUC of 0.98.

These metrics suggest that we have a model that makes accurate predictions.

Unsupervised Learning

- ★ **Definition:** Unsupervised learning involves drawing inferences from datasets without labeled responses.
- ★ **Objective:** The main goal is to model the underlying structure or distribution in the data to learn more about the data itself.



K-Means Clustering

- A method for **partitioning n observations into k clusters,** where each observation belongs to the cluster with the nearest mean.
- Use Case: **Identifying inherent groups within unlabeled data**, such as customer segmentation for marketing strategies.
- Before applying K-Means, **acquiring a suitable dataset is necessary.** For illustration, we use make_blobs from sklearn.datasets to generate a synthetic dataset mimicking real-world data.

Quick Detour: Feature Scaling

- Feature scaling is a crucial preprocessing step in many machine learning algorithms. It **ensures that all features contribute equally to the model, enhancing the performance and stability of the algorithm.**
- **Challenge:** Features on different scales can distort the distance metrics, **leading to biased results in models** relying on distance calculations.
- **Solution:** Scaling methods, such as standardization and normalization, **adjust features to a common scale.**

Standardization

- Standardization transforms data to have **a mean of zero and a standard deviation of one.**
- **Formula:** $z = \frac{x-\mu}{\sigma}$
  - x: original value
  - μ: mean of the feature
  - σ: standard deviation of the feature
- Standardization does not bound values to a specific range, making it suitable for algorithms not sensitive to the magnitude of values.

Standardization

- ★ **Effectiveness:** Particularly useful in optimization algorithms, e.g., gradient descent, where it accelerates convergence.
- ★ **Applications:** Essential for algorithms like Support Vector Machines (SVM) and k-nearest neighbors (k-NN).

Normalization (Min-Max Scaling)

★ **Normalization rescales the data into a fixed range**, typically [0, 1].

★ **Formula:** $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$

- x: original value
- $\min(x), \max(x)$: minimum and maximum values of the feature

★ Normalization is sensitive to outliers since the presence of a single outlier can reduce the range of the normalized values.

Normalization (Min-Max Scaling)

- ★ **Suitability:** Ideal for algorithms that assume data is on a bounded interval, e.g., neural networks.
- ★ **Visualization:** Enhances the interpretability of features with bounded values, making it easier to visualize data distributions.

Normalization vs Standardization

- **Considerations:**
  - **Algorithm requirements:** Some models inherently require data on a similar scale.
  - **Data distribution:** Standardization is less affected by outliers, whereas normalization provides bounded scaling.

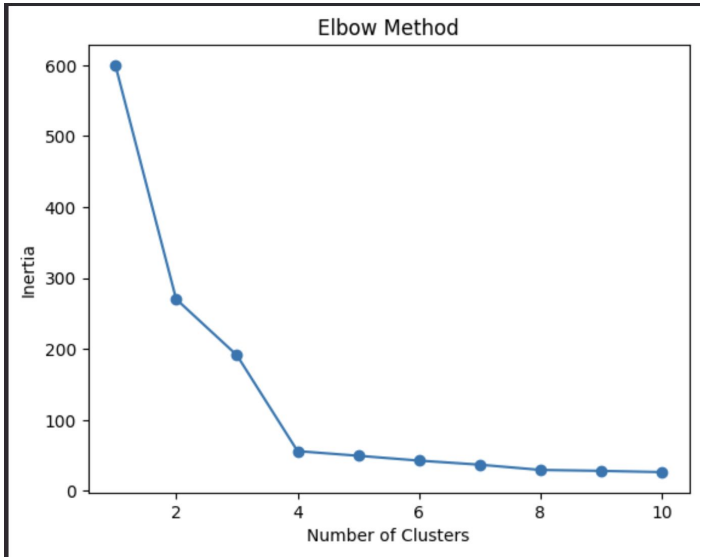
- **Best Practice:** Experiment with both methods to determine which yields better performance for your specific dataset and algorithm.

K-Means Clustering Example

- **Determine the Optimal Number of Clusters**
  - **Elbow Method:** A heuristic used to determine the optimal number of clusters by identifying the "elbow" point on a plot of the explained variation versus the number of clusters.
  - **Process:** Calculate and plot the inertia for a range of cluster numbers to find **the elbow point** indicating the recommended number of clusters.

K-Means Clustering Example

★ After implementing K-Means, which you could see in the code along with this lecture, we get:



K-Means Clustering Example

★ This result suggests that 4 clusters are optimal in our case, which we can confirm by visualising the clusters:

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining us

1. Take regular breaks
2. Stay hydrated
3. Avoid prolonged screen time
4. Practise good posture
5. Get regular exercise

"With great power comes great responsibility"