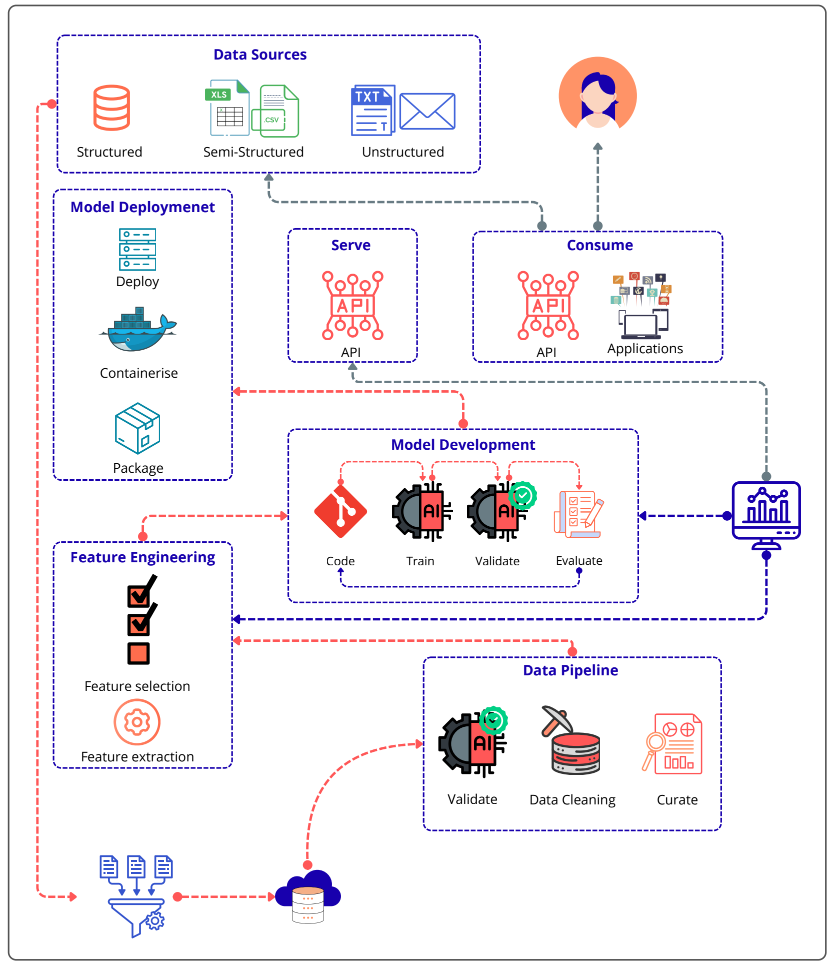
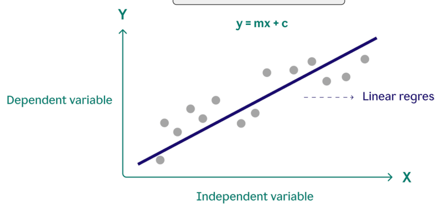
Generic ML Project Pipeline



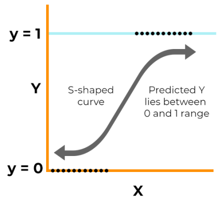
Comparing ML models (in modelling)

Comparing different ML models									
Algorithm	Type	Results interpretable by you ?	Average predictive accuracy	Amount of parameter tuning needed (excluding feature selection)	Performs well with small number of observations ?	Handles lots of irrelevant features well (seperates signal from noise) ?	Automatically learns feature interactions ?	Gives calibrated probabilities of class membership ?	Features might need scaling ?
Linear Regression	Regression	✓ Yes	Low	**None** (excluding regularization)	✓ Yes	✗ No	✗ No	N/A	✗ No
Logistic Regression	Classification	Somewhat	Low	**None** (excluding regularization)	✓ Yes	✗ No	✗ No	✓ Yes	✗ No
KNN	Either	✓ Yes	Low	Minimal	✗ No	✗ No	✗ No	✓ Yes	✓ Yes
Decision Trees	Either	Somewhat	Low	**Some**	✗ No	✗ No	✓ Yes	Possibly	✗ No
Random Forests	Either	A little	High	**Some**	✗ No	✓ Yes	✓ Yes	Possibly	✗ No

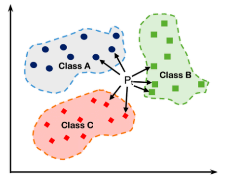
Linear Regression



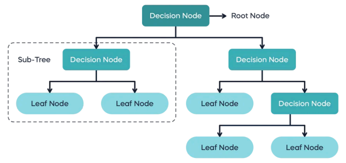
Logistic Regression



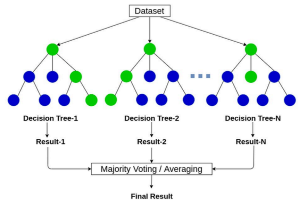
KNN



Decision Trees



Random Forests



Comparing ML models (in execution)

Comparing different ML models				
Algorithm	Parametrization	Memory size	Data quantity required	Overfitting risk
Linear Regression	Simple	Small	Small	**None** (excluding regularization)
Logistic Regression	Simple	Small	Small	**None** (excluding regularization)
KNN	Strong	Small	Small	Minimal
Decision Trees	Simple/Intuitive	Large	Large	**Some**
Random Forests	Simple/Intuitive	Very large	Large	**Some**

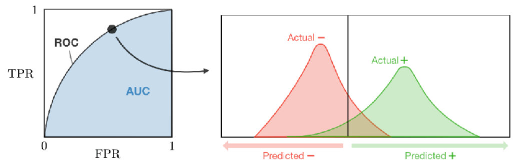
Performance metrics of ML models

In a context of a binary classification, here are the main metrics that are important to track to assess the performance of the model.

❒ **Confusion matrix** – The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:

		Predicted class	
		+	–
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	–	**FP** False Positives Type I error	**TN** True Negatives

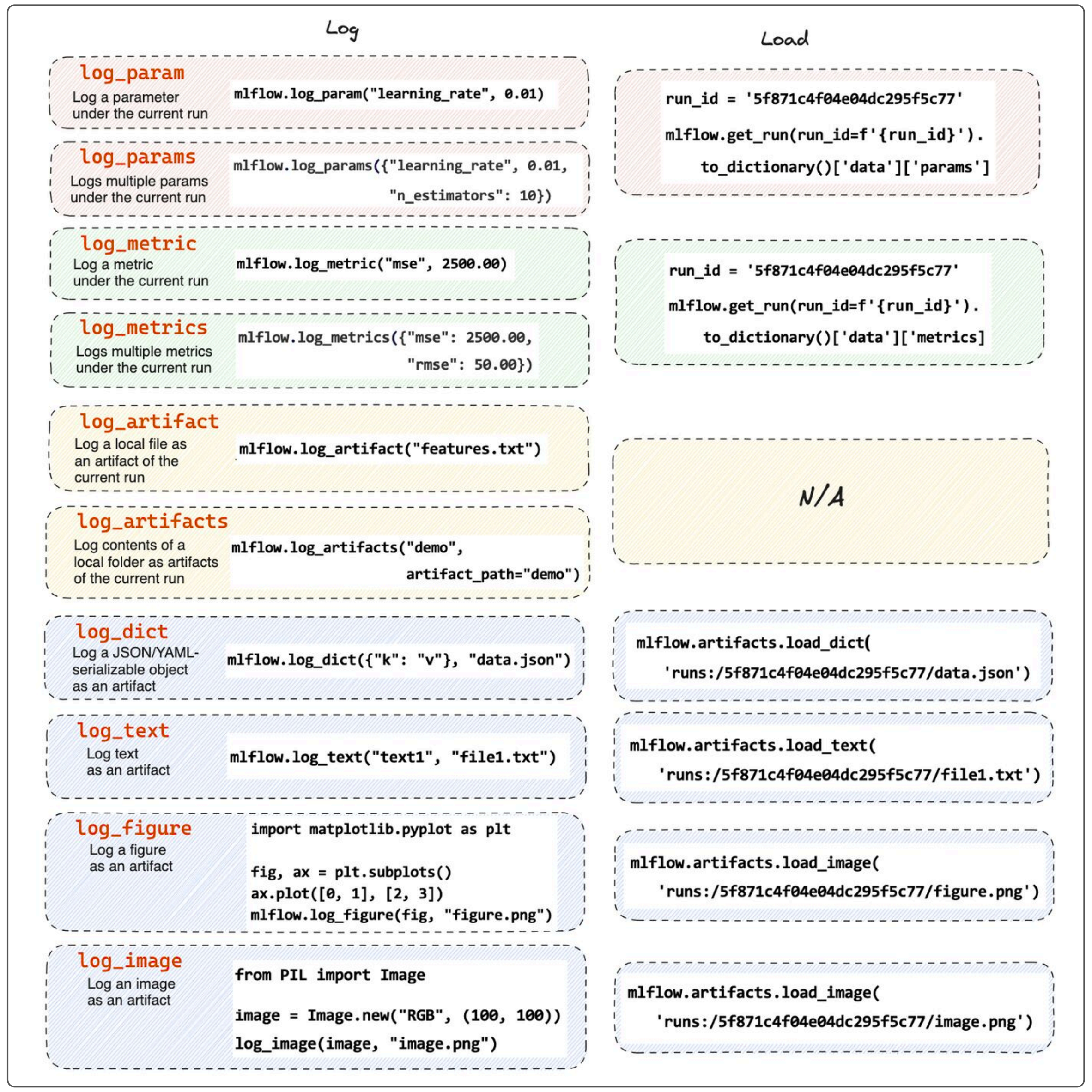
❒ **AUC** – The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes

Different versions of models, benchmarks and training/validation and testing experiments can be tracked, logged and set for production using **MLOps** in **MLflow** framework for example

MLFlow main use cases



Log

log_param
Log a parameter under the current run

```
mlflow.log_param("learning_rate", 0.01)
```

log_params
Logs multiple params under the current run

```
mlflow.log_params({"learning_rate", 0.01,
                   "n_estimators": 10})
```

log_metric
Log a metric under the current run

```
mlflow.log_metric("mse", 2500.00)
```

log_metrics
Logs multiple metrics under the current run

```
mlflow.log_metrics({"mse": 2500.00,
                    "rmse": 50.00})
```

log_artifact
Log a local file as an artifact of the current run

```
mlflow.log_artifact("features.txt")
```

log_artifacts
Log contents of a local folder as artifacts of the current run

```
mlflow.log_artifacts("demo",
                     artifact_path="demo")
```

log_dict
Log a JSON/YAML-serializable object as an artifact

```
mlflow.log_dict({"k": "v"}, "data.json")
```

log_text
Log text as an artifact

```
mlflow.log_text("text1", "file1.txt")
```

log_figure
Log a figure as an artifact

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([0, 1], [2, 3])
mlflow.log_figure(fig, "figure.png")
```

log_image
Log an image as an artifact

```
from PIL import Image

image = Image.new("RGB", (100, 100))
log_image(image, "image.png")
```

Load

```
run_id = '5f871c4f04e04dc295f5c77'

mlflow.get_run(run_id=f'{run_id}').
   to_dictionary()['data']['params']
```

```
run_id = '5f871c4f04e04dc295f5c77'

mlflow.get_run(run_id=f'{run_id}').
   to_dictionary()['data']['metrics]
```

N/A

```
mlflow.artifacts.load_dict(
   'runs:/5f871c4f04e04dc295f5c77/data.json')
```

```
mlflow.artifacts.load_text(
   'runs:/5f871c4f04e04dc295f5c77/file1.txt')
```

```
mlflow.artifacts.load_image(
   'runs:/5f871c4f04e04dc295f5c77/figure.png')
```

```
mlflow.artifacts.load_image(
   'runs:/5f871c4f04e04dc295f5c77/image.png')
```