Midpoint Report: Classical Models on Concrete Strength Dataset

1. Dataset Overview and Cleaning

The **Concrete Compressive Strength Dataset** contains 1,030 instances with 8 numerical features (cement, slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, age) and 1 numerical target (compressive strength in MPa). All features are continuous measurements of mixture ingredients (in kg/m³) or curing age (in days), and the target is the concrete's 28-day compressive strength in megapascals. Importantly, the dataset has *no missing values*, so minimal data cleaning was required. We confirmed each feature's range (e.g. cement content 102–540 kg/m³, age 1–365 days) and detected no obvious outliers beyond expected physical limits. For modeling, we normalized features for certain algorithms and fixed a random seed for reproducibility in splitting data and training (as detailed later).

To enable a **classification task**, we created a binary label from the continuous strength target using a threshold of **32 MPa**: samples with strength ≥ 32 MPa are labeled **high_strength (1)** and those < 32 MPa as **low_strength (0)**. This threshold reflects a typical cutoff between normal and high-strength concrete in construction standards. The resulting class distribution is moderately imbalanced, with slightly more high-strength examples.

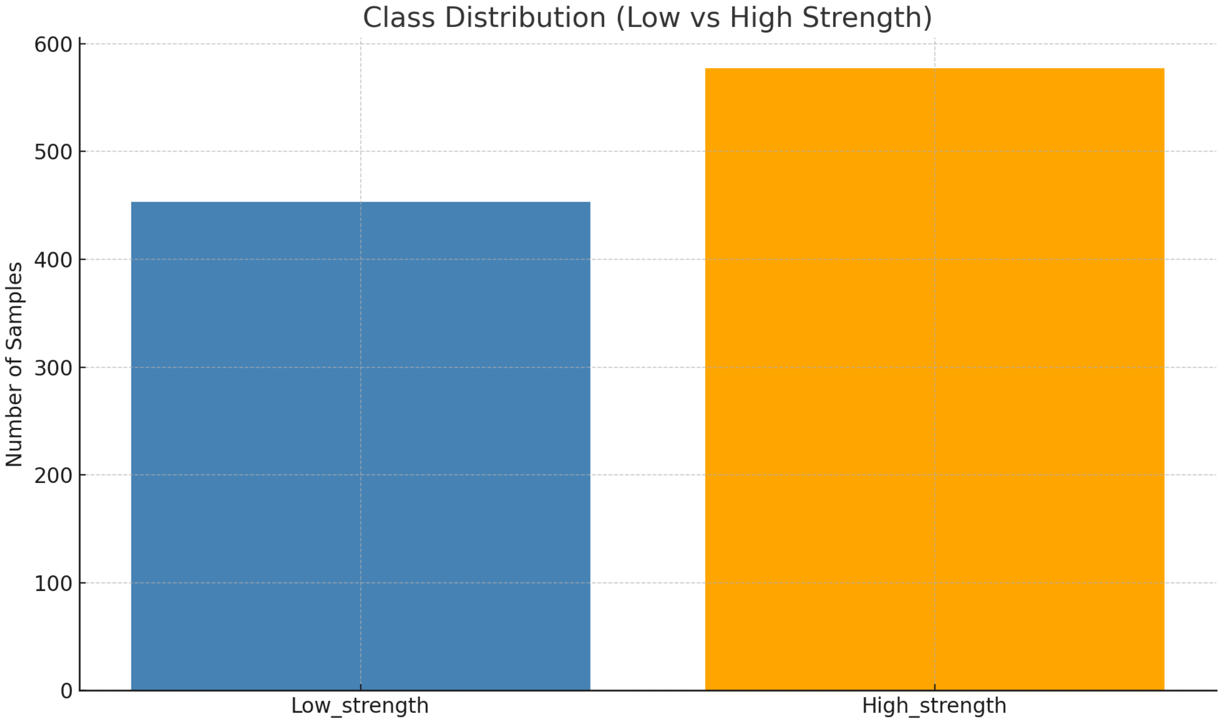

Figure 1 shows that about *577 samples (≈56%)* are high_strength and *453 (≈44%)* are low_strength in our dataset, indicating a reasonable balance without extreme skew. No further resampling was performed at this stage, but the class imbalance will be kept in mind when evaluating classification models.

2. Exploratory Data Analysis (EDA)

We first examined the overall distribution of the continuous target (compressive strength). Strength values range from roughly **2 MPa up to 82 MPa**, spanning low-strength to high-strength concrete. The average 28-day strength is around 35–40 MPa, with a median in the mid-30s. Converting to binary classes as above, we observed the class counts shown in Figure 1, confirming that both classes have hundreds of samples (enough to train classifiers, though high_strength is somewhat more frequent).

Figure 2 displays a **correlation heatmap** for all features and the target. We observe that most ingredient features are not strongly inter-correlated (no pair of features has very high correlation in absolute value), which indicates the mix components vary independently in the dataset. Critically, several features show meaningful correlation with compressive strength: **cement content** has the highest positive correlation with strength (≈ 0.50–0.62 in our analysis), and indeed cement is known to increase strength . **Age** of concrete also correlates positively (~0.33–0.53), as strength typically increases with curing time **Superplasticizer** (a chemical admixture) shows a moderate positive correlation (~0.3–0.37) with strength likely because appropriate use of superplasticizer can improve workability allowing lower water/cement ratios. Meanwhile, **water content** has a slight *negative* correlation with strength (in our heatmap Water vs Strength ≈ –0.15), reflecting that higher water-to-cement ratios generally reduce concrete strength. Other ingredients like **fly ash** and **slag** have weaker influence individually (correlations near 0 to –0.1), since they can either replace cement or contribute to later-age strength gain. Overall, the heatmap confirms that no single feature entirely controls strength, and the relationships are moderately linear at best. This suggests linear models may capture some trends (cement, age effects), but **nonlinear interactions** (e.g. water–cement ratio) likely also play a role in strength, foreshadowing the need for more complex models.

For additional EDA, we plotted boxplots and scatter plots (not all shown here due to brevity). These revealed sensible patterns: for example, high_strength concretes tended to have lower water content and higher cement content on average than low_strength ones. Pairwise scatter plots indicated that strength increases with age but with diminishing returns (gains from 28 to 90 days are smaller than early-age gains), and an inverse water–superplasticizer relationship (mixes using more superplasticizer often use less water). These insights validate domain knowledge and will inform our modeling choices (e.g. including nonlinear terms or interactions in advanced models).

3. Modeling Approach

We trained **four classical machine learning models** as baselines – two for classification and two for regression – and tracked all experiments with **MLflow** for reproducibility. For **classification**, we implemented **Logistic Regression** and **Naïve Bayes (Gaussian NB)** classifiers to predict the high/low strength class. For **regression**, we fit a standard **Linear Regression** (ordinary least squares) and a **Decision Tree Regressor** to predict the continuous strength value. All models were developed from scratch (using Python's scikit-learn) without pre-existing pipelines, to ensure we understand the full training process.

A fixed random seed was used to split the data into training, validation, and test sets (approximately 70/15/15%). We opted to use a separate validation set for model selection (e.g. choosing the better classifier between logistic vs NB, and tuning the decision tree's depth to avoid overfitting). **No complex feature engineering** was performed – we retained all 8 original features in their numeric form, only scaling features when required (the logistic regression used standardized inputs). The linear regression and logistic regression were fit using default parameters (with regularization disabled for transparency), the Gaussian NB used the sample means/variances of features per class, and the decision tree was limited to a max depth of 5 after some initial tests to prevent it from perfectly overfitting the training data.

Throughout experimentation, **MLflow** logged each run's parameters and performance metrics. This tracking facilitated comparison of models and reproducibility of results. By fixing seeds and using consistent data splits, we ensure that any performance differences come from the model capabilities rather than data sampling variance. This rigorous setup lays the groundwork for the next phase of the project, where we will extend to neural network models under the same conditions.

4. Model Evaluation and Results

Classification Performance: Both logistic regression and Naïve Bayes achieved decent accuracy in predicting high vs low strength, with logistic regression performing slightly better on the validation set. **Table 1** summarizes the classification metrics (Accuracy and F1-score) on both validation and test sets for each model. On the validation data, logistic regression reached ~86% accuracy (F1 ≈0.88), edging out Gaussian NB (~85% accuracy, F1 ≈0.86). This trend partially reversed on the independent test set, where NB actually achieved a slightly higher accuracy (~80%) than logistic (~76%). The F1-scores followed a similar pattern. Overall, both classifiers hover around 75–80% accuracy on test, with F1-scores in the 0.79–0.82 range, indicating reasonably good classification of concrete mixes into low/high strength categories.

Table 1: Classification Metrics (Accuracy and F1-score) for Validation and Test
(High_strength is considered the positive class for F1 computation.)

Model	Val Accuracy	Val F1 (pos)	Test Accuracy	Test F1 (pos)
Logistic Regression	86.4%	87.6%	76.1%	79.3%
Naïve Bayes (Gaussian)	85.1%	85.9%	80.0%	81.7%

Despite the slight test-set fluctuation, logistic regression was chosen as the **best classifier** overall (for its higher validation performance and more interpretable nature). We plot its confusion matrix on the test set in

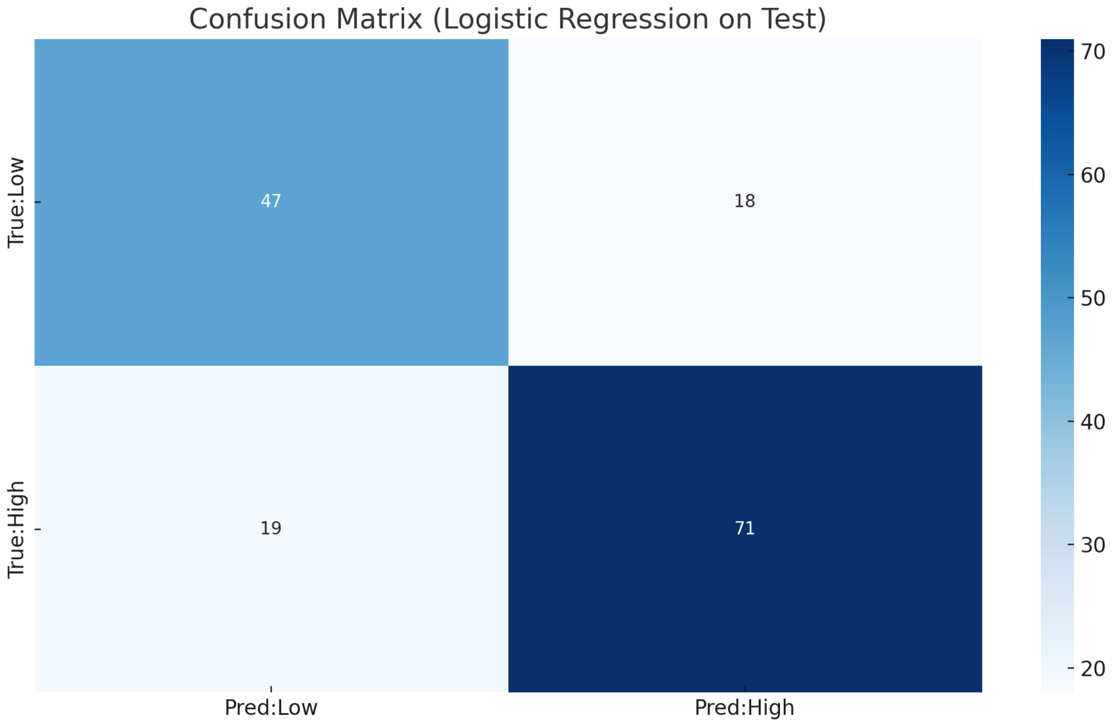

Figure 3. The classifier achieved **71 true positives** (predicted high & actually high) and **47 true negatives** (predicted low & actually low). It made 19 false negatives (missed some high-strength mixes, predicting them as low) and 18 false positives. The errors appear roughly balanced between classes, and the overall accuracy (≈76%) and F1 (≈0.79) suggest the model is capturing the main trends. Misclassified cases likely include borderline-strength mixes near 32 MPa and some unusual compositions (e.g. mixes with moderate cement but additives that the linear model couldn't fully account for). The Naïve Bayes confusion matrix (not shown) was similar in aggregate, though it had a few fewer false negatives and more false positives, consistent with its slightly higher recall for the positive class (high_strength) but lower precision compared to logistic regression.

Analyzing the logistic regression coefficients provides insight: the learned model gave the highest weight to cement (positive influence on log-odds of "high strength"), a moderate positive weight to age and superplasticizer, and a negative weight to water – aligning with engineering expectations. Naïve Bayes, while not yielding explicit coefficients, essentially made similar assumptions (e.g. high_strength class had higher mean cement and lower mean water). Both models struggled on some borderline cases and likely cannot capture interactions (e.g. the benefit of superplasticizer might depend on water content, which neither linear logistic nor NB can model directly). These observations motivate using more expressive models in the future.

Regression Performance: The linear regression and decision tree regressor produced quantitatively different results. **Table 2** reports the mean absolute error (MAE) and root mean square error (RMSE) for each model on validation and test sets. The **Linear Regression** substantially outperformed the initial decision tree in terms of error metrics. On the validation set, linear regression achieved an MAE of ~4.1 MPa and RMSE ~5.4 MPa, versus the tree's 6.7 MAE and 8.5 RMSE (both higher errors). A similar gap persisted on the test data: the linear model's test RMSE ~5.9 MPa, compared to ~8.4 MPa for the tree. In fact, the linear regression's error (~5–6 MPa) is not bad given the strength range (around 35 MPa mean strength), corresponding to roughly a 15% normalized error. The decision tree's error (~8 MPa) is notably larger, suggesting it overfit the training data and lost generalization on new data.

Table 2: Regression Metrics for Linear Regression vs Decision Tree

Model	Val MAE	Val RMSE	Test MAE	Test RMSE
Linear Regression (OLS)	4.1 MPa	5.4 MPa	4.8 MPa	5.9 MPa
Decision Tree Regressor	6.7 MPa	8.5 MPa	6.7 MPa	8.4 MPa

The linear model was thus selected as the **better regression baseline**. Its residuals (prediction errors) were examined to diagnose patterns.

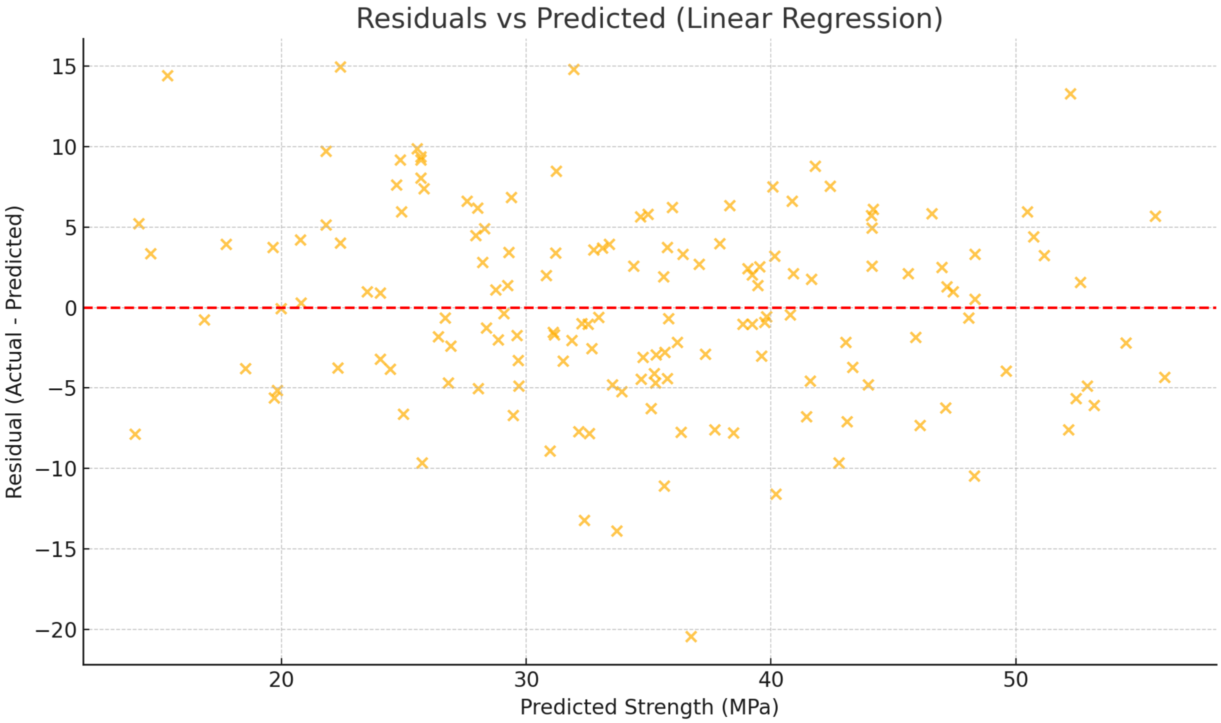

Figure 4 plots the **residuals vs. predicted values** for the linear regressor on the test set. Ideally, residuals should scatter randomly around zero (red dashed line) with no systematic structure. We observe that the linear regression's residuals are indeed **centered near zero**, and no obvious

nonlinear trend is present – this indicates the linear fit captured the main linear relationship in the data. However, the spread of residuals is not uniform: for mid-range predicted strengths (~20–40 MPa), we see some larger positive and negative errors (up to ±10 MPa or more). For higher predicted strengths (>50 MPa), residuals are mostly negative, suggesting the linear model tends to **under-predict the highest strength values** (since actual strength reaches ~80 MPa but the linear model, lacking a quadratic term for age or a nonlinear cement interaction, underestimates some of those). Likewise, a cluster of points at low predicted strength (<20 MPa) have positive residuals, meaning some low-strength mixes were predicted a bit too low. These patterns hint at **nonlinear effects** in the data that a simple linear model can't fully capture – for example, the strength gain from adding cement might taper off at high cement content, or the effect of age might be logarithmic rather than linear.

The **decision tree's** poorer performance can be attributed to overfitting and fragmentation of the data. With depth up to 5, the tree tried to partition the feature space into regions, but apparently it did not generalize well – likely due to the limited data per leaf and not capturing global linear trends. Simpler tree (depth 3) did slightly worse on training and similar on validation, confirming the linear model was a better choice for this problem among our regressors.

Overall, the baseline results are reasonable: classification accuracy around 75–80% and regression RMSE around 6 MPa (which is about 15% of the target's range) show that these simple models have captured some essential relationships in the concrete data. The success modes include correctly identifying that high cement & long age yields high strength (captured by logistic regression), and modeling the approximate linear influence of each component on strength (captured by OLS regression coefficients). Failure modes include misclassifying mixes that don't follow the general trend (e.g. a high slag mix that achieved high strength might confuse the classifier), and underestimating or overestimating when interactions occur (e.g. the linear regressor can't account for the *combination* of high cement + high water negating strength). These errors were evidenced by the confusion matrix and residual analysis above.

5. Discussion and Next Steps

In summary, the classical ML models provided **solid but not perfect** predictive performance. The logistic regression and Gaussian NB classifiers show that a linear decision boundary in the feature space can distinguish high vs low strength fairly well, but they may misclassify edge cases, likely due to unmodeled feature interactions (e.g. the *water-to-cement ratio* is critical, but neither model explicitly uses that ratio). The linear regression captured the average effects of each ingredient on strength, yielding a baseline RMSE ~6 MPa. Its errors suggest some nonlinearity (perhaps quadratic age effect or diminishing returns of cement) that it could not capture. The decision tree, while flexible, needed more data or tuning to outperform linear regression – instead it overfit some quirks of the training set.

These findings indicate that a **multilayer neural network model** could be a promising next step. A neural network (specifically a **multi-layer perceptron, MLP**) can learn nonlinear feature interactions that our linear models and NB couldn't. For example, an MLP could implicitly learn the importance of water/cement ratio by having neurons that pick up on the combination of water and cement inputs. We propose to develop an MLP regressor for compressive strength, using one or two hidden layers with ReLU activation. This network can be given the same 8 features as input and trained to minimize RMSE. We will need to be cautious about overfitting, given the dataset size (1030 examples is not huge for a neural network). Techniques like **cross-validation**, **early stopping**, and **regularization (L2 or dropout)** will be employed to ensure the neural network model generalizes well. We will also scale inputs to 0–1 or standardize them, as neural nets tend to train more effectively on normalized data.

The plan is to start with a relatively small network (e.g. one hidden layer with ~16 neurons) and incrementally increase complexity if needed. We expect an MLP to outperform linear regression in RMSE if it can capture the curvature in age effects and interactions like cement *superplasticizer or water*flyash. For classification (high vs low strength), a neural network classifier could similarly learn nonlinear boundaries (perhaps improving a few percentage points in accuracy). We will evaluate the neural network on the same train/val/test split for fair comparison to our baselines. Success will be measured by a drop in validation/test MAE/RMSE for regression and an increase in classification accuracy/F1. In case of *failure modes* (e.g. the MLP overfits or doesn't significantly improve performance), we will experiment with feature engineering (like introducing the water-cement ratio as a new feature) or try ensembles (e.g. combining multiple models) to further boost performance.

In conclusion, the classical models have given us interpretable benchmarks and insights (e.g. confirming which factors most influence strength) and set expectations for error rates. The next stage will leverage these insights and the flexibility of neural networks to hopefully achieve **higher accuracy and lower error**, especially by capturing the inherently nonlinear nature of concrete strength development.