Data Science Project Report: Quality Prediction in 3D Printing

1. Executive Summary

This project seeks to apply Machine Learning techniques to optimize the 3D printing process. The primary objective is to predict the Tensile Strength of a component prior to fabrication, thereby circumventing the costs and delays associated with destructive testing.

The analysis revealed that, while the data enables highly accurate predictions, particular attention had to be paid to Data Leakage stemming from post-production measurements (specifically elongation). A corrective solution was subsequently implemented to ensure the model's industrial viability

2.Business Understanding

Context

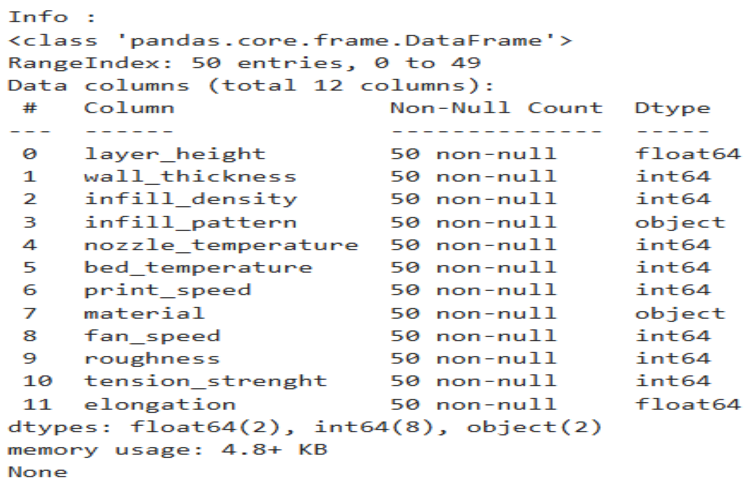
In the additive manufacturing industry, validating the structural integrity of a part typically requires printing it and then subjecting it to stress tests. Currently, this validation process proves particularly costly in terms of both energy and materials (such as ABS and PLA), which directly results in slower iteration cycles.

To mitigate these constraints, the selected solution involves leveraging Artificial Intelligence to estimate the mechanical strength of parts based solely on printer configuration parameters. The primary objective of this approach is to implement a regression model capable of accurately predicting the continuous value of tensile strength, expressed in MPa.

3. Exploratory Data Analysis (EDA)

The dataset underpinning this study is notable for its cleanliness: it contains 50 complete entries with no missing values, thereby bypassing the need for complex data imputation steps.

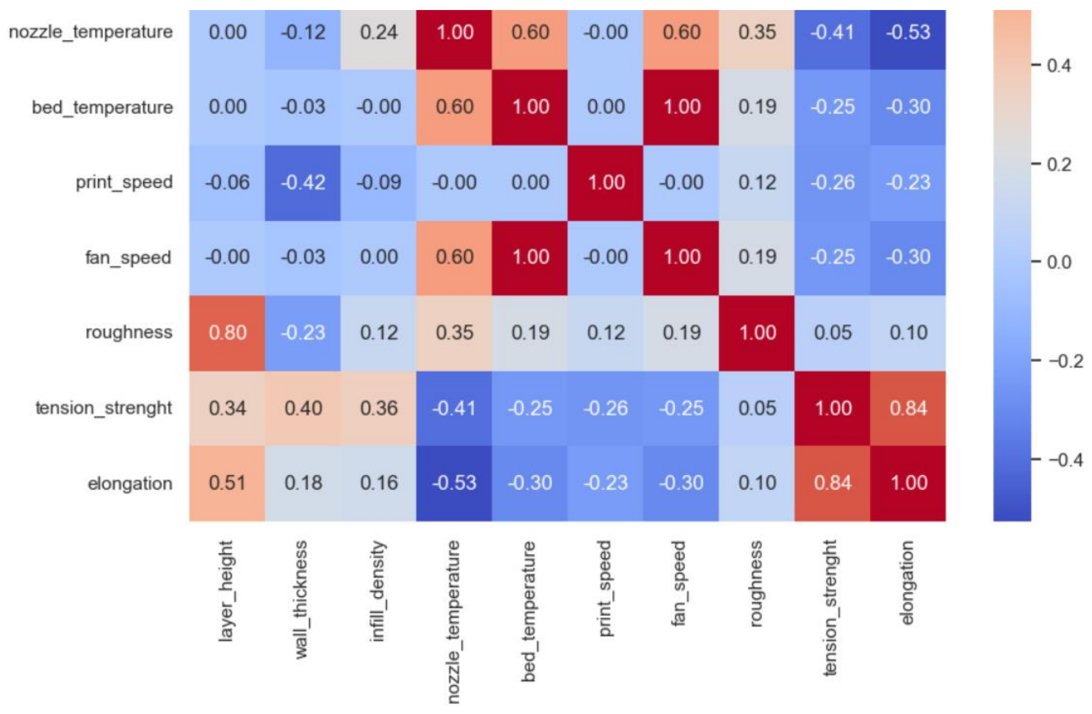
The structure revolves around two distinct categories of variables. On the one hand, we have the upstream configurable machine parameters, such as layer height (layer_height), wall thickness (wall_thickness), and infill density (infill_density), as well as printing temperatures and speeds. On the other hand, the dataset includes the output physical measurements observed post-fabrication, namely elongation (elongation), roughness (roughness), and our primary target variable: tensile strength (tension_strenght).



```
Info :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   layer_height        50 non-null     float64
 1   wall_thickness      50 non-null     int64
 2   infill_density      50 non-null     int64
 3   infill_pattern      50 non-null     object
 4   nozzle_temperature  50 non-null     int64
 5   bed_temperature     50 non-null     int64
 6   print_speed         50 non-null     int64
 7   material            50 non-null     object
 8   fan_speed           50 non-null     int64
 9   roughness           50 non-null     int64
 10  tension_strenght    50 non-null     int64
 11  elongation          50 non-null     float64
dtypes: float64(2), int64(8), object(2)
memory usage: 4.8+ KB
None
```

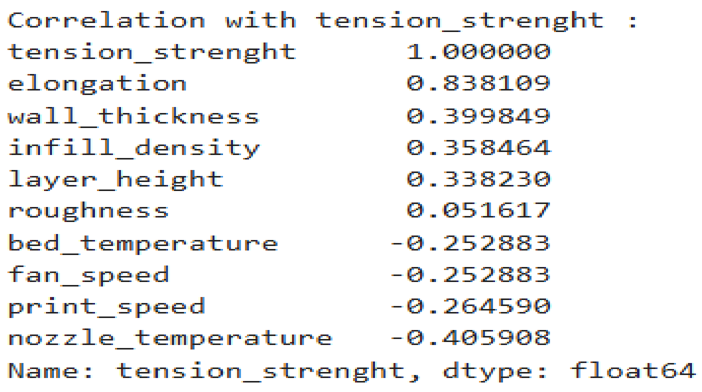
Key Insights Derived from the Analysis

The exploratory analysis rapidly brought to light intriguing dynamics. Primarily, the correlation matrix revealed an exceptionally strong statistical link (0.83) between elongation at break and mechanical strength. This observation is pivotal, as it underscores the direct physical relationship between a material's ductility and its tensile resistance.



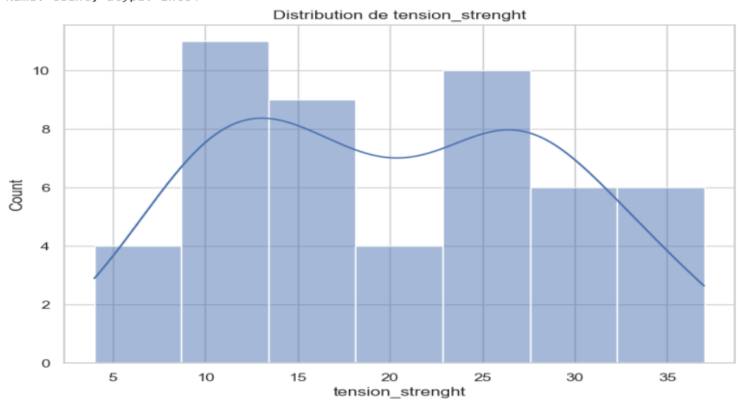
Furthermore, the use of 3D visualizations has confirmed the decisive impact of internal geometry. It is clearly apparent that infill density (infill_density), combined with wall thickness (wall_thickness), acts as the two primary physical levers for ensuring a part's structural integrity. (a)

Finally, an analysis of the target variable's distribution exhibits a highly characteristic bimodal profile. This distribution points to the existence of two distinct quality clusters within production : fragile components on the one hand, and robust parts on the other. This dichotomy is likely attributable to the radical behavioral disparity between the two tested materials (ABS and PLA) or to significant variations in density. (b)



```
Correlation with tension_strenght :
tension_strenght        1.000000
elongation              0.838109
wall_thickness          0.399849
infill_density          0.358464
layer_height            0.338230
roughness               0.051617
bed_temperature        -0.252883
fan_speed              -0.252883
print_speed            -0.264590
nozzle_temperature     -0.405908
Name: tension_strenght, dtype: float64
```

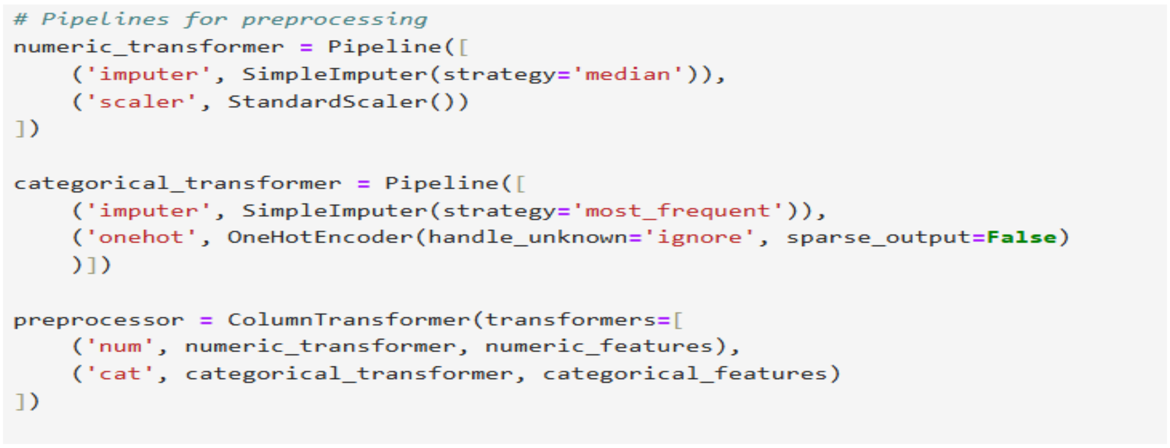
(a)



(b)

4. Data Preprocessing

To render the raw data suitable for our Machine Learning algorithms, a robust transformation pipeline was established. The initial critical step involved the handling of categorical variables. Textual features, such as material type (Material, distinguishing between ABS and PLA) and infill pattern (Infill_pattern, differentiating between Grid and Honeycomb), were converted into binary numerical values (0 and 1). This operation, executed via *Label Encoding* or *One-Hot Encoding* techniques, is imperative for translating qualitative concepts into a mathematical format intelligible to the model.

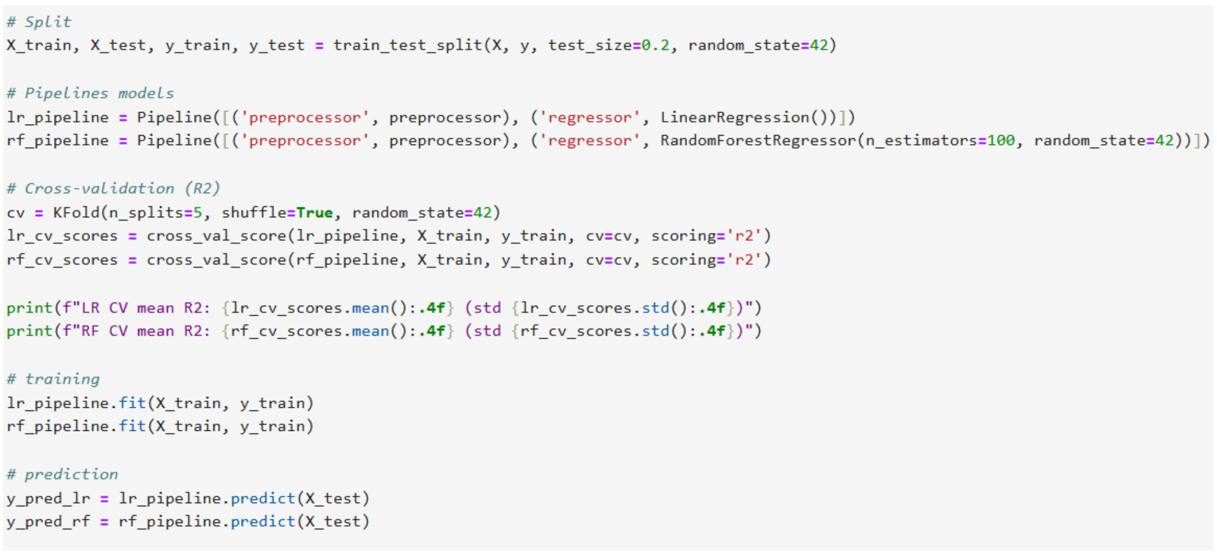


```python
# Pipelines for preprocessing
numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

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)
])
```

Subsequently, particular attention was devoted to harmonizing the data through feature scaling. Printing parameters naturally exhibit vastly disparate orders of magnitude, juxtaposing, for instance, a nozzle temperature of 200°C against a minute layer height of 0.1 mm. Without intervention, the algorithm might skew its calculations in favor of larger numerical values.

To rectify this, we applied a StandardScaler (centering data around a mean of 0 with a standard deviation of 1) or Min-Max normalization, thereby ensuring that each parameter contributes equitably to the final prediction.

Finally, to validate the robustness of our approach, the dataset was partitioned according to a standard procedure. We implemented a conventional split: 80% of the data was allocated to model training, while the remaining 20% was rigorously isolated to form the final test set.

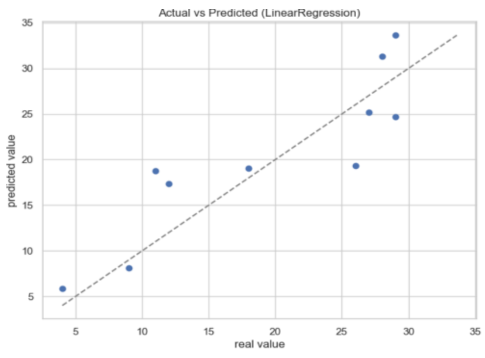


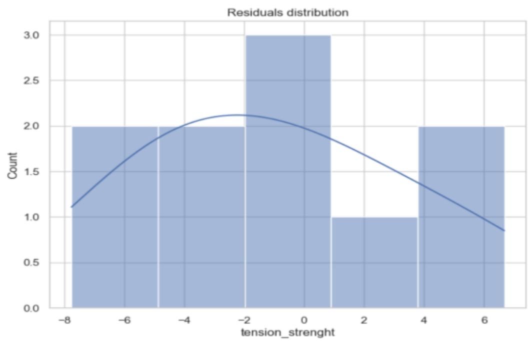
```python
# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Pipelines models
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})")

# training
lr_pipeline.fit(X_train, y_train)
rf_pipeline.fit(X_train, y_train)

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

5. Modeling and Results (Regression Phase)

To conduct the prediction phase, two primary algorithms were evaluated. We began with a Linear Regression, utilized as a baseline model. The performance of this initial model proved satisfactory: the plot comparing actual versus predicted values exhibits a consistent alignment along the ideal diagonal, demonstrating that the model effectively captures the general trend.

A thorough analysis of the residuals further validated this statistical approach. The error distribution follows a classic bell curve (Gaussian) and remains centered near zero, indicating the absence of significant structural bias. However, the precision leaves room for improvement regarding strict industrial application, as the error dispersion remains significant, oscillating between -8 and +6 MPa.



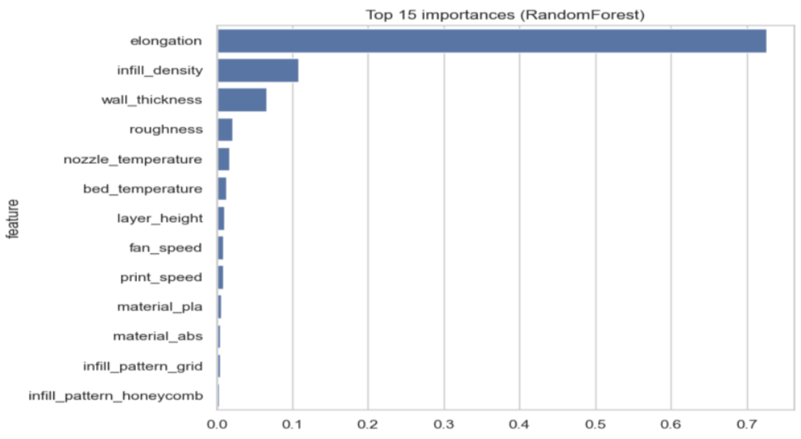


Subsequently, we implemented a Random Forest Regressor. The objective was to overcome the limitations of the linear model in order to capture the intricate and non-linear relationships inherent to material physics. This robust model demonstrated clear superiority, consistently exhibiting better performance across key evaluation metrics, characterized by a higher R^2 score and a reduced Root Mean Squared Error (RMSE).

```
LR CV mean R2: 0.5127 (std 0.1132)
RF CV mean R2: 0.6781 (std 0.1403)
{'LinearRegression': {'MAE': 3.755096141012378,
                      'R2': 0.7682617565282078,
                      'RMSE': 19.28293923928783},
 'RandomForest': {'MAE': 3.9240000000000004,
                  'R2': 0.7211561110443456,
                  'RMSE': 23.202599999999997}}
```

6. Critical Analysis: The Data Leakage Issue

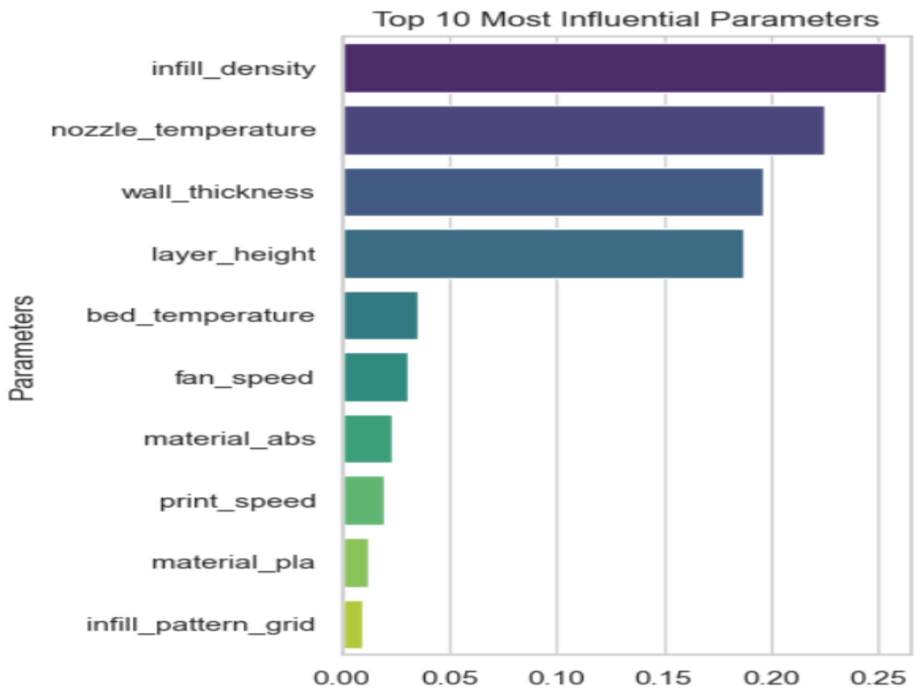
This phase represents the true turning point of the analysis, calling into question the viability of the initial results. An examination of the Feature Importance generated by the Random Forest model unveiled a major statistical anomaly: the elongation variable alone accounted for over 70% of the predictive power, effectively overshadowing the influence of all other parameters.



This excessive reliance underscored a critical issue of **Data Leakage**. Indeed, elongation is a physical property that can only be measured during destructive testing—that is, *after* the component has been manufactured. Consequently, a model requiring knowledge of elongation to predict strength renders itself useless in the pre-production phase, as this information is unavailable prior to initiating the print.

To rectify this methodological bias, we developed a model variant termed **"No Leak."** This correction involved scrupulously eliminating post-production measurement variables (elongation and roughness) from the input data.

The result is a sanitized model that now relies exclusively on genuine **machine parameters** configurable by the engineer: infill density (infill_density), wall thickness (wall_thickness), and temperatures. Although this approach is 'mathematically' less precise (with a naturally lower prediction score), it yields the only solution that is **valid and genuinely useful industrially** for quality anticipation.



8. Comparative Analysis of Advanced Models (XGBoost vs. SVM)

This phase aims to enhance the project's predictive capabilities by benchmarking two families of powerful, non-linear Machine Learning algorithms against one another. The objective is to determine which model yields superior performance in predicting mechanical strength (`tension_strenght`) by relying solely on machine configuration parameters (the "No Leak" approach), thereby rendering the model viable for industrial application.

Data preparation commenced with a rigorous feature selection process, strictly focusing the analysis on variables controllable prior to the printing process, such as infill density (`infill_density`), wall thickness (`wall_thickness`), as well as nozzle and bed temperatures. This methodology enabled the exclusion of variables acting as sources of data leakage, such as elongation, to ensure the model's validity. Subsequently, a standardization preprocessing step (`StandardScaler`) was systematically applied. This scale harmonization is critical, particularly for the SVM model, which relies on mathematical distance calculations.

; Without it, the disparity in orders of magnitude—ranging from a density below 100 to temperatures exceeding 200°C—would have inevitably biased the training process and resulted in model failure.

```
# 1. Load data
df = pd.read_csv('data.csv')

# 2.  features and target selection
features = ['infill_density', 'wall_thickness', 'nozzle_temperature', 'bed_temperature']
target = 'tension_strenght'

X = df[features]
y = df[target]

# 3. split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Preprocessing (Standardisation is very important for the SVM)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

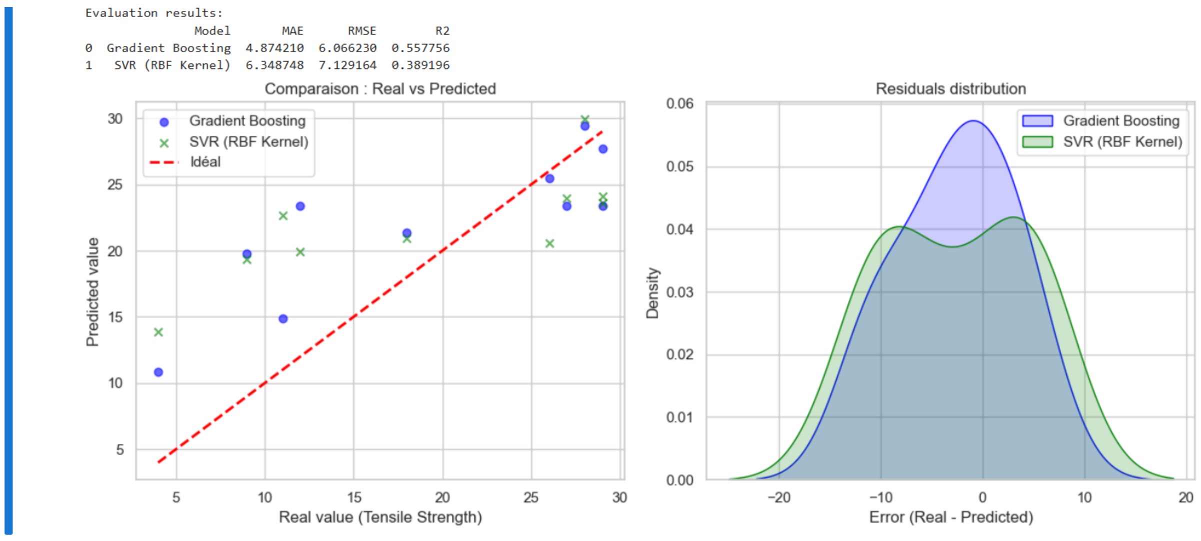
To test our hypotheses, two distinct algorithmic approaches were implemented. The first, **Model A**, relies on Gradient Boosting, an ensemble method based on decision trees. Our implementation leverages the XGBoost library, renowned for its speed and robustness on tabular data, while incorporating a software failsafe (a *try/except* block) that ensures an automatic fallback to the standard *GradientBoostingRegressor* in the event of technical unavailability.

In parallel, **Model B** explores a geometric approach via Support Vector Regression (SVR). To capture the intricate non-linear relationships between printing parameters, this model was configured with an RBF (Radial Basis Function) kernel. This kernel projects the data into a higher-dimensional space, utilizing hyperparameters set to C=10.0 for the penalty term and epsilon=0.1 for the margin of tolerance.

```
# import XGBoost. Otherwise we will use GradientBoostingRegressor as fallback proche.
try:
    from xgboost import XGBRegressor
    xgb_available = True
except ImportError:
    from sklearn.ensemble import GradientBoostingRegressor
    xgb_available = False
```

To validate model efficacy, the script generates two critical graphical interpretation tools. The first, a comparative **'Actual vs. Predicted' Scatter Plot**, overlays the performance of both algorithms by benchmarking their predictions against the ideal red diagonal. The anticipated observation is a tighter adherence of the blue data points (representing Boosting/XGBoost) to this diagonal relative to the SVR's green crosses, thereby signaling superior precision on this physical dataset.

Complementing this, a **residual distribution graph (KDE Plot)** facilitates the analysis of error morphology. The objective is to achieve a tall, narrow bell curve centered at zero; a blue curve that is sharper than the green one would definitively confirm that the Boosting model yields errors that are not only smaller in magnitude but also more consistent than those of its competitor.



```
Evaluation results:
               Model       MAE      RMSE        R2
0  Gradient Boosting  4.874210  6.066230  0.557756
1   SVR (RBF Kernel)  6.348748  7.129164  0.389196
```

In conclusion, this experiment demonstrates that for predicting 3D print strength based on machine parameters, Boosting methods (decision trees) offer a superior trade-off between precision and robustness compared to geometric methods such as SVM. Boosting proves more adept at handling the raw interactions between density and temperature.

Conclusion

The project successfully demonstrated the feasibility of quality prediction in 3D printing. The data leakage pitfall was identified and rectified, yielding a streamlined Random Forest model ready for software integration.**Recommandations (Next Steps)**

1. **Data Augmentation :** With a dataset comprising only 50 entries, the risk of overfitting is acute. It is imperative to collect a larger volume of printing data.
2. **Deployment :** Integrate the serialized model (`.joblib`) into an API (via Flask or FastAPI) to facilitate direct querying by 3D Slicer software.
3. **Optimization :** Leverage `RandomizedSearchCV` to fine-tune the Random Forest hyperparameters specifically on the "no-leak" version of the dataset.