Medical Equipments Cost Prediction Challenge

AIT 511 / Machine Learning

Assignment: Checkpoint 1

Project Overview

Aim

Predict the cost required to transport medical equipment to hospitals based on the information provided in the dataset. Kaggle challenge page mentioned in References [1]

Data set folder taken from kaggle includes

1. train.csv: **5000 x 20**
2. test.csv: **500 x 19**

Features Overview

#	Column	Non-Null Count	Dtype
0	Hospital_Id	5000 non-null	object
1	Supplier_Name	5000 non-null	object
2	Supplier_Reliability	4413 non-null	float64
3	Equipment_Height	4717 non-null	float64
4	Equipment_Width	4557 non-null	float64
5	Equipment_Weight	4540 non-null	float64
6	Equipment_Type	4401 non-null	object
7	Equipment_Value	5000 non-null	float64
8	Base_Transport_Fee	5000 non-null	float64
9	CrossBorder_Shipping	5000 non-null	object
10	Urgent_Shipping	5000 non-null	object
11	Installation_Service	5000 non-null	object
12	Transport_Method	3929 non-null	object
13	Fragile_Equipment	5000 non-null	object
14	Hospital_Info	5000 non-null	object
15	Rural_Hospital	4414 non-null	object
16	Order_Placed_Date	5000 non-null	object
17	Delivery_Date	5000 non-null	object
18	Hospital_Location	5000 non-null	object
19	Transport_Cost	5000 non-null	float64

Figure 1: Training Data Specs

Data Processing Steps

Step 1 : Import Libraries and Load Dataset

The initial step in the EDA process involves importing essential libraries such as pandas, numpy, matplotlib, seaborn, and relevant modules from scikit-learn for data preprocessing. The dataset is loaded into a pandas DataFrame for further analysis and manipulation.

'Libraries Used'

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

```

```python
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, RobustScaler, FunctionTransformer, PowerTransformer, PolynomialFeatures

from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from sklearn.linear_model import LinearRegression, ElasticNetCV, Lasso, Ridge
from sklearn.kernel_ridge import KernelRidge
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from xgboost import XGBRegressor

from sklearn.svm import SVR
from sklearn.compose import TransformedTargetRegressor

```

Step 2 : Feature Grouping

'Feature Grouping'

```python
# Unique Columns - should be dropped
drop_cols = ["Hospital_Id", "Supplier_Name", "Hospital_Location"]

# Numerical features with missing values
numeric_features_median = ['Equipment_Height', 'Equipment_Weight', 'Supplier_Reliability']

# Categorical Features with missing values.
categorical_features_unknown = ['Equipment_Type', 'Transport_Method', 'Rural_Hospital']

# Numerical features with no missing values
no_missing_features_numerical = ['Equipment_Value', 'Base_Transport_Fee']

# Categorical features with no missing values
no_missing_features_categorical = ['Fragile_Equipment', 'Hospital_Info', 'CrossBorder_Shipping', 'Urgent_Shipping', 'Installation_Service']

# Date features
date_features = ['Order_Placed_Date', 'Delivery_Date']
```

We grouped features so that dealing with the same kind of features can be done in a simple pipeline process (instead of using pipeline, it can also be done using arrays and for loops)

Step 3 : Handling Date Feature

It is important to encode **cyclical features** properly.

Why is this needed ? Because although months 12 and 1 are adjacent in time, their raw numerical values suggest otherwise. If we feed the model the raw month numbers, it may incorrectly assume that 12 is much larger than 1, leading to misleading patterns and poor learning.

'Dealing with Cyclic features like Date'

```python
def cyc(x, period):
    xr = x.replace(-1, 0)
    rad = 2 * np.pi * xr / period

    return np.sin(rad), np.cos(rad)
```

Step 4 : Exploratory Data Analysis (EDA)

Various plots were generated to analyse the data distribution, detect outliers, and explore relationships among features. These visualisations provide critical insights that inform subsequent preprocessing and modelling decisions. Key visualisations and their interpretations are detailed below:

Histogram

For each numerical feature, histograms and kernel density estimates (KDEs) were plotted to observe the data's spread, skewness, and distribution shape.

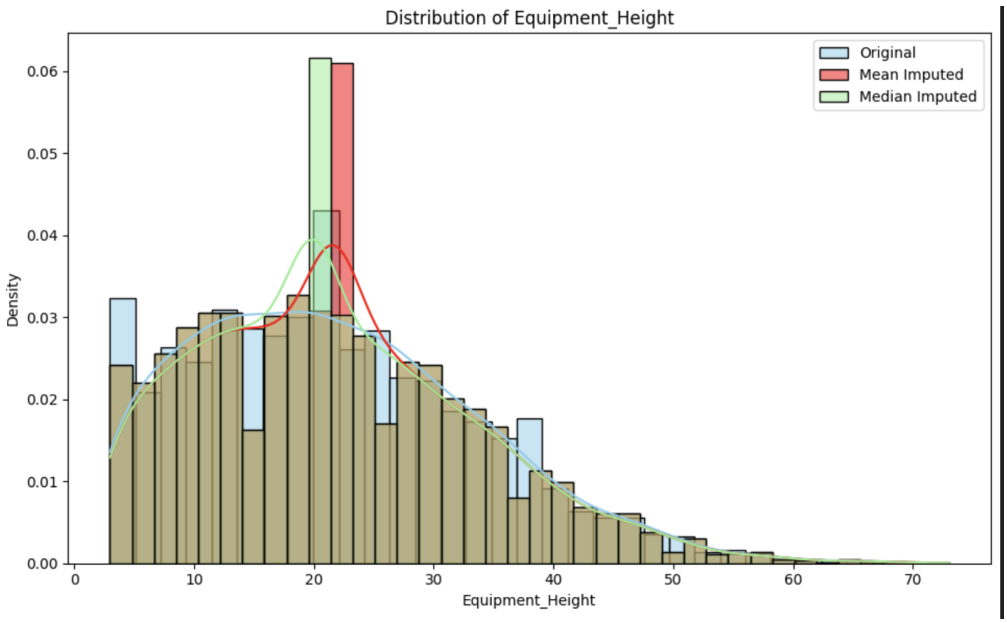

Figure 2: Equipment Height Feature Data Distribution

> **Key Takeaways**
>
> ✔ The original data displays a right-skewed (positively skewed) distribution, with a higher frequency of lower equipment heights and a long tail extending toward larger values.
> ✔ The original data displays a right-skewed (positively skewed) distribution, with a higher frequency of lower equipment heights and a long tail extending toward larger values.
> ✔ The original data displays a right-skewed (positively skewed) distribution, with a higher frequency of lower equipment heights and a long tail extending toward larger values.

Boxplots

Boxplots were used to examine each numerical feature for potential outliers, which appear as points beyond the plot's whiskers.
In the given data set there were few data points outside the whiskers of feature 'equipment height' as shown in the plot.

> **Key Takeaways**
>
> ✔ There are several outliers on the higher end (beyond the upper whisker), indicating a small number of equipment entries with heights significantly above the typical range, which matches the right-skewed distribution seen before.

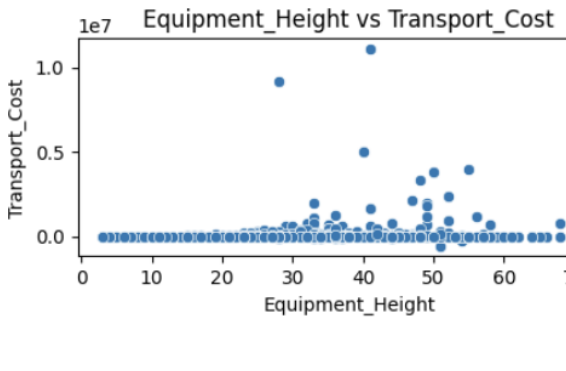

Figure 3: Scattered Plot for feature 'equipment Height'

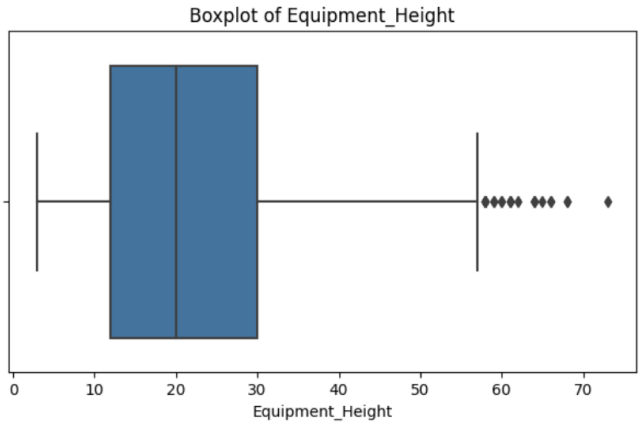

Figure 4: Box Plot for feature 'equipment Height'

- The median (middle line of the box) is around 22-23, and the interquartile range (IQR—the box itself) spans roughly from 12 to 32, showing that most equipment heights fall within this central region.
- The whiskers extend much further to the right, with the upper whisker reaching values above 50, confirming strong positive skewness and the presence of high-value outliers in the data.

Correlation Heat Map

The correlation matrix heatmap reveals the relationships between different variables in the dataset.

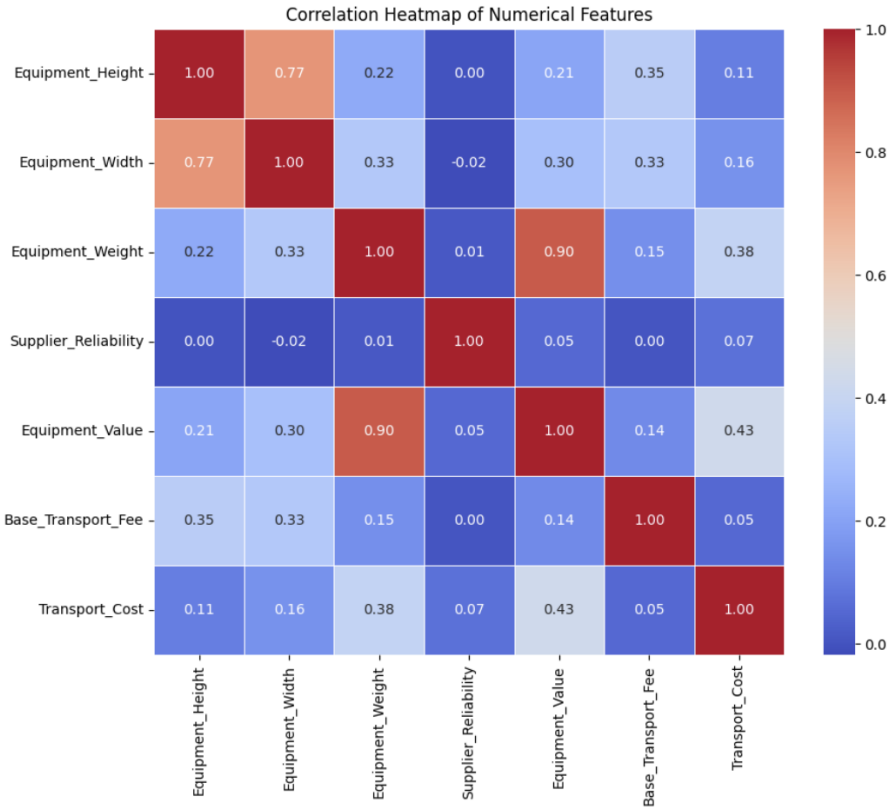

Figure 5: Correlation heat map using only Numeric Columns

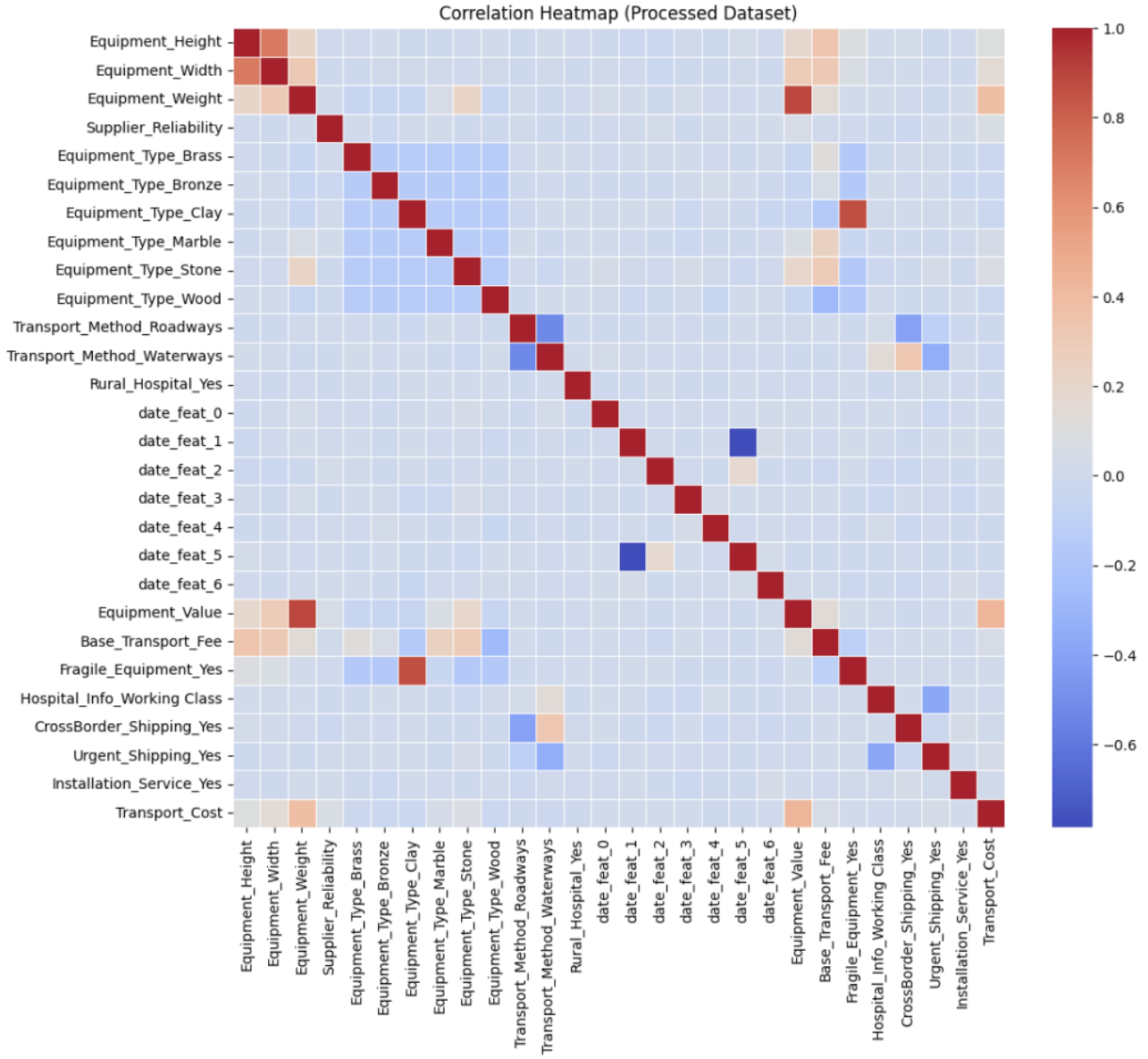

Figure 6: Correlation heat map after preprocessing data

Key Takeaways

✔ Most feature pairs have low absolute correlation values (close to 0, light-colored), suggesting that multicollinearity is generally not a concern in this dataset.
✔ Some feature pairs, such as 'Equipment_Height' and 'Equipment_Weight' show a moderate positive correlation (darker red), which is expected since physical attributes of equipment often scale together.
✔ There are a few strong negative or positive correlations involving encoded or engineered features (e.g., Equipment_Type_Marble with some target), indicating these may be important for modeling but should be checked for possible redundancy or explanation power in ML tasks.

Step 5 : Data Preprocessing

Data Preprocessing includes:

1. Feature Engineering
2. Handling missing values
3. Scaling of dataset
4. Encoding

Before doing anything first we need to clean our data set from bad data. Here unique value containing rows are bad data so we are going to drop them.

```
drop_cols = ["Hospital_Id", "Supplier_Name", "Hospital_Location"]
train = train.drop(columns=drop_cols)
test = test.drop(columns=drop_cols)
```

Feature Engineering

We applied feature engineering for **Date** data frame.

New Feature	Type	Relation to Original Data
delivery_days	Numeric	Difference between delivery date and order date (in days).
order_dow_sin	Numeric	Cyclical encoding of order day-of-week using sine function.
order_dow_cos	Numeric	Cyclical encoding of order day-of-week using cosine function.
order_month_sin	Numeric	Cyclical encoding of order month using sine function.
order_month_cos	Numeric	Cyclical encoding of order month using cosine function.
order_is_weekend	Binary	1 if the order is on a weekend (Saturday or Sunday), else 0.
delivery_is_weekend	Binary	1 if the delivery is on a weekend (Saturday or Sunday), else 0.

Table 1: Engineered Features from Date Columns

We also tried applying feature engineering for other columns but the result was bad so we dropped these features later.

New Feature	Type	Relation to Original Data
Equipment_Volume	Numeric	Product of Equipment Height, Width, and Weight, representing the overall equipment size or space occupied.
Value_to_Weight_Ratio	Numeric	Ratio of Equipment Value to its Weight, indicating value density or cost efficiency.
Is_Heavy	Binary	1 if Equipment Weight is greater than the median weight, otherwise 0. Identifies heavier-than-average items.
Fragile_Heavy	Binary	Product of Fragile_Equipment and Is_Heavy, representing items that are both fragile and heavy.
Urgent_CrossBorder	Binary	Product of Urgent_Shipping and CrossBorder_Shipping, indicating high-priority international shipments.
HighValue_Heavy	Numeric	Product of Equipment Value and Is_Heavy, capturing items that are valuable and heavy simultaneously.
Value_into_Weight_Ratio	Numeric	Product of Equipment Value and Equipment Weight, emphasizing large-value, heavy items.

Table 2: Engineered Features Derived from Equipment and Shipping Attributes

Handling missing values

1. Numerical Columns
   Handling numerical columns you need to see the data distribution of the feature and based on that you need to decide whether to use mean or median to fill missing values. When we saw the variance comparison for all the numerical data features all showed replacing them with median is good.

	Original	Mean Imputed	Median Imputed
Supplier_Reliability	7.073668e-02	6.243053e-02	6.244290e-02
Equipment_Height	1.426695e+02	1.345928e+02	1.347582e+02
Equipment_Width	2.924151e+01	2.665019e+01	2.685310e+01
Equipment_Weight	7.888458e+12	7.162575e+12	7.176390e+12

Figure 7: Variance Comparision Original vs Mean vs Median for each data feature

```
'Handling Missing values of Numerical Columns'

numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', RobustScaler())
])
```

2. Categorical Columns
   For categorical columns, we took a look at the count of each category the feature has. If a certain category count far exceeds others then we replaced the missing values with it, but if they are nearly equal then we tried replacing missing values with a new category named as **'Unknown'**.

```
'Handling Missing values of Categorical Columns'

categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent', add_indicator=True)),
    ('encoder', OneHotEncoder(drop='first', handle_unknown='ignore'))
])
```

3. Date Column

```
'Handling Missing values of Date Columns'

date_transformer = Pipeline([
    ('date_feat', FunctionTransformer(compute_date_features, validate=False)),
    ('imputer', SimpleImputer(strategy='median', add_indicator=True)),
    ('scaler', RobustScaler(with_centering=False))
])
```

Scailing of Data set

Scaling of all numerical columns is done using ***Robust Scaling method***. Why ? Because to keep our dataset insensitive to outliers.

Encoding

We used ***One hot*** encoding to encode the categorical columns as shown in above snippet.

Step 6 : Training-Validation Split

The dataset is split into training and testing sets, ensuring that model performance can be evaluated on unseen data. A common split ratio is 80% for training and 20% for testing. This step prepares the dataset for model training and evaluation.

'Split Data into Training and Validation'

```
# Split processed data into 80% train and 20% validation
X_train, X_val, y_train, y_val = train_test_split(
    X_processed_df,  # processed features
    y,               # target
    test_size=0.2,   # 20% validation, 80% training
    random_state=42  # for reproducibility
)

# Check shapes
print("X_train:", X_train.shape)
print("X_val:", X_val.shape)
print("y_train:", y_train.shape)
print("y_val:", y_val.shape)
```

Models Used and Hyperparameter tuning

Models Used:

1. Linear Regression
2. Polynomial Regression
3. Lasso Regression
4. Ridge Regression
5. Random Forest
6. Adaboost
7. XG Boosting
8. Elastic Net Regression

Linear Regression

This section implements a Linear Regression model for predicting hospital transport costs. The code constructs a robust preprocessing pipeline that imputes missing values, scales numerical columns, encodes categorical variables using one-hot encoding, and generates date-based cyclical features. After preprocessing, the model is trained, validated on unseen data, and evaluated using MAE and R^2 metrics. Finally, predictions on the test dataset are generated and exported as a submission file.

Data processing: same data processing steps are followed.

> **Key Takeaways**
>
> Linear Regression
> - R^2 = -0.064
> - MSE = 25985.28
> - Leaderboard score = 6515164682.050

Polynomial Regression

Polynomial Regression extends the linear model by fitting polynomial transformations of the predictors, allowing it to capture non-linear relationships. Using features like Equipment_Value, Equipment_Weight, and Equipment_Width, the model was assessed with R^2 and MAE to quantify its ability to explain variance and its prediction accuracy.

> **Key Takeaways**
>
> Polynomial Regression
> - R^2 = -7.409
> - MSE = 58293.44
> - Leaderboard score = 29309575885.102

Lasso Regression

Lasso Regression was used with the same numerical predictors, incorporating L1 regularization to reduce overfitting and perform implicit feature selection by driving less important coefficients to zero. Model evaluation using R^2 and MAE highlighted its predictive performance and the relative importance of each feature.

Data Processing: same as discussed in above section.

⇒**Initial Lasso Params:** Lasso(
alpha=1.0,
max_iter=10000,
random_state=42))

> **Key Takeaways**
>
> Lasso - fixed
> - R^2 = -0.064
> - MSE = 25985.22
> - Leaderboard score = 6515160094.766

Applied Grid Search to find out best parameters.

⇒**Best Lasso Params:** Lasso Params:
alpha = 10,
max_iter=10000,
random_state=42

> **Key Takeaways**
>
> Lasso using GridSearch
> - Best CV R^2 = 0.0180
> - R^2 = -0.061
> - MAE = 25922.39
> - Leaderboard score = 6510598453.306

Ridge Regression

Ridge Regression was applied to the numerical features, employing L2 regularization to penalize large coefficients and improve generalization while retaining all predictors. Model performance was measured using R^2 and MAE, indicating its capacity to explain variance and average prediction error.

Data processing: For this model too , we have done data preprocessing same as we done for Linear regression.

⇒**Initial Ridge Params:** Ridge(
alpha=1.0,
max_iter=10000,
random_state=42))

> **Key Takeaways**
>
> Ridge - fixed
> - R^2 = -0.063
> - MSE = 25965.91
> - Leaderboard score = 6513573403.475

Applied Grid Search to find out best parameters.

⇒**Best Ridge Params:** Ridge Params:
alpha = 100,
max_iter=10000,
random_state=42

> **Key Takeaways**
>
> Ridge using GridSearch
> - Best CV R^2 = 0.0326
> - R^2 = -0.019
> - MAE = 24467.35
> - Leaderboard score = 6387594703.305

Random Forest

Random Forest, an ensemble of decision trees, was trained to capture complex non-linear relationships among Equipment_Value, Equipment_Weight, and Equipment_Width. By averaging predictions from multiple trees, the model reduces overfitting. Performance was evaluated using R^2 and MAE, providing insights into its predictive accuracy.

Data processing: For this model too , we have done data processing same as we done for Linear regression.
⇒**Initial RF Params:** RandomForestRegressor(
n_estimators=200,
max_depth=10,
random_state=42,
n_jobs=-1))])

> **Key Takeaways**
>
> RF - fixed
> - R^2 = -0.642
> - MSE = 17747.77
> - Leaderboard score = 17212452660.530

Applied Grid Search to find out best parameters.

⇒**Best Random Forest Params:** Best RF Params:
'regressor__max_depth': 15,
'regressor__min_samples_leaf': 4,
'regressor__min_samples_split': 5,
'regressor__n_estimators': 200

> **Key Takeaways**
>
> Random Forest - Grid Search
> - Best CV R^2 = -0.241
> - R^2 = -0.279
> - MAE = 7558.19
> - Leaderboard score = 20906750713.724

AdaBoost

AdaBoost combines sequential weak learners, where each subsequent model focuses on the errors of the previous ones to improve overall predictions. Using numerical features, the ensemble was assessed with R^2 and MAE, demonstrating how well it reduces bias and captures variance in the target.

Data processing: For this model too , we have done data processing same as we done for Linear regression.

⇒**Intial AdaBoost Params:** AdaBoostRegressor(
random_state=42))

> **Key Takeaways**
>
> Ada Boost - fixed params
> - R^2 = -0.705
> - MSE = 13651.95
> - Leaderboard score = 11090862349.089

Applied Grid Search to find out best parameters.

⇒**Best AdaBoost Params:** AdaBoost Params:
'regressor__learning_rate': 0.1,
'regressor__loss': 'square',
'regressor__n_estimators': 100

> **Key Takeaways**
>
> Ada Boost - grid search
> - Best CV R^2 = -0.119
> - R^2 = 0.243
> - MAE = 8428.03
> - Leaderboard score = 6240594182.929

XG Boost

XGBoost implements gradient boosting on decision trees, iteratively combining weak learners to minimize prediction error. Using features like Equipment_Value, Equipment_Weight, and Equipment_Width, model performance was evaluated with R^2 and MAE, reflecting its ability to accurately capture complex relationships.

Data processing: same as mentioned in above section.
⇒**Intial Fixed Params:** XGBRegressor(
n_estimators=500,
learning_rate=0.05,
max_depth=6,
subsample=0.8,
colsample_bytree=0.8,
random_state=42,
n_jobs=-1))

> **Key Takeaways**
>
> XG Boost fixed params
> - R^2 = -5.710
> - MSE = 12976.56
> - Leaderboard score = 24642928711.251

Applied Grid Search to find out best parameters.

⇒**Best XG boost Params:** XGB Params:
'regressor__colsample_bytree': 0.8,
'regressor__learning_rate': 0.01,
'regressor__max_depth': 7,
'regressor__n_estimators': 200,
'regressor__subsample': 1.0

> **Key Takeaways**
>
> XG Boost using Grid Search
> - Best CV R^2 = 0.165
> - R^2 = -2.588
> - MAE = 12469.355
> - Leaderboard score = 19417933355.693

Elastic Net Regression - **best leaderboard score**

Elastic Net combines both L1 and L2 regularization, balancing feature selection and coefficient shrinkage. It was implemented in a pipeline with preprocessing and a target transformation (Yeo–Johnson) to stabilize variance.

Model hyperparameters alpha and l1_ratio are optimized using **GridSearchCV** with 5-fold cross-validation to maximize the R^2 score.

Data Processing: same as mentioned in above section, and we also removed outliers below 4.4 percentile, which decreased our leaderboard score (quite shocking for us)

⇒ **Best ElasticNet params:**
'regressor__regressor__alpha': 0.001,
'regressor__regressor__l1_ratio': 0.2

> **Key Takeaways**
>
> Elastic Net
> - $R^2 = 0.0144$
> - MSE = 297813.64
> - Leaderboard score = 3985281371.543

Tested Some Other Models

We have named the file **'Best Model'** but it is not actually. We have tried three different models but again we got Elastic Regression better.

Model	R^2	RMSE
ElasticNet	0.2952	39,540.41
KernelRidge	0.0357	46,250.26
SVR	0.3901	36,782.20

Table 3: Model Performance Comparison

Best Parameters:

- **Elastic Net (fixed):** cv=5, max_iter=20000, random_state=42
- **Kernel Ridge:** $\alpha = 1.0$, $\gamma = 0.01$, kernel = `rbf`
- **SVR:** $C = 1$, $\epsilon = 0.3$, $\gamma =$ `scale`

Best Model: ElasticNet
Reason: Although SVR achieved the highest R^2 score (0.3901), the ElasticNet model was selected as the best due to its balance between model complexity, generalization performance, and interpretability. The final submission file generated was `submission_best_model.csv`.

Summary

We found out that the Elastic Net Regression model gave us the best leaderboard score for this dataset.

Table 4: Model Leaderboard Scores (Increasing Order)

Model	Leaderboard Score
Elastic Net CV	3,985,281,371.543
Elastic Net (Fixed)	4,523,090,717.491
Linear 2 Features	5,376,252,785.576
Ada Boost (GridSearch)	6,240,594,182.929
Ridge Regression (GridSearch)	6,387,594,703.305
Lasso Regression (GridSearch)	6,510,598,453.306

(Continued on next page)

Model	Leaderboard Score
Ridge Regression (Fixed)	6,513,573,403.475
Lasso Regression (Fixed)	6,515,160,094.766
Linear Regression	6,515,164,682.050
Polynomial Regression	9,309,575,885.102
Ada Boost (Fixed)	11,090,862,349.089
Random Forest (Fixed)	17,212,452,660.530
XG Boost (GridSearch)	19,417,933,355.693
Random Forest (GridSearch)	20,906,750,713.724
XG Boost (Fixed)	24,642,928,711.251

After Elastic Net, We have obsereved that Ada Boost dominates linear regression, but when I do feature selection I got Linear regression better than Adaboost.

Team Members

Project Katakam - (Team Name)

S.No.	Name (Roll No.)	Role
1	Mohit Jagini (IMT2023528)	Member
2	Katakam Shashidhar Sai (IMT2023567)	Team Leader
3	Hardhik Dhavala (IMT2023579)	Member

> **Note**
>
> Click here to email the team leader with team members in CC

References

[1] A. Subhedar, K. Chandak, K. Goyal, R. Ratnani, S. Chaturvedi, T. 1812, and V. II-ITB, "Medical equipments cost prediction challenge." https://www.kaggle.com/competitions/Medical-Equipments-Cost-Prediction-Challenge, 2025. Kaggle Competition.

[2] P. Katakam, "Medical equipment cost prediction: Project repository." https://github.com/DHardhik/ML-Challenge, 2025. Accessed on October 25, 2025.