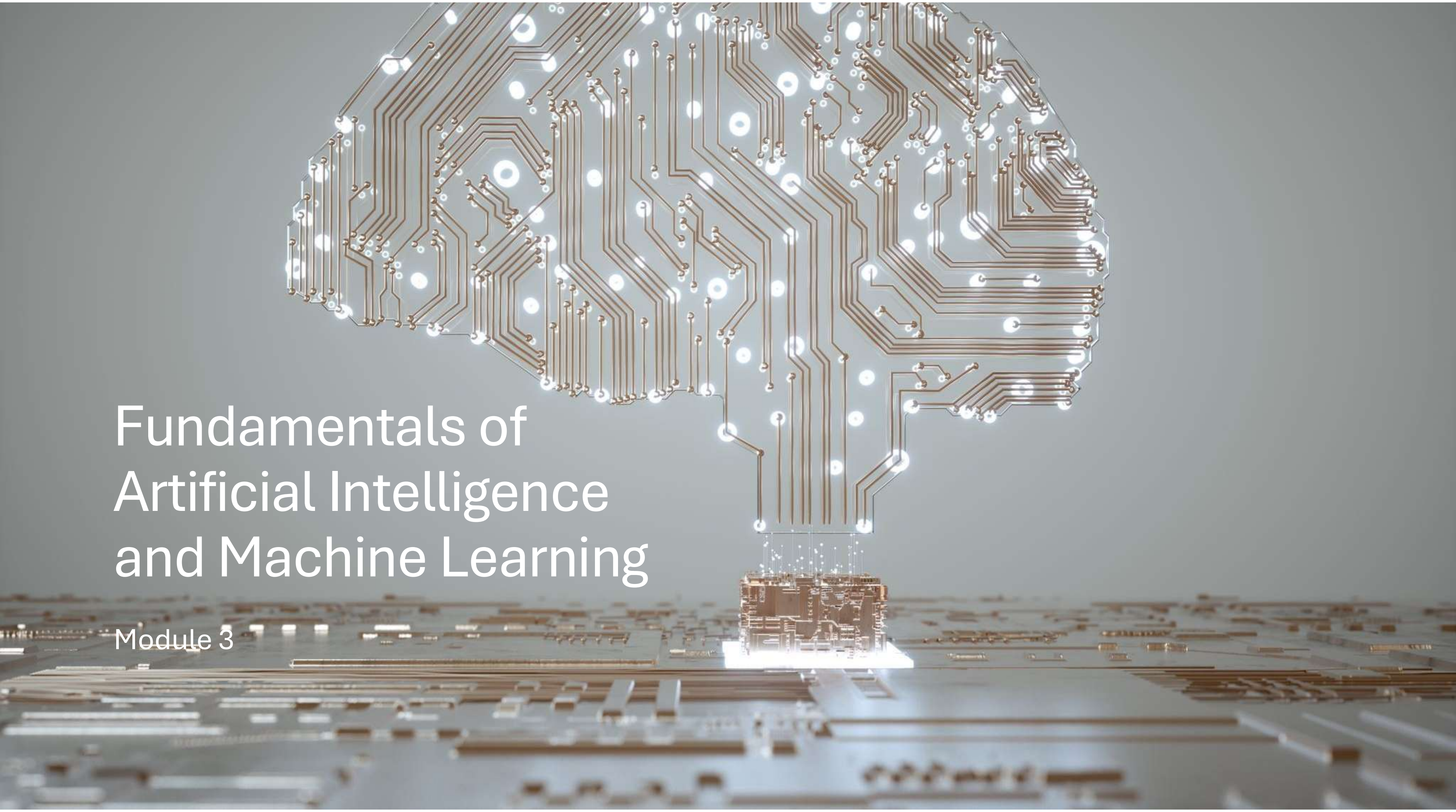


Fundamentals of Artificial Intelligence and Machine Learning

Module 3

Review of Last Class

- Review of Case Studies
  - Target’s AI-Driven Transformation
    - Years of Data Collection/Early Adoption
    - Customer Experience/Personalization
    - Supply Chain Innovation
    - Impact
  - IBM Watson
    - Natural Language Processing (NLP)
    - Programming vs Cognitive Computing
    - Watson as a product
- Supervised Learning
  - Applications and Examples
- Unsupervised Learning
  - Applications and Examples
  - Generative Adversarial Networks (GAN Algorithms)

Review of Last Class (continued)

- Regression Techniques
  - Regression (continuous values)
  - Logistic Regression (binary values)
- Customer Churn Prediction

Telephone Company Churn Prediction Recap

Cleaning the Data – College Column

- Convert "College" column to integer, so it can be categorized as a boolean

```
[4]: # Transform COLLEGE column to a numeric variable
     df["COLLEGE2"] = (df.COLLEGE == "one").astype(int)
     df.head(5)
```

Cleaning the Data – Outcome Column

- Convert the Outcome to an integer so it can be evaluated as a boolean

```
[6]: df["LEAVE2"] = (df.LEAVE == "STAY").astype(int)
     df.head(5)
```

Select Predictor Columns

- Assign the columns we want to use to make the prediction, and the target column we want to predict

```
# Names of different columns
predictor_cols = ["INCOME", "OVERAGE","LEFTOVER","HOUSE","OVER_15MINS_CALLS_PER_MONTH","AVERAGE_CALL_DURATION","COLLEGE2"]
target_col = "LEAVE2"
```

Split the Data Set

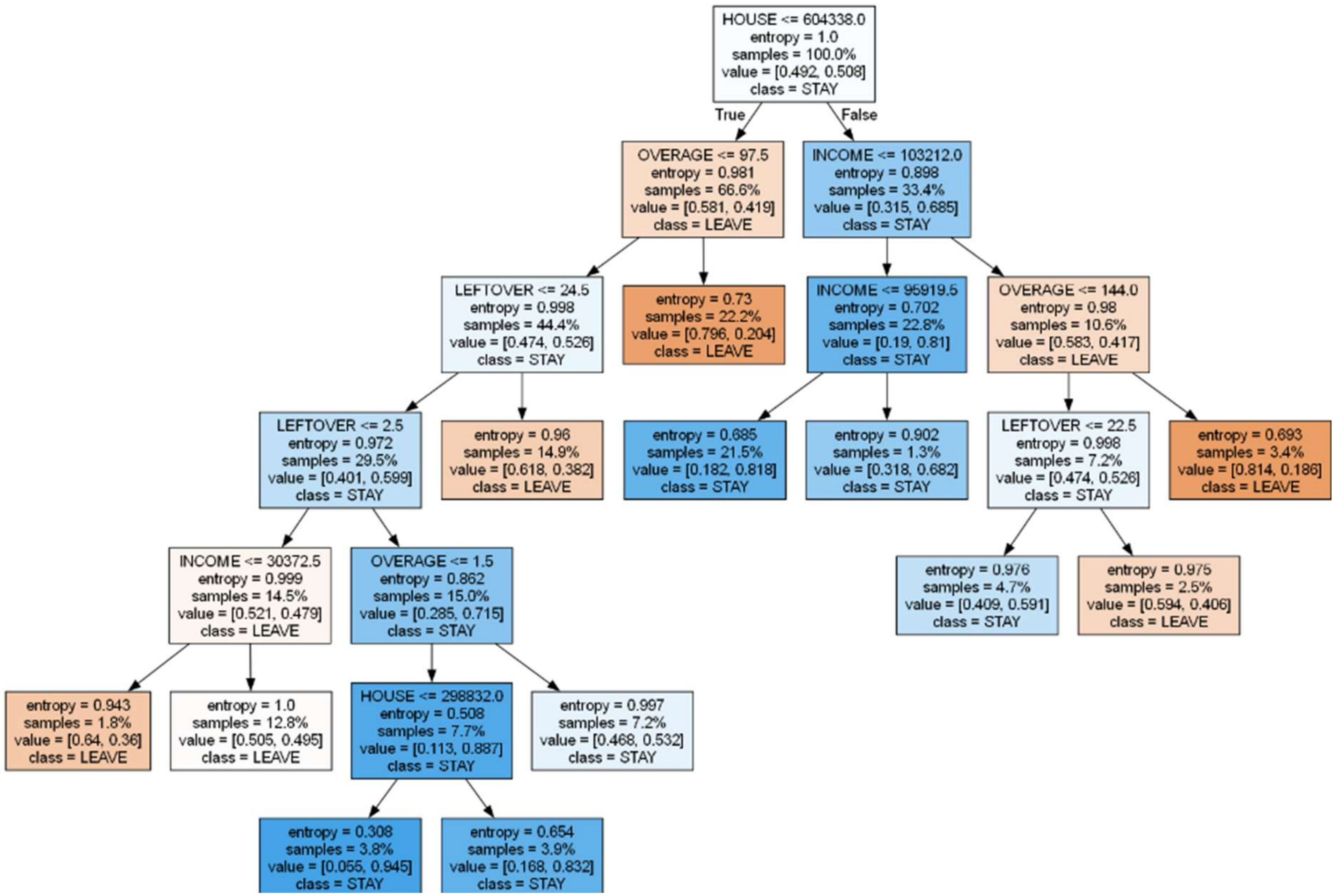
- Function **train_test_split** splits the data set into training and datasets to evaluate the performance of the machine learning model
  - Training set – used to train the machine learning model. The model learns from these data
  - Testing set – used to evaluate the performance of the trained model on unseen data.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[predictor_cols],df[target_col],test_size = 0.5,random_state = 0)
```

Fit the Model

- Define a DecisionTree
  - Define parameters about depth, leafs and criterion

[10]:
```
from sklearn.tree import DecisionTreeClassifier
# Let's define the model (tree)
decision_tree = DecisionTreeClassifier(max_depth=6, criterion="entropy",max_leaf_nodes = 12, min_samples_leaf = 1)
# Let's tell the model what is the data
decision_tree.fit(X_train, y_train)
```


HOUSE <= 604338.0
entropy = 1.0
samples = 100.0%
value = [0.492, 0.508]
class = STAY
True
False
OVERAGE <= 97.5
entropy = 0.981
samples = 66.6%
value = [0.581, 0.419]
class = LEAVE
INCOME <= 103212.0
entropy = 0.898
samples = 33.4%
value = [0.315, 0.685]
class = STAY
LEFTOVER <= 24.5
entropy = 0.998
samples = 44.4%
value = [0.474, 0.526]
class = STAY
entropy = 0.73
samples = 22.2%
value = [0.796, 0.204]
class = LEAVE
INCOME <= 95919.5
entropy = 0.702
samples = 22.8%
value = [0.19, 0.81]
class = STAY
OVERAGE <= 144.0
entropy = 0.98
samples = 10.6%
value = [0.583, 0.417]
class = LEAVE
LEFTOVER <= 2.5
entropy = 0.972
samples = 29.5%
value = [0.401, 0.599]
class = STAY
entropy = 0.96
samples = 14.9%
value = [0.618, 0.382]
class = LEAVE
entropy = 0.685
samples = 21.5%
value = [0.182, 0.818]
class = STAY
entropy = 0.902
samples = 1.3%
value = [0.318, 0.682]
class = STAY
LEFTOVER <= 22.5
entropy = 0.998
samples = 7.2%
value = [0.474, 0.526]
class = STAY
entropy = 0.693
samples = 3.4%
value = [0.814, 0.186]
class = LEAVE
INCOME <= 30372.5
entropy = 0.999
samples = 14.5%
value = [0.521, 0.479]
class = LEAVE
OVERAGE <= 1.5
entropy = 0.862
samples = 15.0%
value = [0.285, 0.715]
class = STAY
entropy = 0.976
samples = 4.7%
value = [0.409, 0.591]
class = STAY
entropy = 0.975
samples = 2.5%
value = [0.594, 0.406]
class = LEAVE
entropy = 0.943
samples = 1.8%
value = [0.64, 0.36]
class = LEAVE
entropy = 1.0
samples = 12.8%
value = [0.505, 0.495]
class = LEAVE
HOUSE <= 298832.0
entropy = 0.508
samples = 7.7%
value = [0.113, 0.887]
class = STAY
entropy = 0.997
samples = 7.2%
value = [0.468, 0.532]
class = STAY
entropy = 0.308
samples = 3.8%
value = [0.055, 0.945]
class = STAY
entropy = 0.654
samples = 3.9%
value = [0.168, 0.832]
class = STAY

Evaluate the Model

- Get the Test Set Score and accuracy score
  - Make a prediction on all of the values from the test set, and determine the accuracy of the predictions

```
[20]: y_pred = decision_tree.predict(X_test)
      print("Test set score: {: 2f}".format(np.mean(y_pred == y_test)))
```

```
Test set score:  0.696800
```

```
[21]: from sklearn import metrics
      print ( "Accuracy = %.3f" % (metrics.accuracy_score(decision_tree.predict(X_test), y_test) ))
```

```
Accuracy = 0.697
```

Make a prediction for a new customer

```
]: predictor_cols = ["INCOME", "OVERAGE","LEFTOVER","HOUSE","OVER_15MINS_CALLS_PER_MONTH","AVERAGE_CALL_DURATION","COLLEGE2"]
X_new = np.array([[90000, 100,30,500000,3, 7,1]])
def Predict_for_New_Value(X_new):
    print("X_new.shape: {}".format(X_new.shape))
    prediction = decision_tree.predict(X_new)
    print("Prediction: {}".format(prediction))
    if(prediction == 0):
        return("LEAVE")
    elif(prediction == 1):
        return("STAY")
    else:
        return("UNKNOWN STATUS..")

predicted_status = Predict_for_New_Value(X_new)
print("Predicted value for new record is %s", predicted_status)
```

```
X_new.shape: (1, 7)
Prediction: [0]
Predicted value for new record is %s LEAVE
```

Logistic Regression

- Logistic Regression
  - A type of regression used for binary classification (two possible outcomes)
  - Will the customer leave? **Yes** or **No**
- Core of the Logistic Regression is the Logistic Function (sigmoid function) which transforms any input value into a value between – and 1.

$$P(y = 1\,|X) = \frac{1}{1 + e^{-(b_0+b_1x_1+b_2x_2+\cdots+b_nx_n)}}$$

- Decision Boundary
  - Output of the function is a probability. A common threshold would be 0.5
    - If $P(y = 1\,|X) \geq 0.5$ the model is classified as 1 or "LEAVE"
    - If $P(y = 1\,|X) < 0.5$ the model is classified as 1 or "STAY"

Logistic Regression – Prepare Data

- Declare X variable which contains all columns except the prediction (“LEAVE”)
- Declare Y variable which contains the target column, which the model will predict

[53]:
```
# Define predictor (X) and target (y) columns
X = df[["COLLEGE2", "INCOME", "OVERAGE", "LEFTOVER", "HOUSE", "HANDSET_PRICE",
        "OVER_15MINS_CALLS_PER_MONTH", "AVERAGE_CALL_DURATION"]]
y = df["LEAVE"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Logistic Regression – Data Splitting

- Variable **test_train_split** will split the data into training (70%) and testing (30%) subsets.
- The training set is used to fit the model and the testing set will evaluate it's performance

```
[53]: # Define predictor (X) and target (y) columns
X = df[["COLLEGE2", "INCOME", "OVERAGE", "LEFTOVER", "HOUSE", "HANDSET_PRICE",
        "OVER_15MINS_CALLS_PER_MONTH", "AVERAGE_CALL_DURATION"]]
y = df["LEAVE"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Logistic Regression – Model Training

- Initialize the model with a maximum number of iterations to ensure it's convergence.
- Train the model using log_reg.fit() on the training data (x_train, y_train)

```
# Initialize logistic regression model
log_reg = LogisticRegression(max_iter=1000, random_state=0)

# Train the model on the training data
log_reg.fit(X_train, y_train)
```

Logistic Regression- Make a Prediction

- Make predictions on the test data given

```
# Predict on the testing data
y_pred = log_reg.predict(X_test)
```

Logistic Regression – Model Evaluation

- Calculate Accuracy – percentage of correctly predicted instances
- Confusion Matrix – Shows the number of true positives, true negatives, false positives, and false negatives
- Classification Report – Precision, Recall, F1-score, and support

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)


# Print evaluation metrics
print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

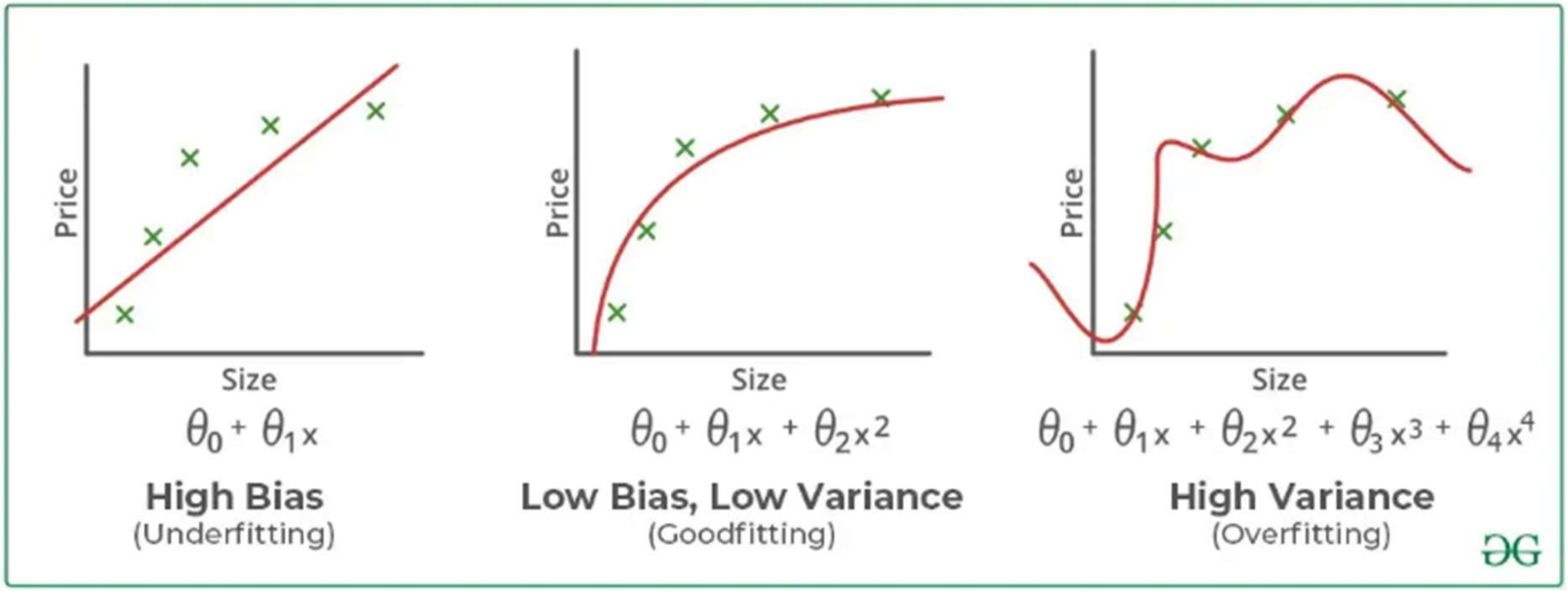
```
Accuracy: 0.64
Confusion Matrix:
[[1811 1119]
 [1013 2057]]
Classification Report:
              precision    recall  f1-score   support

       LEAVE       0.64      0.62      0.63      2930
        STAY       0.65      0.67      0.66      3070

    accuracy                           0.64      6000
   macro avg       0.64      0.64      0.64      6000
weighted avg       0.64      0.64      0.64      6000
```

Overview of Today’s Class

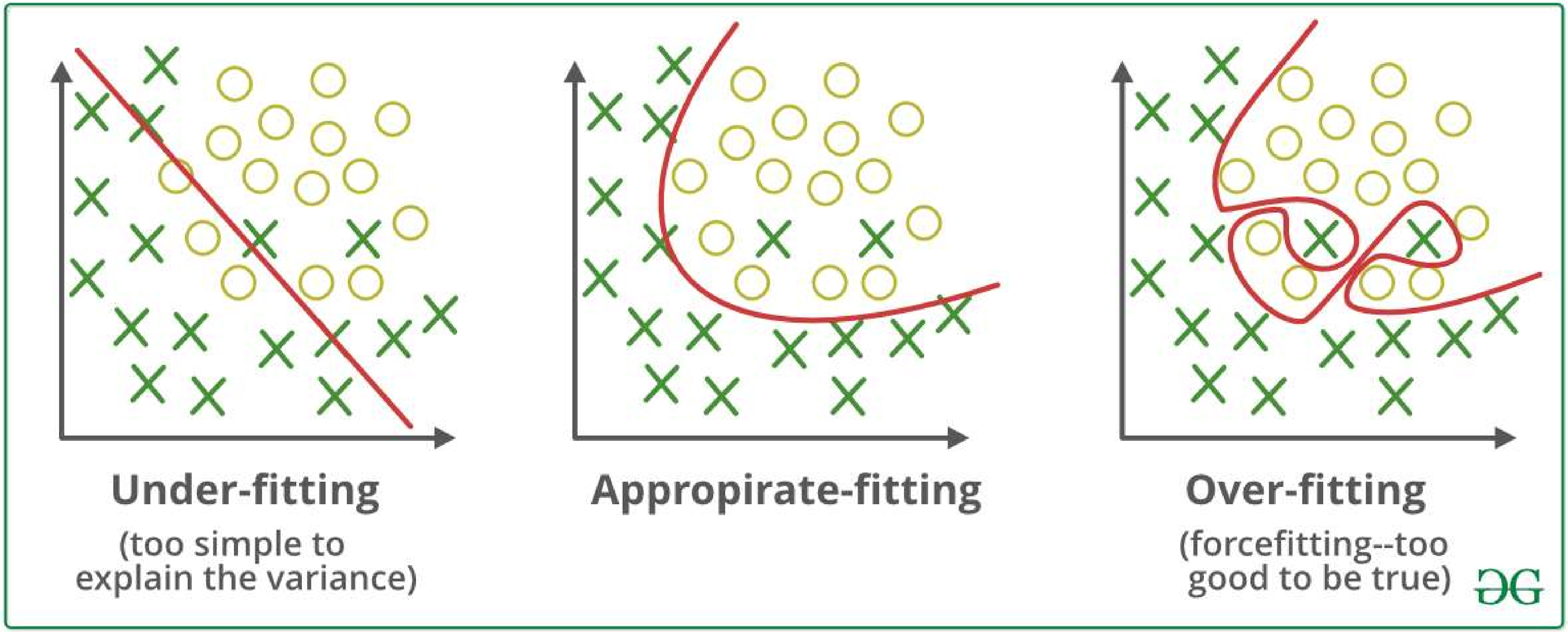
- Validating Machine Learning Models
  - Overfitting
  - Underfitting
  - Cross Validation
    - Holdout Validation
    - K-Fold Validation
- Text Mining Basics
- Everyday Applications in AI
- Ethics and Societal Impacts of AI

Validating Machine Learning Models

- Bias
  - Error due to overly simplistic assumptions in the learning algorithm. These assumptions make the model easier to comprehend and learn, but might not capture the complexities of the data. **Bias because of a simple model indicates underfitting.**
- Variance
  - Error due to model's sensitivity to fluctuations in the training data. High variance occurs when the model learns the training data's noise and fluctuations rather than the underlying pattern. **This can indicate overfitting.**

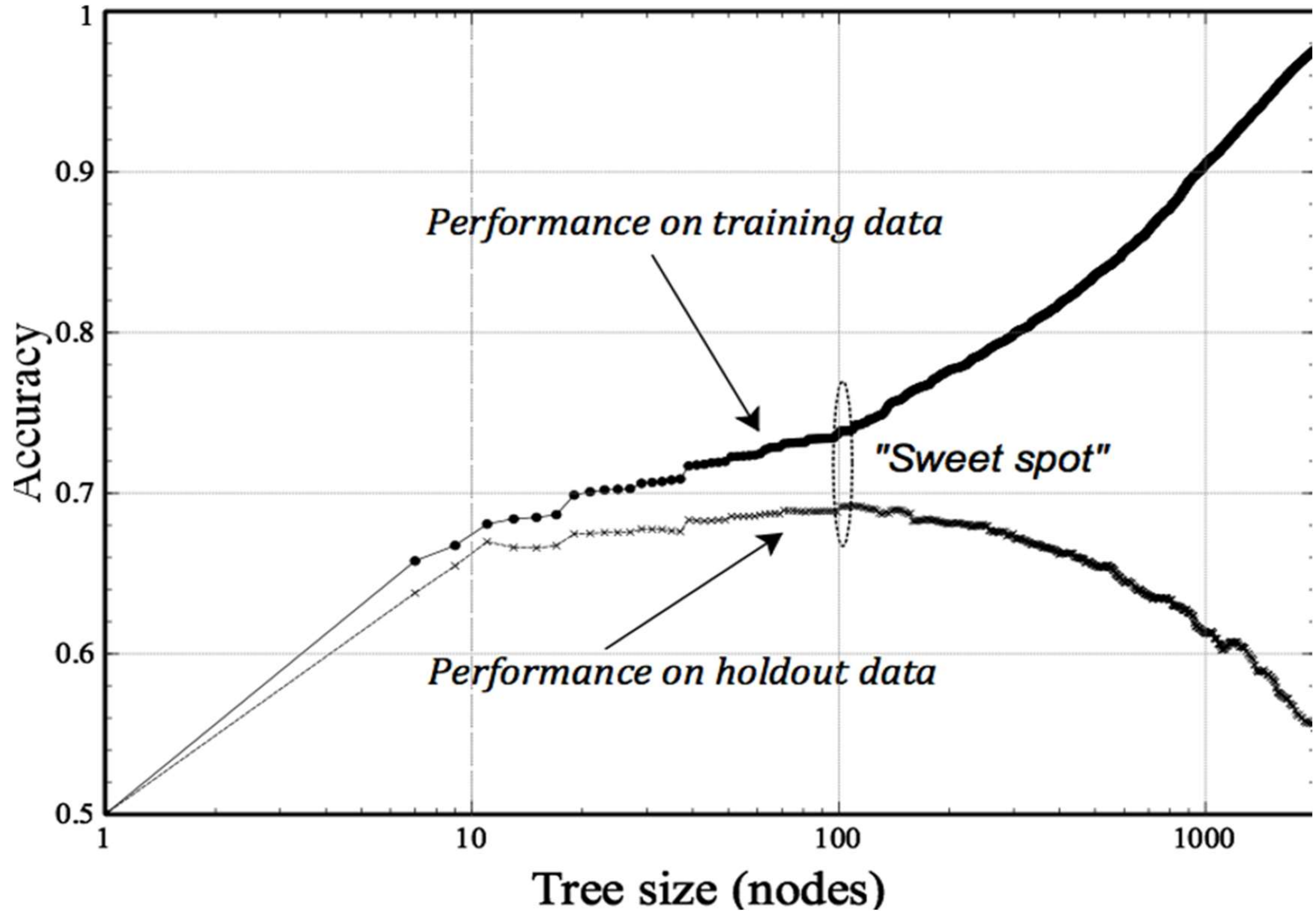


Bias and Variance


Under-fitting
(too simple to
explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too
good to be true)

How to Reduce Overfitting

- Improve quality of training data by focusing on meaningful patterns.
- Increase the amount of data to improve the model's ability to generalize to unseen data.
- Reduce model complexity
- Ridge and Lasso Regularization
- For Tree models, control the number of nodes in the tree or require minimum data points at leaf nodes
- For Regression – use regularization


Accuracy
1
0.9
0.8
0.7
0.6
0.5
1
10
100
1000
Tree size (nodes)
Performance on training data
Performance on holdout data
"Sweet spot"

Cross Validation

- Technique used to evaluate the performance of model on unseen data
- Dividing the data into multiple folds or subsets, and using one of the folds as a validation set
- The process is repeated multiple times, each using a different fold as the validation set

Model Cross-Validation Techniques

- **K-Fold Cross Validation**
- **Holdout Validation**
- Leave One Out Cross Validation (LOOCV)
- Stratified Cross-Validation

Holdout Validation

- Perform training on 50% of the given dataset and the rest of the 50% is used for testing.
- The benefit is that it is simple and quick.
- Major drawback is that the remaining 50% may contain important info which is left out, and can lead to a higher bias.

K-Fold Cross Validation

- **Step 1:** Partition the Dataset
  - Divide the dataset into *k* equally sized subsets (or folds)
  - If k = 5, split the dataset into 5 folds of equal size
- **Step 2:** Training and Validation
  - The model is trained *k* times, each using k-1 folds for training, and the rest for validation
  - Example:
    - **Run 1:** Train on folds 1, 2, 3, 4, test on 5
    - **Run 2:** Train on folds 1, 2, 3, 5, test on 4
    - **Repeat**
- **Step 3:** Evaluate Performance
  - Calculate accuracy metric for each call, and take the average

Features of K-Fold Validation

- You can choose *k*
  - Depending on the size of the dataset, and how much computational resource you have, you may want to choose different values for k.
  - Larger values for k evaluate better, but are more computationally exhaustive.
- Reduces the bias associated with a single test-train split
- Gives a more reliable estimate of model performance
- Helps detect overfitting

Text Mining

- Extracting useful information and patterns for textual data
- Subfield of data science that involves processing and analyzing text
- Examples:
  - Sentiment in social media posts, customer reviews, and news articles
- More data out there than ever before, being able to mine and analyze this data is a useful skill

Text Preprocessing

- Before analyzing text data we need to clean and prepare it.
- Remove **stopwords** (common words like “the”, “and”, “is”)
- Perform **tokenization** (splitting text into words and tokens)

Vectorization

- Machines understand numbers, not text.
- Convert the text into numerical form with two strategies
  - Bag of Words (BoW)
    - Counting the frequency of each word in a document
    - Google Cloud Tech Intro
  - TF-IDF (Term Frequency – Inverse Document Frequency)
    - Adjusts the frequency of words by considering how often they appear in the entire dataset.
    - Identify important words

Text Classification

- Now that the text is represented numerically, we can apply machine learning algorithms to classify the text into different categories
  - Naïve Bayes
  - Logistic Regression
  - Support Vector Machines
- Example Categories:
  - Spam or Non Spam
  - Positive or Negative Sentiment

Example: Determine Category

- Use the ‘sklearn.datasets.fetch_20newsgroups’ dataset
  - Collection of newsgroup documents
- Query for categories on baseball and space

•[27]:
```python
from sklearn.datasets import fetch_20newsgroups
import pandas as pd

# Load dataset (next line of code takes forever... leave commented for now)
newsgroups = fetch_20newsgroups(subset='all', categories=['rec.sport.baseball', 'sci.space'], shuffle=True, random_state=42, data_home='./newsgroups_data
data = newsgroups.data
target = newsgroups.target

# Create a DataFrame for easy manipulation
df = pd.DataFrame({'text': data, 'label': target})
df.head()
```

[27]:

	text	label
0	From: mss@netcom.com (Mark Singer)\nSubject: R...	0
1	From: cuz@chaos.cs.brandeis.edu (Cousin It)\nS...	0
2	From: J019800@LMSC5.IS.LMSC.LOCKHEED.COM\nSubj...	0
3	From: tedward@cs.cornell.edu (Edward [Ted] Fis...	0
4	From: snichols@adobe.com (Sherri Nichols)\nSub...	0

Preprocess the Data

- Use a TF-IDF to translate the text into numerical vectors

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Transform the text data to feature vectors
X = vectorizer.fit_transform(df['text'])

# Labels
y = df['label']
```

Fit the Model for Classification

- Use a Support Vector Machine model for classification of the vectorizer

```python
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and train the classifier
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

Evaluate the Model

- Evaluate the model using the accuracy score and classification report

```
[33]: from sklearn.metrics import accuracy_score, classification_report

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=newsgroups.target_names)

print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(report)
```

```
Accuracy: 0.9966
Classification Report:
                    precision    recall  f1-score   support

rec.sport.baseball       0.99      1.00      1.00       286
         sci.space       1.00      0.99      1.00       309

          accuracy                           1.00       595
         macro avg       1.00      1.00      1.00       595
      weighted avg       1.00      1.00      1.00       595
```

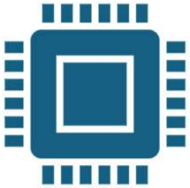
Try making a category prediction

```python
def predict_category(text):
    """
    Predict the category of a given text using the trained classifier.
    """
    text_vec = vectorizer.transform([text])
    prediction = clf.predict(text_vec)
    return newsgroups.target_names[prediction[0]]

# Example usage
sample_text = "NASA announced the discovery of new exoplanets."
predicted_category = predict_category(sample_text)
print(f'The predicted category is: {predicted_category}')
```

```
The predicted category is: sci.space
```

Conclusion – The Future of AI in Business

AI is no longer optional – it’s essential for business success



Business professionals don’t need to code, but must understand AI’s capabilities and limitations

Be a leader in adopting and applying AI effectively in your domain!

Resources

- https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
- https://www.geeksforgeeks.org/cross-validation-machine-learning/
- https://www.geeksforgeeks.org/text-classification-using-scikit-learn-in-nlp/

More Programs at CCM

- Advancing Your Career program https://www.ccm.edu/programs/advancing-your-career/
- Unemployed information https://www.ccm.edu/workforce-development/ under FAQs - Includes many helpful links for unemployed, underemployed or dislocated individuals
- All Workforce Development programs/classes https://www.ccm.edu/workforce-development/
- Grant-Supported Training program https://www.ccm.edu/programs/grant-supported-training/ - Must be employed by a N.J. non-governmental business to qualify for no-cost but could register as a paid student by contacting cbt@ccm.edu.