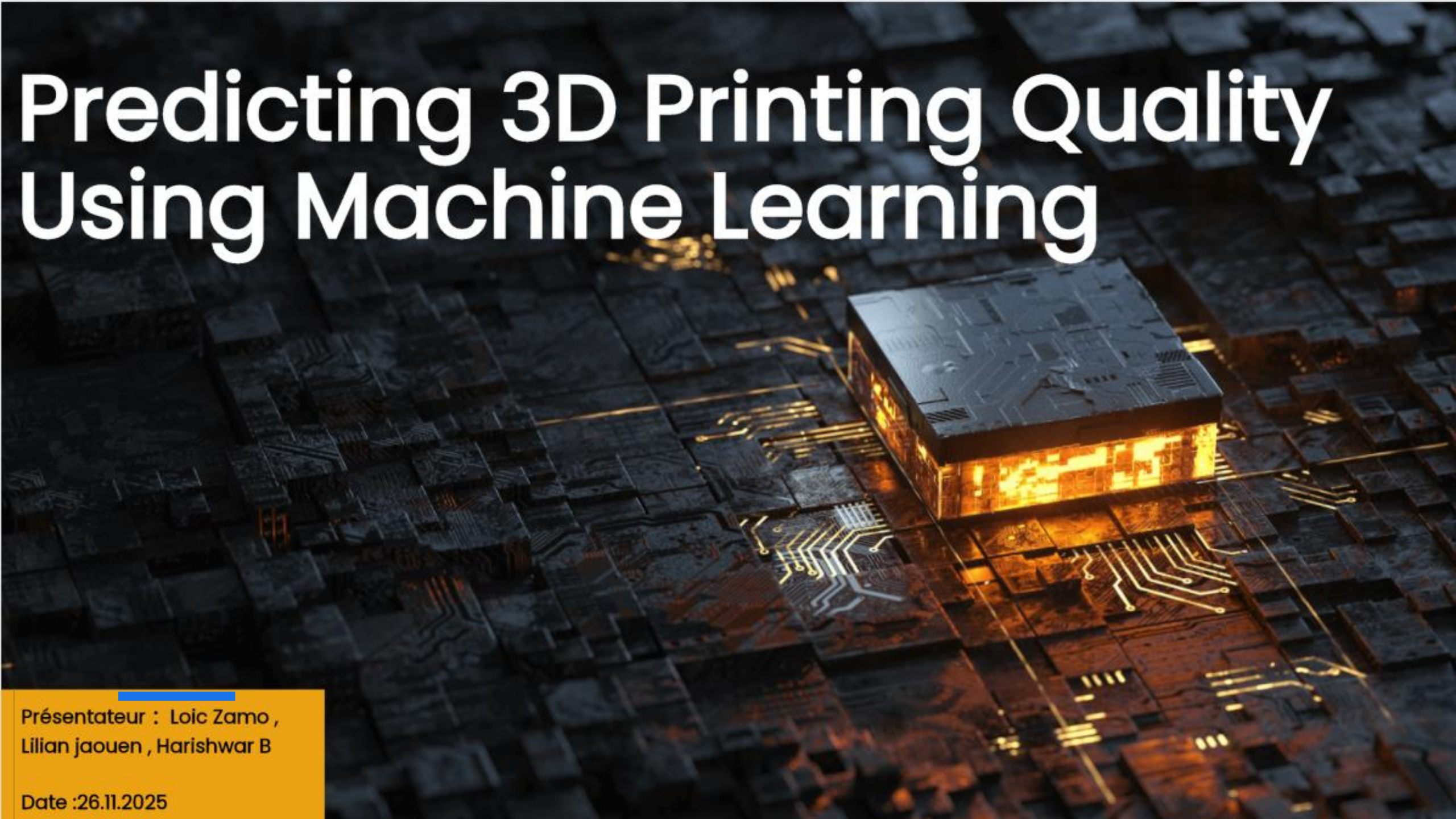
Predicting 3D Printing Quality Using Machine Learning



Présentateur : Loic Zamo , Lilian jaouen , Harishwar B

Date :26.11.2025

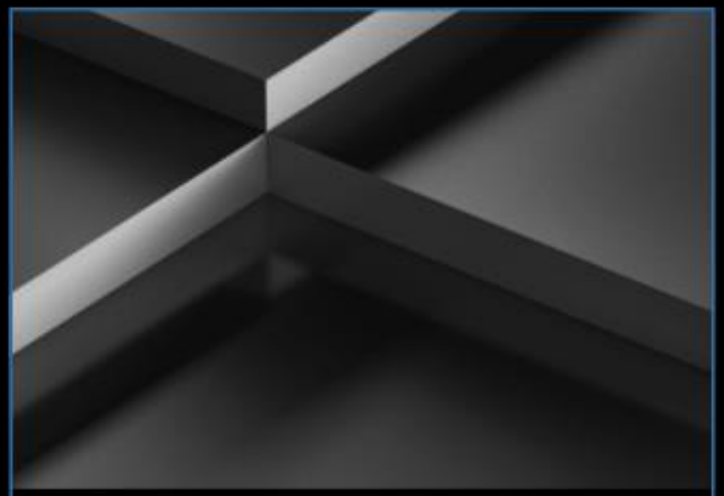
Sommaire


01
Business Problem

01 Business Problem

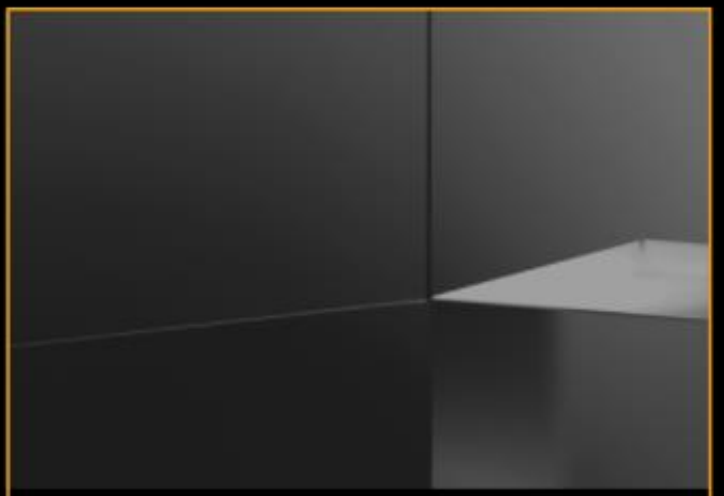
Industrial Context

3D printing of mechanical components requires meeting specific tensile-strength criteria. Currently, validating this strength relies on destructive testing. The part must be physically broken to assess its mechanical performance.



Pain Points

- High Costs: Significant material (ABS/PLA) waste and unnecessary energy consumption.
- Time Loss: Long production and testing cycles.
- Inefficiency: Mechanical quality cannot be evaluated before the print is completed.



Need

Shift from **post-process, destructive quality control to pre-print, virtual strength prediction.**

Introduction to Material Strength

Definition of Material Strength

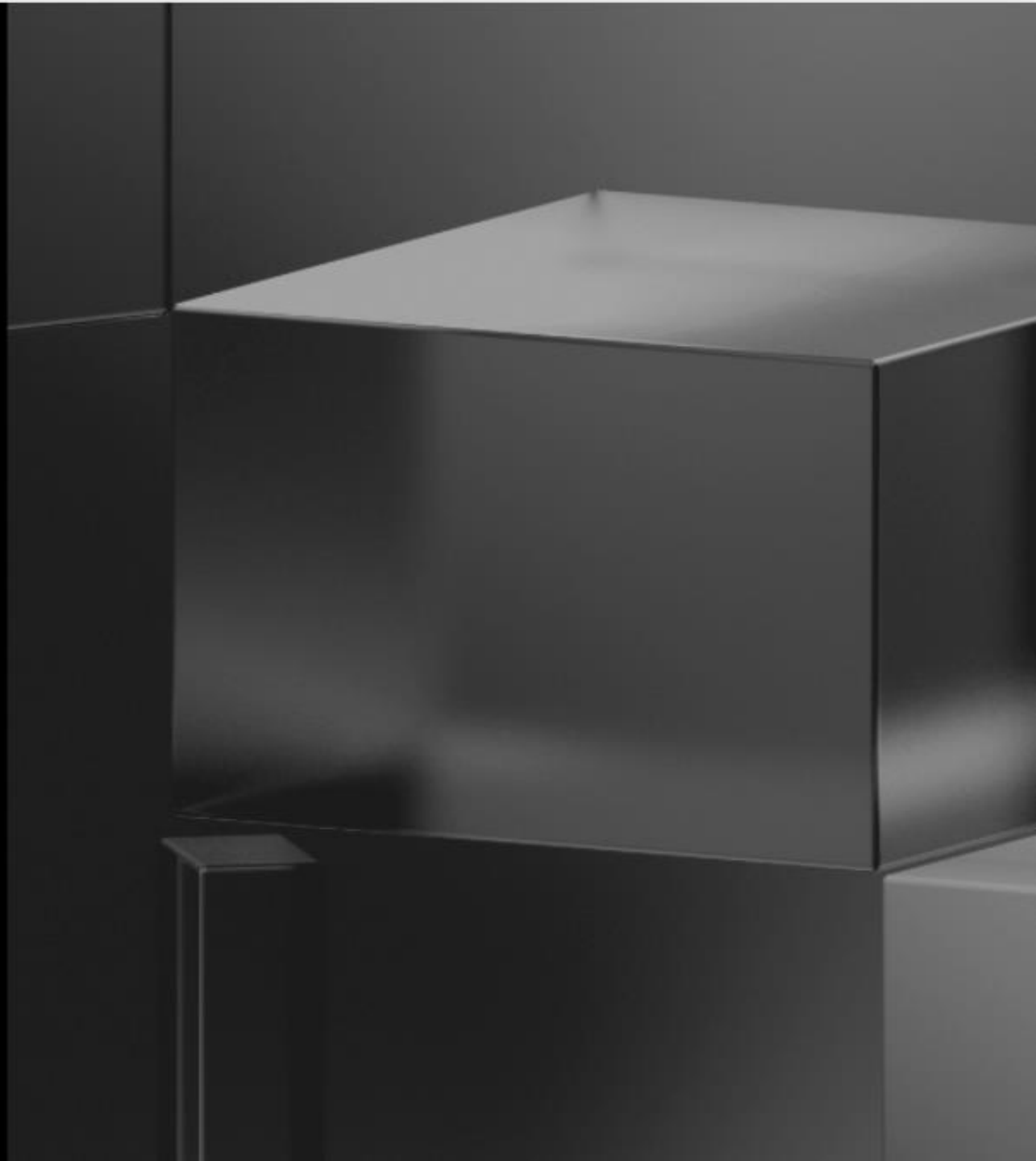
Material strength, specifically tensile strength, is the ability of a material to withstand a force that stretches it until failure, serving as a key indicator of robustness and durability.

Importance in Materials Engineering

Tensile strength is essential in materials engineering to ensure structural safety, long-term durability, and performance under mechanical loads.

Standard Measurements and Testing Methods

Standard tensile tests assess material strength by applying a controlled load, measuring deformation, and observing failure behavior to validate material quality and compliance with safety requirements.

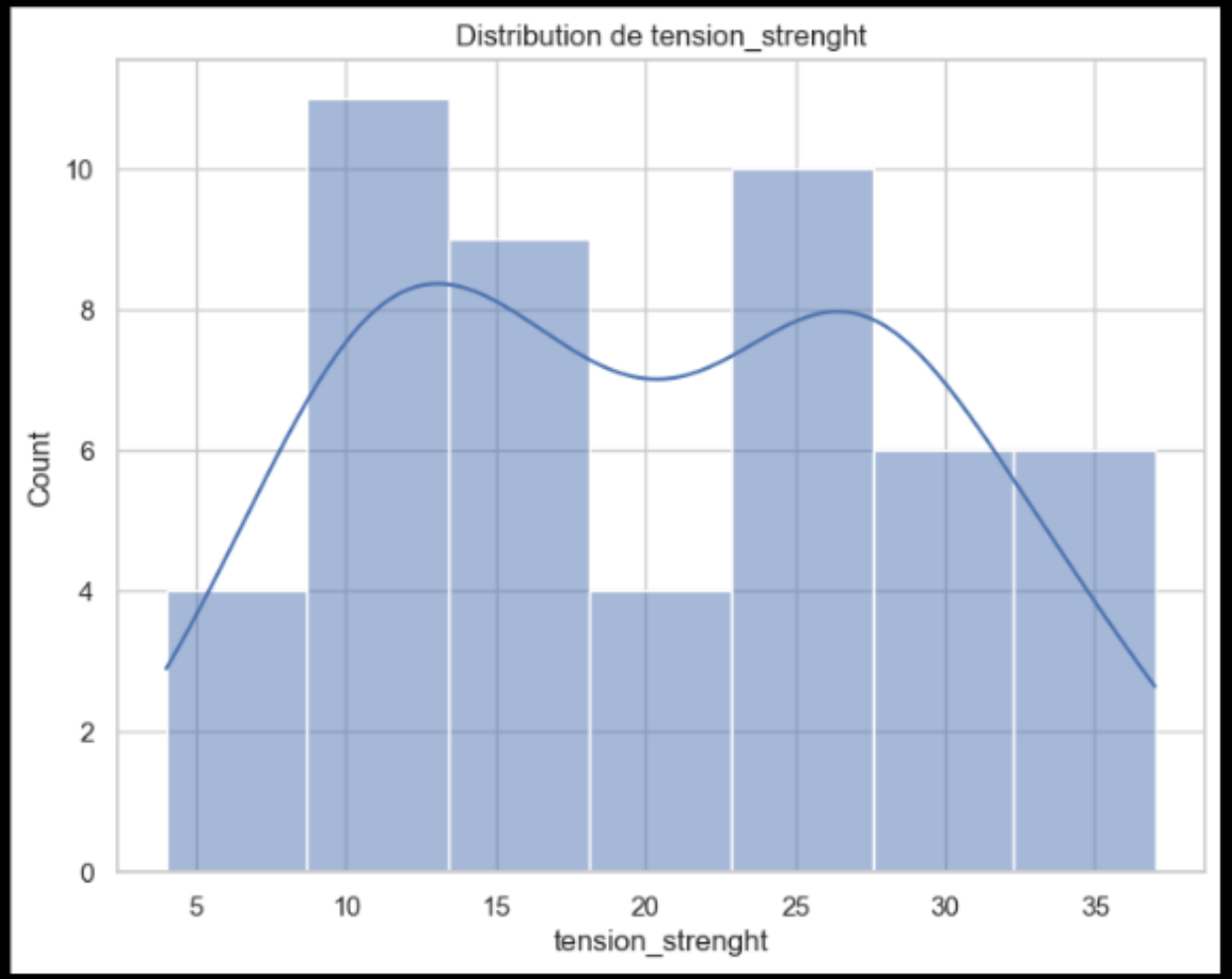



02
Methodological Approach

2.1 Analyse Exploratoire (EDA)

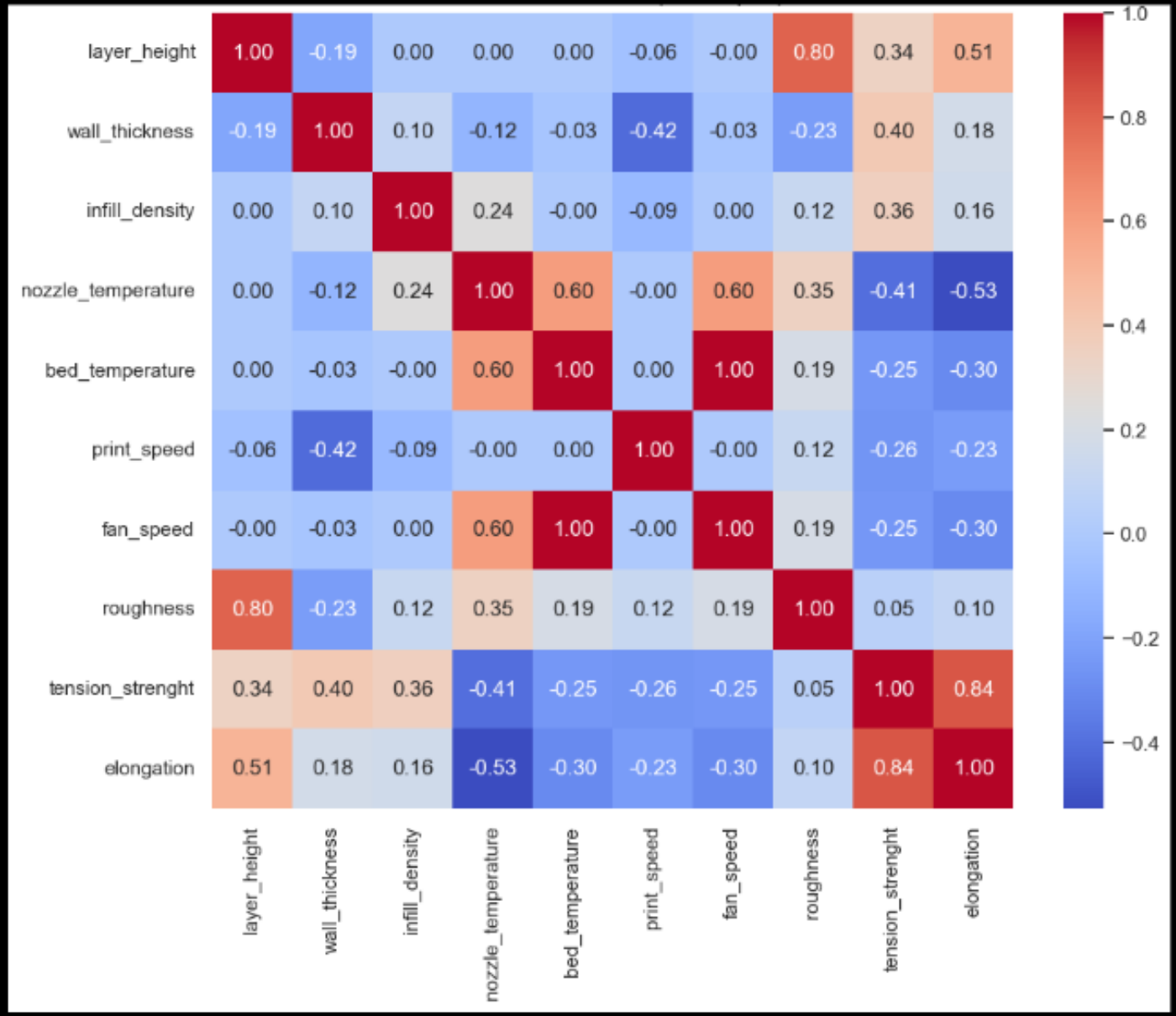
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
layer_height	50.0	NaN	NaN	NaN	0.106	0.064397	0.02	0.06	0.1	0.15	0.2
wall_thickness	50.0	NaN	NaN	NaN	5.22	2.922747	1.0	3.0	5.0	7.0	10.0
infill_density	50.0	NaN	NaN	NaN	53.4	25.36348	10.0	40.0	50.0	80.0	90.0
infill_pattern	50	2	grid	25	NaN	NaN	NaN	NaN	NaN	NaN	NaN
nozzle_temperature	50.0	NaN	NaN	NaN	221.5	14.820353	200.0	210.0	220.0	230.0	250.0
bed_temperature	50.0	NaN	NaN	NaN	70.0	7.142857	60.0	65.0	70.0	75.0	80.0
print_speed	50.0	NaN	NaN	NaN	64.0	29.6923	40.0	40.0	60.0	60.0	120.0
material	50	2	abs	25	NaN	NaN	NaN	NaN	NaN	NaN	NaN
fan_speed	50.0	NaN	NaN	NaN	50.0	35.714286	0.0	25.0	50.0	75.0	100.0
roughness	50.0	NaN	NaN	NaN	170.58	99.034129	21.0	92.0	165.5	239.25	368.0
tension_strenght	50.0	NaN	NaN	NaN	20.08	8.925634	4.0	12.0	19.0	27.0	37.0
elongation	50.0	NaN	NaN	NaN	1.672	0.788188	0.4	1.1	1.55	2.175	3.3

```
Colonnes catégorielles : ['infill_pattern', 'material']

Valeurs de 'infill_pattern':
infill_pattern
grid         25
honeycomb    25
Name: count, dtype: int64

Valeurs de 'material':
material
abs    25
pla    25
Name: count, dtype: int64
```



Correlation analysis (heat map) and 3D visualization to identify influential factors (e.g., highly correlated elongation).

Correlation matrix (numerical)



```python
TARGET = 'tension_strenght'
assert TARGET in df.columns, f"Target {TARGET} non trouvée"

# Features / target
X = df.drop(columns=[TARGET])
y = df[TARGET].astype(float)

# Identifier types
numeric_features = X.select_dtypes(include=['int64','float64']).columns.tolist()
categorical_features = X.select_dtypes(include=['object','category']).columns.tolist()

print("Features numériques :", numeric_features)
print("Features catégorielles :", categorical_features)
```

```
Features numériques : ['layer_height', 'wall_thickness', 'infill_density', 'nozzle_temperature', 'bed_temperature', 'print_speed', 'fan_speed', 'roughnes
s', 'elongation']
Features catégorielles : ['infill_pattern', 'material']
```

```python
# Pipelines de preprocessing
numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False)
    )])

preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

2.2 Preprocessing (Prétraitement)

Encoding of categorical variables (Material, Infill Pattern).

Normalization of numerical data (Scaling).

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

# Pipelines modèles
lr_pipeline = Pipeline([('preprocessor', preprocessor), ('regressor', LinearRegression())])
rf_pipeline = Pipeline([('preprocessor', preprocessor), ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))])

# Cross-validation (R2)
cv = KFold(n_splits=5, shuffle=True, random_state=42)
lr_cv_scores = cross_val_score(lr_pipeline, X_train, y_train, cv=cv, scoring='r2')
rf_cv_scores = cross_val_score(rf_pipeline, X_train, y_train, cv=cv, scoring='r2')

print(f"LR CV mean R2: {lr_cv_scores.mean():.4f} (std {lr_cv_scores.std():.4f})")
print(f"RF CV mean R2: {rf_cv_scores.mean():.4f} (std {rf_cv_scores.std():.4f})")

# Entrainement sur tout l'entrainement
lr_pipeline.fit(X_train, y_train)
rf_pipeline.fit(X_train, y_train)

# Prédictions test
y_pred_lr = lr_pipeline.predict(X_test)
y_pred_rf = rf_pipeline.predict(X_test)
```

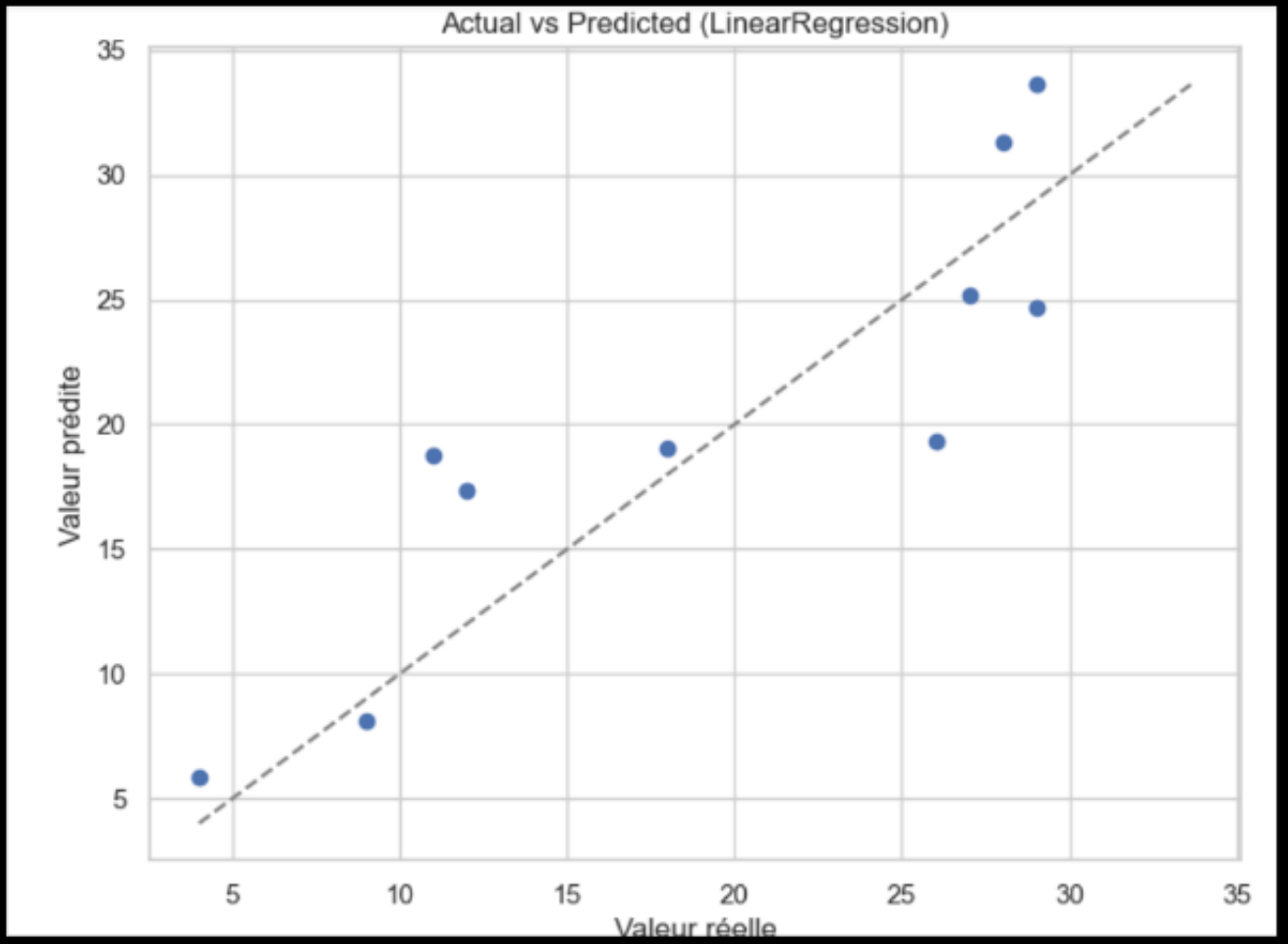
```python
# Fonctions metriques
def regression_report(y_true, y_pred):
    return {
        'RMSE': mean_squared_error(y_true, y_pred, squared=False),
        'MAE': mean_absolute_error(y_true, y_pred),
        'R2': r2_score(y_true, y_pred)
    }

results = {
    'LinearRegression': regression_report(y_test, y_pred_lr),
    'RandomForest': regression_report(y_test, y_pred_rf)
}

pprint(results)
```

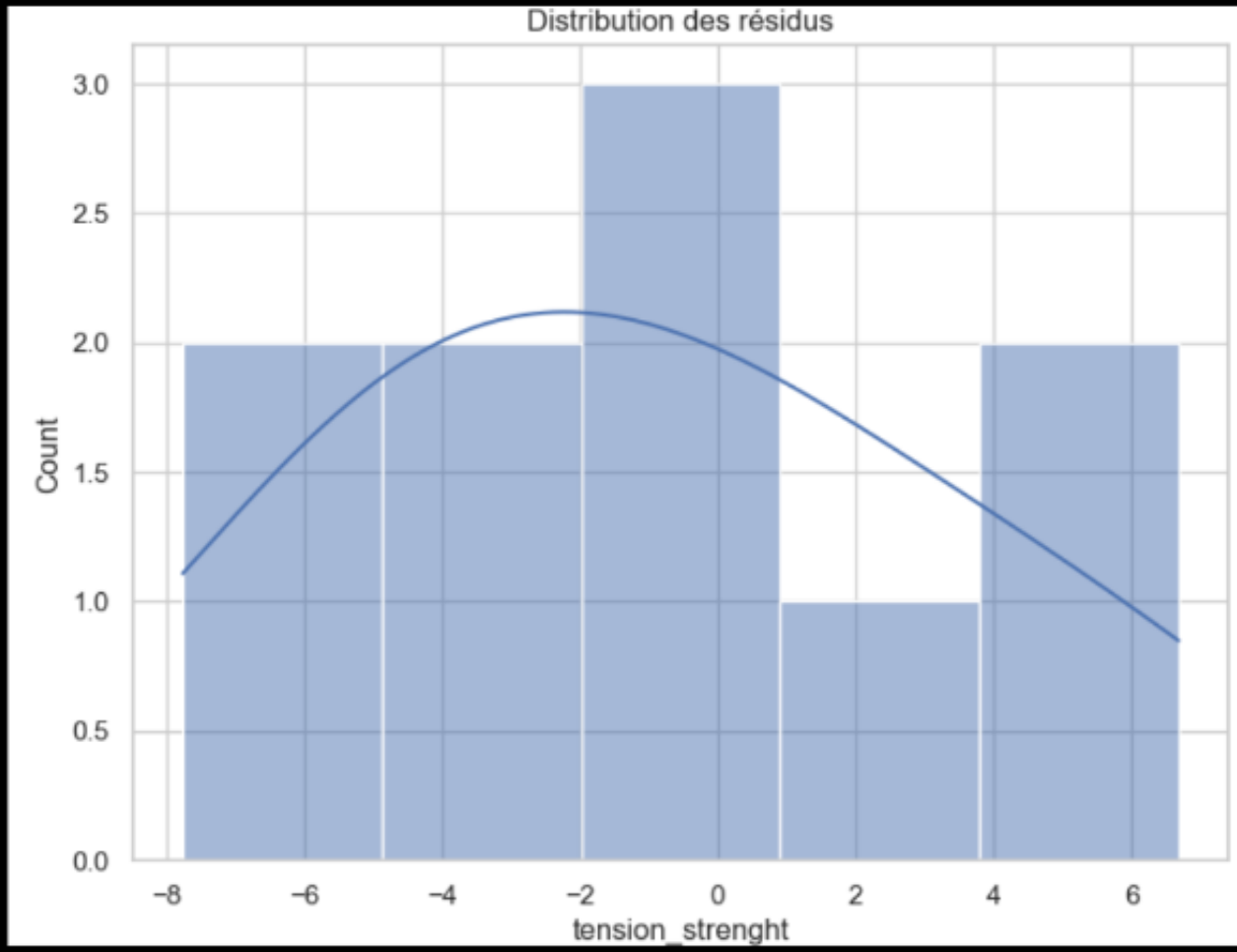
```
LR CV mean R2: 0.5127 (std 0.1132)
RF CV mean R2: 0.6781 (std 0.1403)
{'LinearRegression': {'MAE': 3.7550961410123804,
                      'R2': 0.7682617565282079,
                      'RMSE': 4.39123436396736},
 'RandomForest': {'MAE': 3.9240000000000004,
                  'R2': 0.721156111044356,
                  'RMSE': 4.816907721765074}}
```

2.3 Data Separation

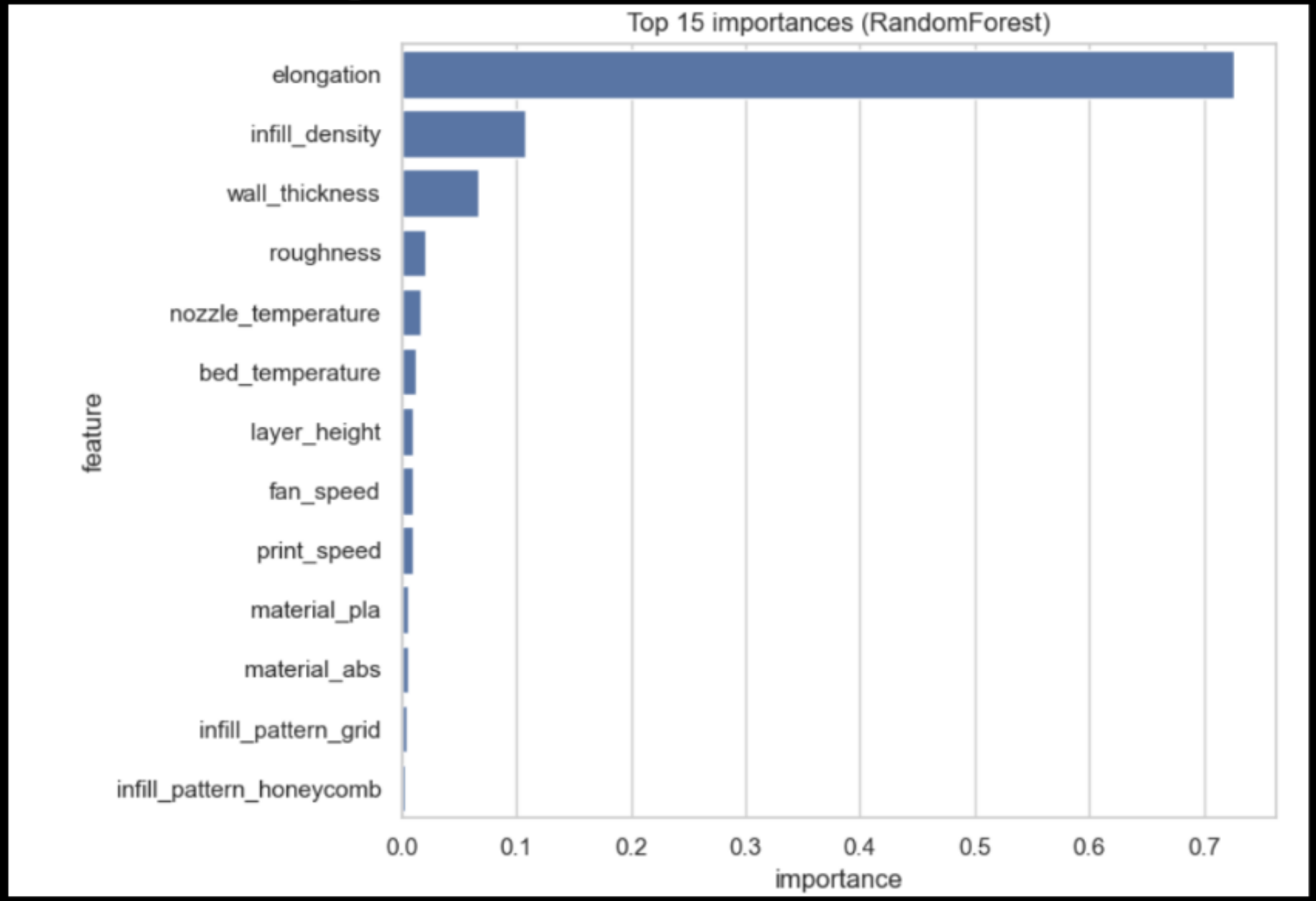
Split Train/Test (80% / 20%) with cross-validation (K-Fold) to avoid overfitting on a small dataset.

2.4 Iterative Modeling





2.4 Iterative Modeling




03
Technology to Apply

Technology to Apply

Key librairies

Data Manipulation:

Pandas, NumPy, DataFrame handling, numerical computation, and matrix operations.

Visualization:

Seaborn, Matplotlib, Correlation heatmaps, 3D scatter plots, and exploratory data visualization.

Machine Learning:

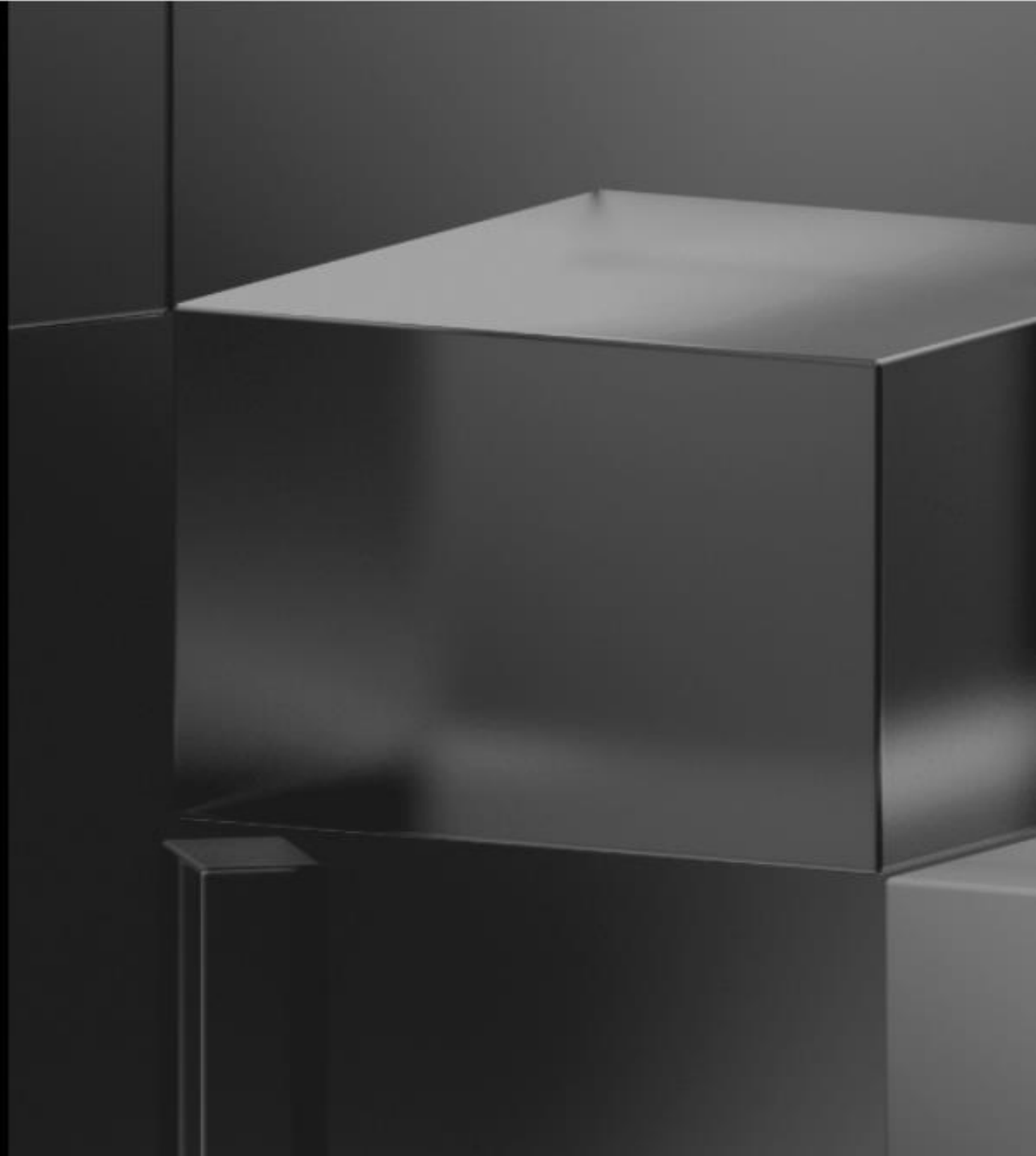
Scikit-learn

Algorithms:

LinearRegression, RandomForestRegressor

Preprocessing:

StandardScaler, OneHotEncoder, Pipeline




04
Data Quality

Strength

- Clean, well-structured dataset with no missing values.
- Strong correlation between input features and the target variable, enabling effective model learning.

Weaknesses and Risks

- **Low data volume**
  Only 50 samples, leading to a high risk of bias and overfitting.
- **Data Leakage Risk**
  The elongation and roughness columns are measurement outputs rather than printer parameters.
- **Corrective Action**
  Develop a model variant that excludes these columns to ensure the predictor remains truly actionable before fabrication.

05
Implementation - First Functional Version

Implementation - First Functional Version

01

Functional Deliverable

A fully operational Scikit-learn pipeline integrating both preprocessing steps and the predictive model.

02

Solution Architecture

Input: Machine parameters (layer height, wall thickness, nozzle temperature, etc.).

Processing: A ColumnTransformer automatically normalizes numerical features and encodes categorical variables.

Inference: The leakage-free Random Forest model predicts tensile strength.

03

Output: Numerical tensile-strength value (MPa).


06
Conclusion

Conclusion

Feasibility

The project demonstrates that predicting print quality from machine parameters is achievable.

Performance

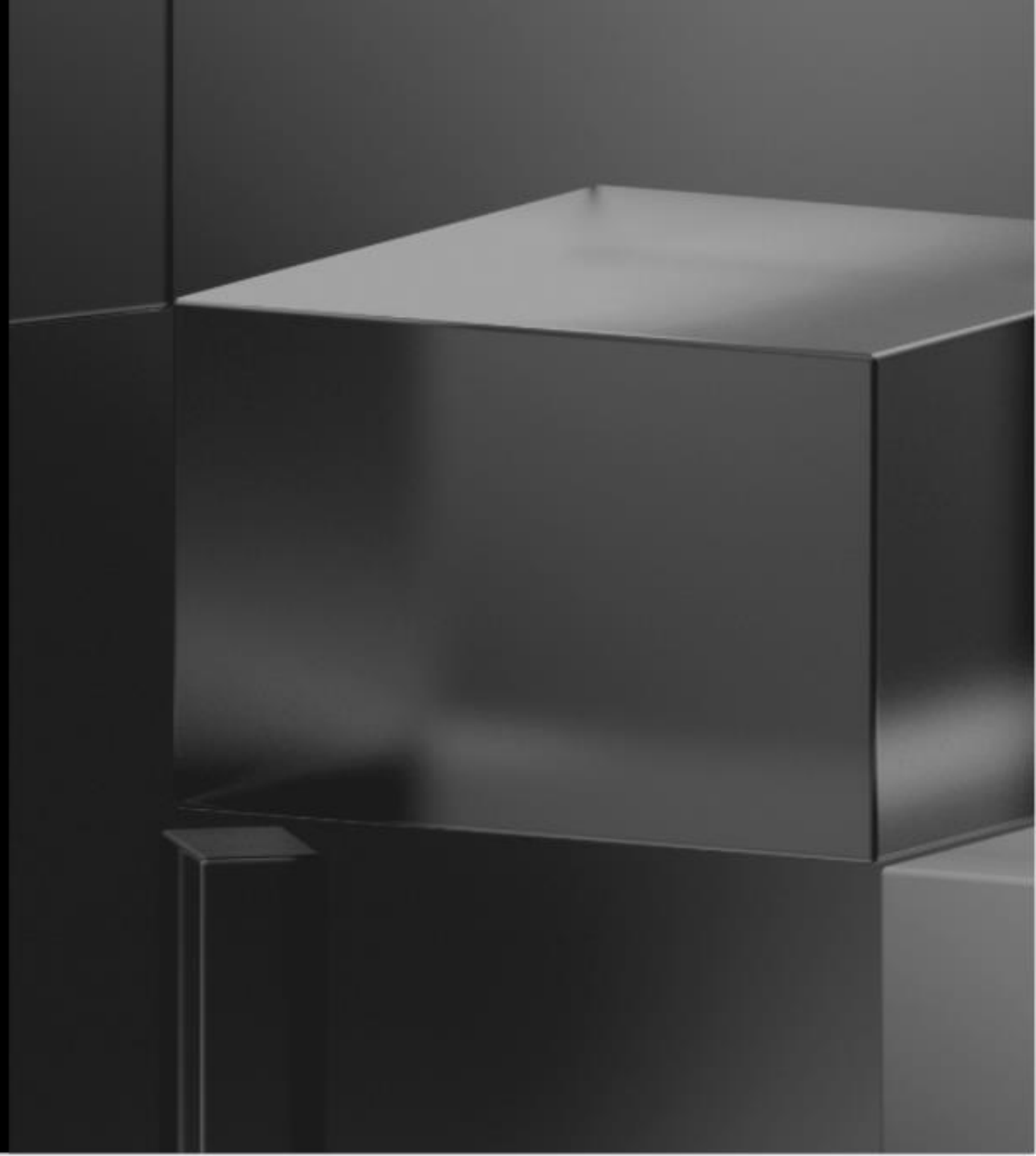
Models reach satisfactory R^2 scores even with 50 simple sample.

Next Steps

Brute-force pipeline optimization
Exhaustively search the parameter space of each model pipeline to identify the most optimal hyperparameters.

Explore additional models
Evaluate alternative algorithms that may be better suited to the problem and potentially deliver improved predictive performance.

Predict additional targets
Extend the workflow to predict another feature, such as the material type, as a complementary classification task.



Thank you for listening,

Question ?