UNIT III – CLASSIFICATION AND ADVANCED LEARNING METHODS

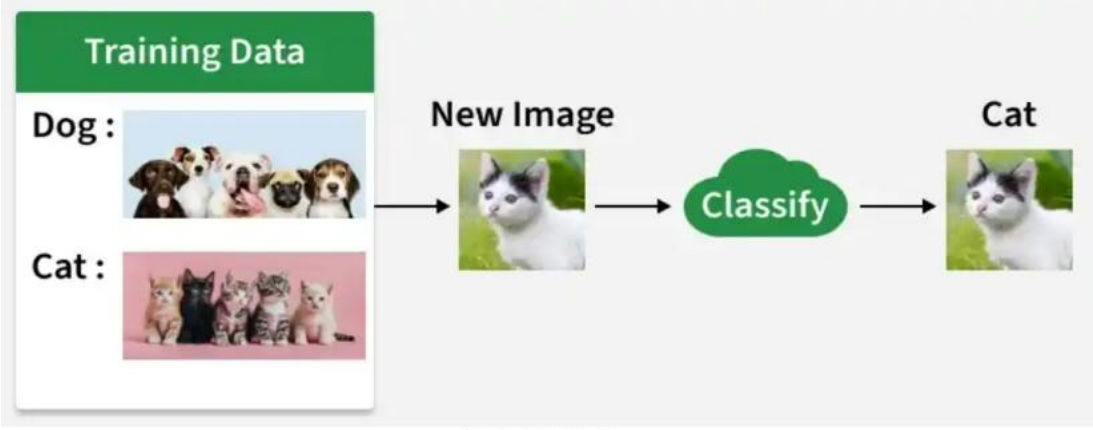
Classification fundamentals – Confusion matrix – Accuracy, Precision, Recall, F1-score, ROC-AUC – Bayesian Classifier – K-Nearest Neighbour (KNN) – Support Vector Machines (SVM) and kernels – Decision Trees – Random Forest – Ensemble Learning – Boosting (AdaBoost, XGBoost) – Multi-class classification – Neural networks basics – Transfer Learning – Introduction to Generative AI and Foundation Models.

CLASSIFICATION FUNDAMENTALS

- Classification is a supervised machine learning technique used to predict labels or categories from input data.
- It assigns each data point to a predefined class based on learned patterns.
- Predict categories: Determines the class of new data points.
- Uses labeled data: Trained on datasets where the correct class is known.
- Common examples: Spam vs non spam emails, diseased vs. healthy patients.

Example

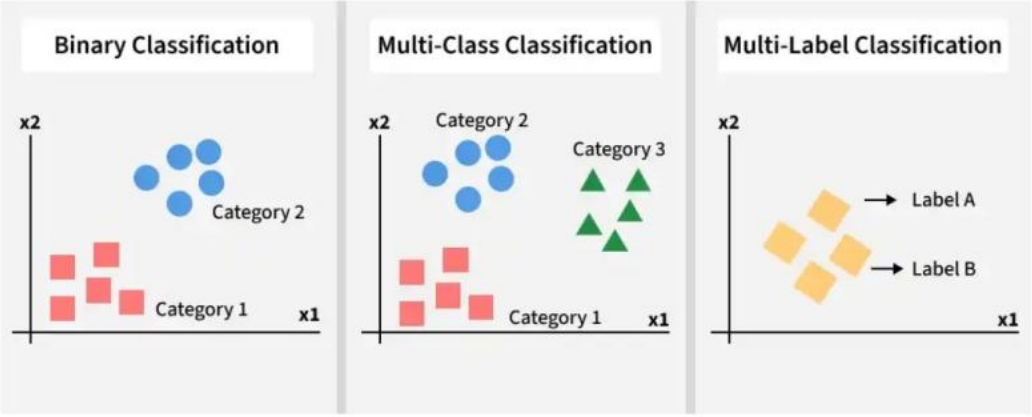
- A classification model might be trained on dataset of images labeled as either dogs or cats and it can be used to predict the class of new and unseen images as dogs or cats based on their features such as colour, texture or shape.



Cat-Dog Classification

TypesofClassification

- Classification in machine learning involves sorting data into categories based on their features or characteristics.
- The type of classification problem depends on how many classes exist and how the categories are structured.



Types of Classification

1.BinaryClassification

- Binary classification is the simplest type of classification where data is divided into two possible categories. The model analyzes input features and decides which of the two classes the data belongs to. Some important aspects of binary classification are:
- Two classes only: Each data point is assigned to one of two categories.
- Common examples: Spam vs not spam emails, diseased vs healthy patients.
- Decision based on features: The model uses input features to determine the correct class.

2.MulticlassClassification

- Multiclass classification is used when data needs to be divided into more than two categories. The model analyzes the input features and selects the class that best matches the data. Some important aspects of multiclass classification are:
- Multiple classes: Each data point is assigned to one of several possible categories.
- Single final prediction: The model selects only one class for each input.
- Common examples: Image classification such as identifying animals like cat, dog or bird.

3.MultiLabelClassification

- Multi label classification allows a single piece of data to belong to multiple categories at the same time. Unlike multiclass classification, where each data point is assigned only one class, this approach allows the model to assign multiple labels to the same input. Key aspects include:
- Multiple labels per data point: One input can belong to more than one category.
- Labels can overlap: Classes are not mutually exclusive.
- Common example: A movie recommendation system may tag a movie as both action and comedy based on features like plot, actors or genre tags.

Working

- Classification works by training a model on labeled data so it can learn patterns and predict the correct class for new inputs. The main steps involved are:
- Data Collection: Start with a dataset where each data point has a correct label.
- Feature Extraction: Identify important features such as color, shape or texture that help distinguish classes.
- Model Training: The algorithm learns patterns that connect features to the correct class.
- Model Evaluation: The trained model is tested on unseen data to measure its accuracy.
- Prediction: The model predicts the class of new data based on learned patterns.
- Model Improvement: If performance is not satisfactory, the model or its parameters are adjusted and retrained.

ClassificationAlgorithms

- To implement a classification model, it is important to understand the algorithms used for classification.
- One of the most commonly used algorithms is Logistic Regression.
- Classification algorithms can be grouped into different categories, such as:

1.LinearClassifiers

- Linear classifier models create a linear decision boundary between classes. They are simple and computationally efficient. Some of the linear classification models are as follows:
- Logistic Regression
- Support Vector Machines
- Single-layer Perceptron
- Stochastic Gradient Descent (SGD) Classifier

2.Non-linearClassifiers

- Non linear models create non linear decision boundaries to separate classes. They can capture more complex relationships between input features and the target variable. Some common non linear classification models include:
- K-Nearest Neighbours
- Kernel SVM
- Decision Tree Classification
- Random Forests
- AdaBoost
- Bagging Classifier
- Voting Classifier
- Extra Trees Classifier
- Multi layer Artificial Neural Networks

Applications

- Email Spam Filtering: Classifies emails as spam or not spam based on message content.
- Credit Risk Assessment: Predicts whether a loan applicant is likely to default using factors like credit score and income.
- Medical Diagnosis: Classifies whether a patient has diseases such as cancer or diabetes using medical data.
- Image Classification: Used in applications like facial recognition, autonomous driving and medical imaging.
- Sentiment Analysis: Determines whether text sentiment is positive, negative or neutral.
- Fraud Detection: Identifies unusual transaction patterns to detect financial fraud.
- Recommendation Systems: Not strictly classification but often uses ranking or similarity-based methods.

CONFUSION MATRIX

- Confusion matrix is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong. This helps you understand where the model is making mistakes so you can improve it. It breaks down the predictions into four categories:
- **True Positive (TP):** The model correctly predicted a positive outcome i.e the actual outcome was positive.
- **True Negative (TN):** The model correctly predicted a negative outcome i.e the actual outcome was negative.
- **False Positive (FP):** The model incorrectly predicted a positive outcome i.e the actual outcome was negative. It is also known as a Type I error.
- **False Negative (FN):** The model incorrectly predicted a negative outcome i.e the actual outcome was positive. It is also known as a Type II error.
- It also helps calculate key measures like accuracy, precision and recall which give a better idea of performance especially when the data is imbalanced

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Confusion Matrix

Metrics based on Confusion Matrix Data

1. Accuracy

- Accuracy shows how many predictions the model got right out of all the predictions.
- It gives idea of overall performance but it can be misleading when one class is more dominant over the other.
- For example a model that predicts the majority class correctly most of the time might have high accuracy but still fail to capture important details about other classes. It can be calculated using the below formula:

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

2. Precision

- Precision focus on the quality of the model's positive predictions. It tells us how many of the "positive" predictions were actually correct. It is important in situations where false positives need to be minimized such as detecting spam emails or fraud. The formula of precision is:

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

3. Recall

- Recall measures how how good the model is at predicting positives. It shows the proportion of true positives detected out of all the actual positive instances. High recall is essential when missing positive cases has significant consequences like in medical tests.

$$\text{Recall} = \frac{TP}{TP+FN}$$

4. F1-Score

- F1-score combines precision and recall into a single metric to balance their trade-off.
- It provides a better sense of a model's overall performance particularly for imbalanced datasets.
- It is helpful when both false positives and false negatives are important though it assumes precision and recall are equally important but in some situations one might matter more than the other.

$$\text{F1-Score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

5. Specificity

- Specificity is another important metric in the evaluation of classification models particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate Formula is given by:

$$\text{Specificity} = \frac{TN}{TN+FP}$$

6. Type 1 and Type 2 error

- Type 1 and Type 2 error are:
- Type 1 error: It occurs when the model incorrectly predicts a positive instance but the actual instance is negative. This is also known as a false positive. Type 1 Errors affect the precision of a model which measures the accuracy of positive predictions.

$$\text{Type 1 Error} = \frac{FP}{FP+TN}$$

- Type 2 error: This occurs when the model fails to predict a positive instance even though it is actually positive. This is also known as a false negative. Type 2 Errors impact the recall of a model which measures how well the model identifies all actual positive cases.

$$\text{Type 2 Error} = \frac{FN}{TP+FN}$$

Confusion Matrix For Binary Classification

- A 2x2 Confusion matrix is shown below for the image recognition having a Dog image or Not Dog image:
- True Positive (TP): It is the total counts having both predicted and actual values are Dog.
- True Negative (TN): It is the total counts having both predicted and actual values are Not Dog.
- False Positive (FP): It is the total counts having prediction is Dog while actually Not Dog.
- False Negative (FN): It is the total counts having prediction is Not Dog while actually, it is Dog.

	Predicted	Predicted
Actual	True Positive (TP)	False Negative (FN)
Actual	False Positive (FP)	True Negative (TN)

Example: Confusion Matrix for Dog Image Recognition with Numbers

Index	1	2	3	4	5	6	7	8	9	10
Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result	TP	FN	TP	TN	TP	FP	TP	TP	TN	TN

- Actual Dog Counts = 6
- Actual Not Dog Counts = 4
- True Positive Counts = 5
- False Positive Counts = 1
- True Negative Counts = 3
- False Negative Counts = 1

		Predicted	
		Dog	Not Dog
Actual	Dog	True Positive (TP =5)	False Negative (FN =1)
	Not Dog	False Positive (FP=1)	True Negative (TN=3)

Implementation of Confusion Matrix for Binary classification using Python

Step 1: Import the necessary libraries

```
import numpy as np
from sklearn.metrics import confusion_matrix,classification_report import seaborn as sns
import matplotlib.pyplot as plt
```

Step 2: Create the NumPy array for actual and predicted labels

actual: represents the true labels or the actual classification of the items. In this case it's a list of 10 items where each entry is either 'Dog' or 'Not Dog'.
predicted: represents the predicted labels or the classification made by the model.

```
actual = np.array(['Dog','Dog','Dog','Not Dog','Dog','Not Dog','Dog','Dog','Not Dog','Not Dog'])
predicted = np.array( ['Dog','Not Dog','Dog','Not Dog','Dog','Dog','Dog','Dog','Not Dog','Not Dog'])
```

Step 3: Compute the confusion matrix

confusion_matrix: This function from sklearn.metrics computes the confusion matrix which is a table used to evaluate the performance of a classification algorithm. It compares actual and predicted to generate a matrix

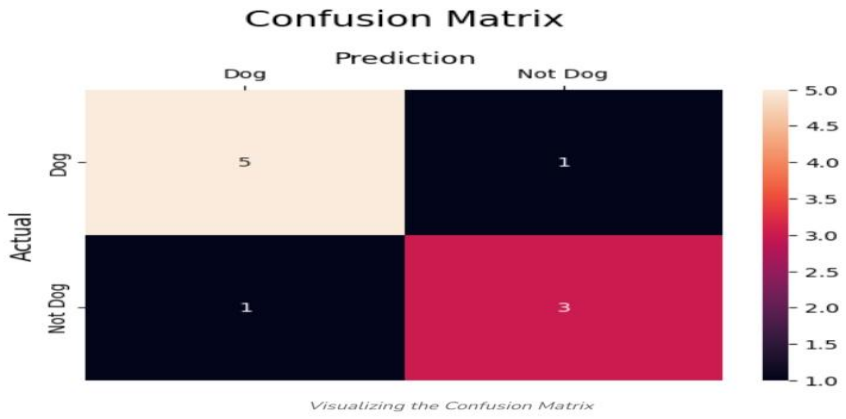
```
cm = confusion_matrix(actual,predicted)
```

Step 4: Plot the confusion matrix with the help of the seaborn heatmap

sns.heatmap: This function from Seaborn is used to create a heatmap of the confusion matrix.
annot=True: Display the numerical values in each cell of the heatmap.

```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Dog','Not Dog'],
            yticklabels=['Dog','Not Dog'])
plt.ylabel('Actual', fontsize=13)
plt.title('Confusion Matrix', fontsize=17, pad=20)
plt.gca().xaxis.set_label_position('top')
plt.xlabel('Prediction', fontsize=13)
plt.gca().xaxis.tick_top()
plt.gca().figure.subplots_adjust(bottom=0.2)
plt.gca().figure.text(0.5, 0.05, 'Prediction', ha='center', fontsize=13)
plt.show()
```

Output



Visualizing the Confusion Matrix

- Step 5: Classifications Report based on Confusion Metrics
- print(classification_report(actual, predicted))

Output:

	precision	recall	f1-score	support
Dog	0.83	0.83	0.83	6
Not Dog	0.75	0.75	0.75	4
accuracy			0.80	10
macro avg	0.79	0.79	0.79	10
weighted avg	0.80	0.80	0.80	10

Classification Report