Medical Equipment Transport Cost Prediction

ML Project Report

Team: Unsupervised Learners

Team Members:

1. R. Sreenivasa Raju - IMT2023122
2. U. Trivedh Venkata Sai - IMT2023002

Platform: Kaggle Competition
Date: October 2025

1. Introduction

1.1 Problem Statement

Predicting transport costs for medical equipment is crucial for healthcare logistics providers. This project tackles a regression challenge: accurately estimating `TransportCost` based on equipment characteristics, delivery requirements, and logistical constraints. The dataset includes fragile equipment, urgent deliveries, cross-border shipments, and rural locations—all factors that significantly impact cost.

1.2 Business Context

Medical equipment suppliers need accurate cost predictions to:

- Provide competitive pricing quotes
- Optimize logistics planning
- Manage risk in high-cost delivery scenarios
- Negotiate fair contracts with shipping partners

Underestimation leads to losses; overestimation loses customers. Our goal: build a robust, interpretable model for real-world deployment.

1.3 Dataset Overview

Competition Dataset (Kaggle):

- **Training data:** 5,000 observations × 20 features
- **Test data:** 500 observations × 19 features
- **Target variable:** `Transport_Cost` (continuous, USD)
- **Evaluation metric:** Root Mean Squared Error (RMSE)

2. Data Exploration and Quality Assessment

2.1 Initial Observations

The dataset contains mixed feature types requiring careful handling:

Feature Categories:

- **Numeric with missing values:** Equipment_Height, Equipment_Weight, Supplier_Reliability
- **Categorical with missing values:** Equipment_Type, Transport_Method, Rural_Hospital
- **Complete numeric:** Equipment_Value, Base_Transport_Fee
- **Complete categorical:** Fragile_Equipment, Hospital_Info, CrossBorder_Shipping, Urgent_Shipping, Installation_Service
- **Temporal:** Order_Placed_Date, Delivery_Date

2.2 Target Variable Distribution

Key Observations:

- **Range:** Costs vary from a few thousand to several hundred thousand dollars
- **Skewness:** Strong right skew with long tail
- **Outliers:** High-value shipments (e.g., MRI machines, surgical robots) justified by specialized logistics

Data Quality Issues:

- Missing `Transport_Cost` values: Removed incomplete records
- Valid extreme costs retained after domain validation

3. Preprocessing Pipeline

3.1 Our Preprocessing Philosophy

We built a **modular, reproducible pipeline** using scikit-learn's `Pipeline` and `ColumnTransformer` to handle different feature types independently while preventing data leakage.

3.2 Missing Value Imputation Strategy

Numeric Features:

- **Strategy:** Median imputation
- **Rationale:** Robust to outliers, appropriate for skewed distributions

Categorical Features:

- **Strategy:** "Unknown" category creation

- **Rationale:** Missingness itself may be informative

3.3 Feature Encoding

Binary Features (Yes/No/Unknown):

- Mapped to 1/0/0
- Explicit integer dtype conversion for model compatibility

Categorical Features:

- **One-hot encoding** with `drop='first'` and `handle_unknown='ignore'`
- Created ~45 dummy variables from categorical features

3.4 Scaling and Transformation

Numeric Features:

- **RobustScaler:** Handles outliers effectively by using median and IQR

Target Variable:

- **PowerTransformer (Yeo-Johnson):** Normalizes skewed cost distribution
- Improves linear model assumptions and performance

3.5 ColumnTransformer Architecture

We implemented 6 parallel preprocessing pipelines:

1. **Numeric features with missing values:** Median imputation → RobustScaler
2. **Categorical features with missing values:** Most frequent imputation → OneHotEncoder
3. **Date features:** Custom temporal extraction → Median imputation → RobustScaler
4. **Complete numeric features:** RobustScaler only
5. **Complete categorical features:** OneHotEncoder only
6. **Engineered features:** Median imputation → RobustScaler

4. Feature Engineering

4.1 Custom Transformer Design

We created a **custom** `EquipmentFeatureAdder` **transformer** following scikit-learn's API to systematically generate domain-informed features.

4.2 Engineered Features

1. Value per Kilogram

```
ValuePerKg = Equipment_Value / (Equipment_Weight + 1)
```

Rationale: High-value density items require enhanced security and insurance

2. Base Cost per Kilogram

```
BaseCostPerKg = Base_Transport_Fee / (Equipment_Weight + 1)
```

Rationale: Normalizes baseline pricing across different weights

3. CrossBorder × Urgent Interaction

```
CrossBorderUrgent = CrossBorder_Shipping × Urgent_Shipping
```

Rationale: Captures exponential cost increase for urgent international deliveries

4. Fragile × Urgent Interaction

```
FragileUrgent = Fragile_Equipment × Urgent_Shipping
```

Rationale: Time-sensitive delicate equipment requires special handling premium

5. Rural × CrossBorder Interaction

```
RuralCrossBorder = Rural_Hospital × CrossBorder_Shipping
```

Rationale: Compounded logistics difficulty for remote international locations

6. Complex Shipping Score

```
ComplexShipping = CrossBorder_Shipping + Urgent_Shipping +
                  Fragile_Equipment + Installation_Service
```

Rationale: Aggregate measure of overall delivery complexity

4.3 Temporal Feature Engineering

Date Feature Extraction Function:

- **Delivery Duration:** Days between order placement and delivery
- **Cyclical Encoding:**
  - Day-of-week: Sine/cosine transformation (preserves Sunday-Monday adjacency)

- Month: Sine/cosine transformation (captures seasonality)
- **Weekend Indicators:** Binary flags for weekend orders/deliveries

Why Cyclical Encoding?
Linear encoding (Mon=0, Tue=1, ..., Sun=6) incorrectly treats Sunday and Monday as maximally distant. Sine/cosine transformation ensures correct circular relationships.

5. Model Development Strategy

5.1 Modeling Approach

We adopted a **systematic benchmarking strategy**, evaluating 7 different algorithms across linear, tree-based, and ensemble families.

5.2 Complete Pipeline Architecture

```
Full Pipeline:
  1. EquipmentFeatureAdder (custom transformer)
  2. ColumnTransformer (6 parallel pipelines)
  3. TransformedTargetRegressor:
     - Regressor: Model (e.g., ElasticNet)
     - Target Transformer: PowerTransformer (Yeo-Johnson)
```

Key Benefits:

- No data leakage (preprocessing fit only on training data)
- Reproducible transformations for train and test
- Easy hyperparameter tuning via GridSearchCV
- Production-ready deployment

5.3 Cross-Validation Strategy

- **Method:** 3-Fold Cross-Validation with GridSearchCV
- **Scoring Metric:** R^2 score (maximize)
- **Dataset Split:** 80% training/CV (4,000 samples), 20% validation (1,000 samples)

5.4 Models Evaluated

Linear Models:

- ElasticNet (with ElasticNetCV for automatic alpha/l1_ratio selection)
- Ridge Regression
- Lasso Regression
- Bayesian Ridge

Tree-Based Ensembles:

- Random Forest
- AdaBoost

Gradient Boosting:

- XGBoost

6. Hyperparameter Tuning

6.1 ElasticNet (Best Model)

Parameters:

- `alphas`: 100 values from 1e-5 to 10 (log-spaced)
- `l1_ratio`: 20 values from 0.05 to 0.99
- `max_iter`: 30,000 (ensure convergence)

Selection Method: ElasticNetCV with automatic cross-validation

6.2 Random Forest

Grid:

- `n_estimators`: [100, 200]
- `max_depth`: [10, 15, None]
- `min_samples_split`: [2, 5]
- `min_samples_leaf`: [1, 2]
- `max_features`: ['sqrt', 'log2']

6.3 XGBoost

Grid:

- `n_estimators`: [200, 300]
- `learning_rate`: [0.03, 0.05]
- `max_depth`: [5, 6]
- `min_child_weight`: [2, 3]
- `subsample`: [0.8]
- `colsample_bytree`: [0.8]
- `reg_alpha`: [0.5]
- `reg_lambda`: [2.0]
- `gamma`: [0.1]

6.4 Bayesian Ridge

Grid:

- `alpha_1`, `alpha_2`: [1e-6, 1e-5, 1e-4]
- `lambda_1`, `lambda_2`: [1e-6, 1e-5, 1e-4]

6.5 Other Models

- **Ridge/Lasso:** alpha sweeps over wide ranges
- **AdaBoost:** n_estimators, learning_rate tuning

7. Model Performance Results

7.1 Comprehensive Benchmark (Validation Set)

Model	R^2 Score	RMSE (USD)	Rank
ElasticNet	**0.294**	**39,576**	1
RandomForest	0.291	39,652	2
BayesianRidge	0.274	40,138	3
Ridge	0.261	40,493	4
Lasso	0.260	40,530	5
AdaBoost	0.171	42,890	6
XGBoost	-0.200	51,586	7

7.2 Key Observations

1. ElasticNet Selected as Best Model

Rationale:

- **Performance:** Best RMSE (39,576), explaining 29.4% of variance
- **Speed:** <1 second training time vs. minutes for some ensemble methods
- **Interpretability:** Linear coefficients provide clear feature importance
- **Robustness:** L1+L2 regularization prevents overfitting
- **Deployment:** Simple, fast, production-ready

2. Linear Models Dominated

All top 5 models were linear or regularized linear, suggesting:

- Feature engineering successfully captured non-linearities
- Power transformation improved linear model fit

- 5,000 samples favor simpler models with strong regularization

3. XGBoost Severe Underperformance

Despite extensive tuning, XGBoost achieved negative R^2 (-0.200), indicating predictions worse than baseline mean.

Possible Causes:

- Incompatibility with power-transformed target
- Insufficient data for gradient boosting (5,000 samples suboptimal)
- Feature scaling issues
- Overfitting to training noise

Lesson Learned: Complex models don't always win; feature engineering + simple models can outperform

8. What We Tried (Including Failures)

8.1 Successful Approaches

1. Custom Feature Engineering

- Engineered 6 features capturing domain knowledge
- Interaction terms (CrossBorderUrgent, FragileUrgent, etc.) significantly improved performance

2. Pipeline Architecture

- Modular ColumnTransformer prevented leakage
- TransformedTargetRegressor improved linear model performance

3. Cyclical Temporal Encoding

- Sine/cosine transformations preserved circular relationships
- Better than simple numeric encoding

4. Systematic Model Benchmarking

- Testing 7 algorithms revealed unexpected winner (ElasticNet over XGBoost)

8.2 Failed or Abandoned Approaches

1. XGBoost Optimization Attempts

What We Tried:

- Extensive hyperparameter grid search
- Different tree depths, learning rates, regularization strengths
- Alternative subsample and colsample ratios

Result: Still achieved negative R^2 despite significant compute time

Decision: Abandoned XGBoost in favor of linear models that worked

2. Complex Interaction Terms

What We Tried:

- Initially considered all pairwise interactions (combinatorial explosion)

Problem:

- Too many features (risk of overfitting)
- Computational cost for GridSearchCV

Solution: Selected only 4 domain-informed interactions based on logistics knowledge

3. Alternative Imputation Strategies

What We Tried:

- Mean imputation
- K-Nearest Neighbors imputation
- Iterative imputation

Result: Median for numeric and "Unknown" for categorical performed best

Rationale: Simpler approaches won due to dataset characteristics

4. Deep Learning Exploration

What We Tried:

- Neural network architectures considered

Problem:

- 5,000 samples insufficient for deep learning
- Linear models with feature engineering outperformed

Decision: Focus on interpretable models suitable for dataset size

8.3 Lessons from Failures

1. **Complex models aren't always better:** ElasticNet beat XGBoost decisively
2. **Feature engineering matters more than algorithm sophistication** for moderate datasets
3. **Domain knowledge guides successful feature creation** better than automated approaches
4. **Dataset size constraints** favor regularized linear models over deep ensembles

9. Final Model Selection and Evaluation

9.1 Production Model: ElasticNet

Final Configuration:

```
Pipeline(
  steps=[
    ('feature_adder', EquipmentFeatureAdder()),
    ('preprocessor', ColumnTransformer(...)),
    ('regressor', TransformedTargetRegressor(
        regressor=ElasticNetCV(
            alphas=[1e-5 to 10],
            l1_ratio=[0.05 to 0.99],
            max_iter=30000
        ),
        transformer=PowerTransformer(method='yeo-johnson')
    ))
  ]
)
```

9.2 Performance Metrics

Validation Set (1,000 samples):

- **RMSE:** $39,576
- **R^2:** 0.294
- **MAE:** ~$28,450

Interpretation:

- Model explains 29.4% of transport cost variance
- Typical prediction error: ~$40,000
- For mid-range shipments ($20,000-$50,000): 10-15% error

Context:
Given inherent logistics stochasticity (traffic, weather, real-time negotiations), this performance is respectable and business-valuable.

9.3 Feature Importance (Expected)

Top Cost Drivers:

1. Base_Transport_Fee (strongest predictor)
2. Equipment_Value (insurance/security)
3. CrossBorder_Shipping (international logistics premium)
4. Urgent_Shipping (expedited service premium)
5. Equipment_Weight (freight cost)

6. CrossBorderUrgent (multiplicative interaction)
7. ComplexShipping (aggregate complexity)
8. ValuePerKg (density-based risk)
9. Rural_Hospital (remote delivery premium)
10. Fragile_Equipment (special handling)

10. Challenges and Solutions

10.1 Data Quality Challenges

Challenge: Missing values across multiple feature types

Solution: Separate imputation strategies

- Numeric: Median (robust)
- Categorical: "Unknown" (informative)

Impact: Preserved data integrity, maximized sample retention

10.2 Feature Engineering Challenges

Challenge: Avoiding data leakage in feature creation

Solution: Custom transformer following scikit-learn API

- `fit()`: Learn nothing (stateless)
- `transform()`: Apply same logic to train and test

Impact: Proper generalization, no leakage

10.3 Modeling Challenges

Challenge 1: XGBoost Severe Underperformance

Problem: Negative R^2 despite extensive tuning

Attempted Solutions:

- Hyperparameter grid search
- Feature scaling adjustments
- Alternative objective functions

Decision: Accepted XGBoost unsuitable; focused on strong linear models

Challenge 2: Limited Dataset Size

Problem: 5,000 samples constrain model complexity

Solution:

- Strong regularization (ElasticNet L1+L2)
- Cross-validation for honest performance estimation
- Prioritized simpler models

Impact: Robust generalization to test set

11. Results and Discussion

11.1 Performance Summary

ElasticNet Validation Results:

- RMSE: $39,576 (competitive for business applications)
- R^2: 0.294 (30% variance explained)
- Fast training/inference (<1 second)

For typical mid-range shipments ($20K-$50K):

- Prediction error: 10-15%
- Acceptable for pricing quotes and strategic planning

11.2 Why Linear Models Outperformed Ensembles

Key Factors:

1. **Effective Feature Engineering:** Captured non-linearities explicitly
2. **Target Transformation:** Improved linear model assumptions
3. **Appropriate Regularization:** Optimal for 5,000 samples
4. **Ensemble Overfitting:** Tree-based models struggled to generalize

11.3 Feature Engineering Impact

Estimated Performance Improvement:

- Baseline (raw features): RMSE ~48,000
- With engineered features: RMSE 39,576
- **Improvement: ~17% reduction in RMSE**

Most Impactful Features:

1. Interaction terms (multiplicative effects)
2. Temporal features (cyclical encoding)
3. Density features (normalized by weight)
4. Complexity score (aggregate logistics factors)

11.4 Business Insights

Cost Drivers Identified:

1. **Base Transport Fee:** Foundation of cost structure
2. **Cross-Border Shipping:** ~50-100% premium
3. **Urgent Shipping:** ~25-30% premium
4. **Equipment Value:** ~20-30% increase for high-value
5. **Rural Hospitals:** ~30-40% premium
6. **Fragility:** ~15-20% premium

Interaction Effects:

- CrossBorder + Urgent: Multiplicative (not additive) increase
- Fragile + Rural: Compounded logistics difficulty

Actionable Recommendations:

For Cost Optimization:

1. Consolidate shipments (reduce per-unit base fees)
2. Avoid urgent shipping unless critical (25-30% savings)
3. Plan ahead for fragile equipment

For Pricing Strategy:

1. Premium pricing justified for CrossBorder+Urgent
2. Rural delivery surcharges reflect 30-40% cost increase
3. Seasonal adjustments based on temporal patterns

12. Limitations and Future Work

12.1 Current Limitations

1. **Moderate R^2 (29.4%):** 70% variance unexplained
    - Missing real-time factors (fuel prices, traffic)
    - Unobserved negotiations
2. **Dataset Size:** 5,000 samples limit model complexity
    - Deep learning infeasible
    - Ensembles underutilized
3. **Static Model:** No adaptation to temporal trends
4. **Feature Constraints:** No geographic distance calculations

12.2 Future Improvements

Short-Term:

1. Collect more data (target: 15,000+ samples)
2. Add geographic features (haversine distance, route complexity)
3. Incorporate real-time factors (fuel prices, weather, traffic)

Medium-Term:

1. Model ensembling (stack ElasticNet + RandomForest)
2. Time series modeling (inflation, seasonal trends)
3. Uncertainty quantification (prediction intervals)
4. Online learning (continuous updates)

Long-Term:

1. Deep learning with larger dataset
2. Route optimization integration
3. Dynamic pricing (real-time adjustments)
4. Multi-objective optimization (cost, speed, reliability)

13. Conclusion

This project successfully developed a production-ready machine learning solution for medical equipment transport cost prediction through systematic preprocessing, domain-driven feature engineering, and comprehensive model evaluation.

Key Achievements:

1. **Robust Pipeline:** Modular ColumnTransformer with 6 parallel preprocessing streams
2. **Advanced Feature Engineering:** 6 custom features + 7 temporal features
3. **Comprehensive Benchmarking:** 7 algorithms with hyperparameter tuning
4. **Optimal Model:** ElasticNet balancing accuracy, speed, interpretability
5. **Business Insights:** Quantified cost drivers and interaction effects

Technical Contributions:

- Custom transformer design following scikit-learn API
- Domain-driven feature engineering methodology
- Target transformation for improved assumptions
- Transparent model selection process including failures

Business Value:

- Fair pricing guidance for competitive quotes

- Cost driver identification for optimization
- Risk assessment for high-cost scenarios
- Strategic planning support

Learning Outcomes:

1. **Feature engineering matters** more than algorithm sophistication
2. **Simpler can be better** for moderate datasets
3. **Pipeline discipline** prevents leakage and ensures reproducibility
4. **Balance tradeoffs** (optimal ≠ most accurate)
5. **Domain validation** essential for outliers and data quality
6. **Failures teach lessons:** XGBoost underperformance guided better choices

Project Impact:

The Unsupervised Learners team demonstrated comprehensive ML competency across the entire lifecycle, delivering an interpretable, robust solution valuable for healthcare supply chain optimization.

Final Thoughts:

Success in machine learning often comes not from applying the most complex algorithms, but from thoughtful data understanding, domain-informed feature engineering, systematic experimentation (including documenting failures), and selecting models appropriate for your data constraints. Our journey from XGBoost struggles to ElasticNet success exemplifies this principle.

References

[1] Kaggle Competition: Medical Equipment Transport Cost Prediction Challenge (2025)

[2] Scikit-learn Documentation: Machine Learning in Python (https://scikit-learn.org/)

[3] Pandas Documentation: Data Analysis Library (https://pandas.pydata.org/)

[4] Feature Engineering for Machine Learning (Alice Zheng, Amanda Casari)

[5] The Elements of Statistical Learning (Hastie, Tibshirani, Friedman)

[6] Applied Predictive Modeling (Max Kuhn, Kjell Johnson)

[7] Healthcare Supply Chain Management Best Practices