# Project Report on Diamond Price Prediction

**- Bhagya Patel**

This comprehensive technical report presents an in-depth analysis of a machine learning system designed to revolutionize diamond pricing through automated valuation. The project implements multiple regression models, leveraging both traditional statistical methods and cutting-edge machine learning techniques to create a robust pricing model for the diamond industry.

## 1. Problem Definition

## 1.1 Business Context

The diamond industry faces a complex challenge in accurately pricing diamonds based on their various characteristics. Traditional pricing methods often rely heavily on human expertise and can be inconsistent or subjective. This creates a need for a more systematic, data-driven approach to diamond valuation. Here are some traditional problems:

* Subjective valuation processes leading to inconsistent pricing
* Time-intensive manual appraisals
* Limited scalability of expert-based pricing
* Market inefficiencies due to information asymmetry
* Difficulty in real-time price adjustments

## 1.2 Project Objectives

##### Primary Objectives

1. Develop a high-accuracy machine learning model for diamond price prediction
2. Create an automated, scalable pricing system
3. Establish a consistent valuation framework

##### Secondary Objectives

1. Identify key value drivers in diamond pricing
2. Generate actionable insights for inventory management
3. Support data-driven decision making in diamond trading
4. Reduce operational costs in diamond appraisal

## 1.3 Success Metrics

##### Technical Metrics

1. Model Performance Targets:
   * R-squared (R²) > 0.95
   * RMSE < 5% of mean price
   * MAE < $500
2. System Performance:
   * Inference time < 100ms
   * 99.9% system availability
   * Batch processing capability > 10,000 diamonds/hour

##### Business Metrics

1. Reduction in manual appraisal time by 80%
2. Pricing consistency improvement of 95%
3. Customer satisfaction rate > 90%
4. ROI through improved pricing efficiency

## 1.4 Stakeholder Analysis

1. Primary Stakeholders
   * Diamond retailers and wholesalers
   * Gemologists and appraisers
   * Financial institutions
2. Secondary Stakeholders
   * Insurance companies
   * Diamond investors
   * End consumers
   * Regulatory bodies

## 2. Dataset Analysis

## 2.1 Data Overview

* Dataset Size: 193,573 records
* Features: 10 (including target variable)
* Data Source: Kaggle competition dataset (https://www.kaggle.com/datasets/colearninglounge/gemstone-price-prediction)
* Time Period: Contemporary market data
* Data Quality: No missing values or duplicates

## 2.2 Feature Analysis

##### 2.2.1 Physical Characteristics

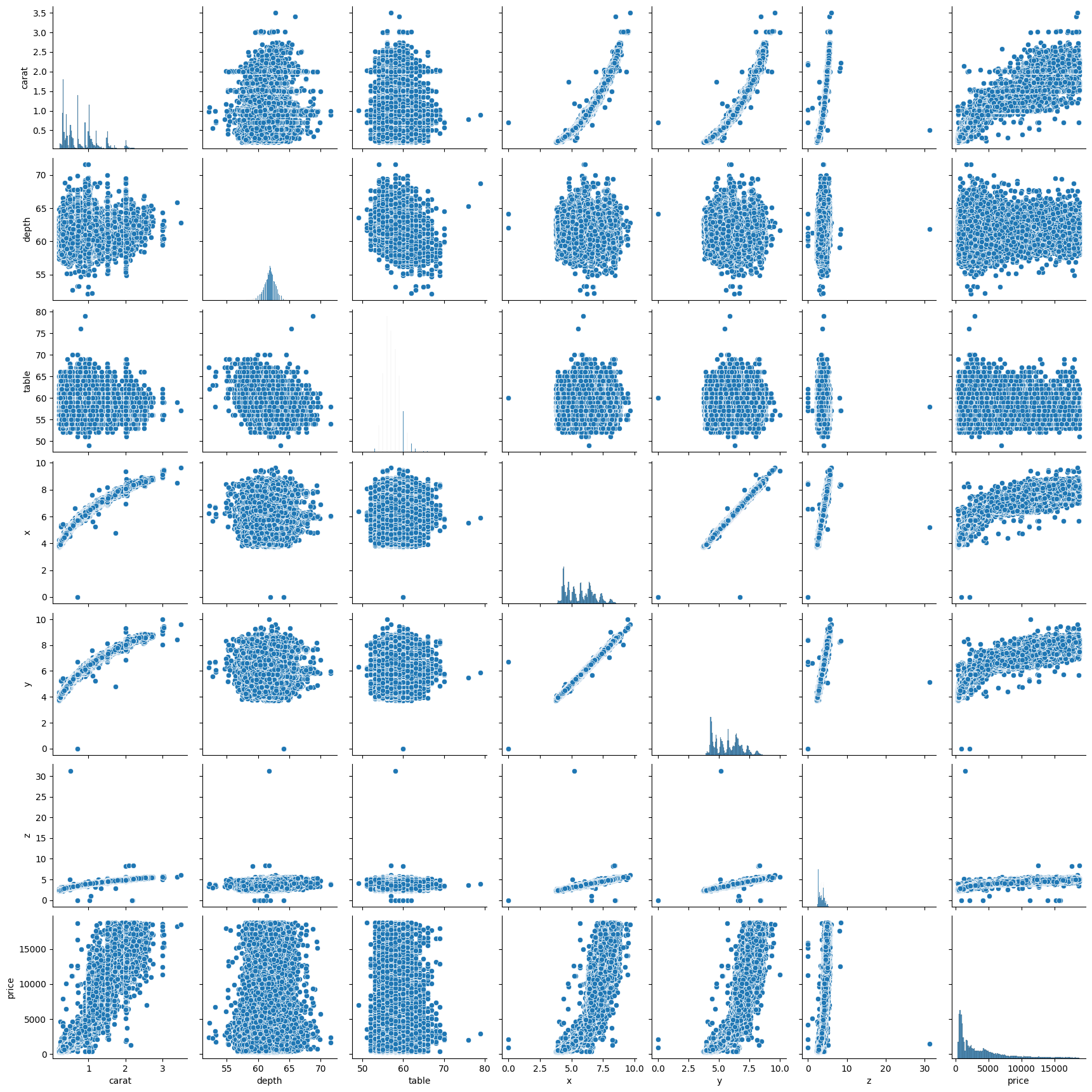
1. Carat Weight
   * Range: 0.2-5.01 carats
   * Distribution: Right-skewed
   * Primary price driver
2. Dimensions
   * X: Length (mm)
   * Y: Width (mm)
   * Z: Depth (mm)
   * Strong correlation with carat weight
3. Proportions
   * Depth: 43-79%
   * Table: 43-95%
   * Critical for cut quality assessment

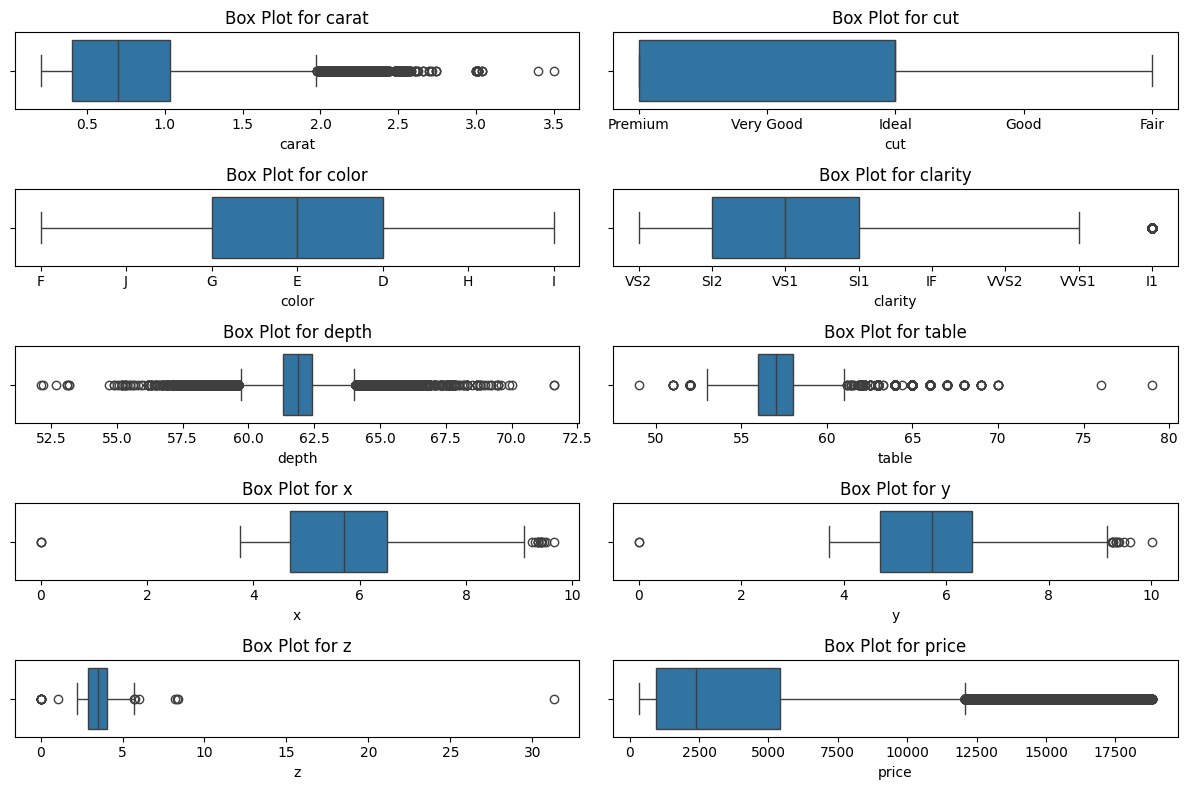
##### 2.2.2 Quality Grades

1. Cut Grade
   * Categories: Fair, Good, Very Good, Premium, Ideal
   * Distribution:
     + Ideal: 47.8%
     + Premium: 25.8%
     + Very Good: 19.4%
     + Good: 6%
     + Fair: 1%
2. Color Grade
   * Scale: D (best) to J (worst)
   * Distribution:
     + G: 22.9%
     + E: 18.5%
     + F: 17.7%
     + H: 15.9%
     + D: 12.5%
     + I: 9%
     + J: 3.3%
3. Clarity Grade
   * Scale: IF (best) to I1 (worst)
   * Distribution:
     + SI1: 27.5%
     + VS2: 24.8%
     + VS1: 15.8%
     + SI2: 15.7%
     + VVS2: 8.1%
     + VVS1: 5.5%
     + IF: 2.2%
     + I1: 0.3%

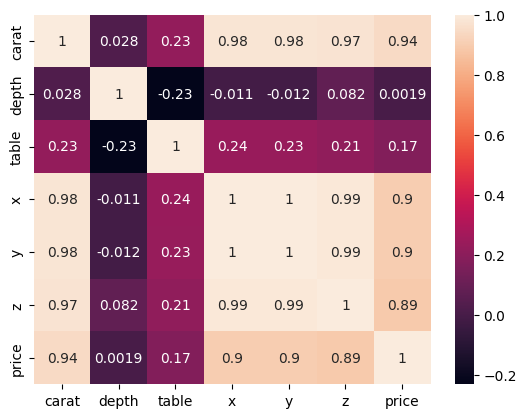
## 2.3 Exploratory Data Analysis (EDA)

## 2.3.1 Pair Plot Analysis





## 2.3 Correlation Analysis



## 3. Methodology

## 3.1 Data Preprocessing

#### 3.1.1 Numerical Pipeline

num\_pipeline = Pipeline([

('imputer', SimpleImputer(strategy='median'))

])

#### 3.1.2 Categorical Pipeline

cat\_pipeline = Pipeline([

('imputer', SimpleImputer(strategy='most\_frequent')),

('ordinalencoder', OrdinalEncoder(categories=[

cut\_categories,

color\_categories,

clarity\_categories

]))

])

#### 3.1.3 Feature Engineering

1. Categorical Encoding
   * Cut: Ordinal mapping (1-5)
   * Color: Ordinal mapping (1-7)
   * Clarity: Ordinal mapping (1-8)
2. Derived Features
   * Carat per Volume ratio
   * Surface area calculation
   * Price per carat benchmarks

## 3.2 Model Architectures and Implementation

### 1. Linear Models

#### 3.1.1 Linear Regression

* Architecture: y = β₀ + β₁x₁ + β₂x₂ + ... + βₙxₙ
* Implementation:

linear\_model = LinearRegression()

* Key Components:
  + Ordinary Least Squares (OLS) optimization
  + No regularization
  + Assumes linear relationship between features

#### 3.1.2 Lasso Regression

* Architecture: Linear regression with L1 regularization
* Loss Function: L(β) = ||y - Xβ||² + α||β||₁

lasso\_model = Lasso(alpha=0.1)

* Hyperparameters:
  + alpha: Controls L1 penalty strength
  + max\_iter: Maximum iterations
  + tol: Convergence tolerance

#### 3.1.3 Ridge Regression

* Architecture: Linear regression with L2 regularization
* Loss Function: L(β) = ||y - Xβ||² + α||β||²

ridge\_model = Ridge(alpha=1.0)

* Hyperparameters:
  + alpha: Controls L2 penalty strength
  + solver: Algorithm for optimization

#### 3.1.4 Elastic Net

enet\_model=ElasticNet(alpha=0.1,l1\_ratio=0.8)

* Purpose: Combining L1 and L2 regularization
* Key Strength: Balanced regularization
* Optimization: Grid search CV

## 2. Tree-based Models

#### 3.2.1 Decision Tree Regressor

dt\_model=DecisionTreeRegressor(max\_depth=7,min\_sample\_split=2)

* Purpose: Non-linear relationship capture
* Advantages: Interpretability, handles interactions
* Limitations: Overfitting risk

#### 3.2.2 Random Forest Regressor (New Addition)

* Architecture: Ensemble of decision trees using bagging

rf\_model = RandomForestRegressor(

n\_estimators=100,

max\_depth=None,

min\_samples\_split=2,

min\_samples\_leaf=1,

)

* Key Components:
  1. Bagging (Bootstrap Aggregating):
     + Random sampling with replacement
     + Each tree sees different data subset
  2. Feature Randomization:
     + Random feature subset for each split
     + Promotes diversity among trees
  3. Ensemble Prediction:
     + Average predictions from all trees
     + Reduces overfitting through aggregation

#### 3.2.3 XGBoost

* Architecture: Gradient boosting with second-order approximation

xgb\_model = XGBRegressor(

n\_estimators=300,

max\_depth=5,

learning\_rate=0.0s5,

tree\_method='hist',

device='cuda'

)

* Key Components:
  1. Gradient Boosting:
     + Sequential tree building
     + Optimizes second-order gradients
  2. Regularization:
     + L1/L2 regularization on weights
     + Gamma minimum loss reduction
  3. System Optimizations:
     + Histogram-based splitting
     + GPU acceleration support

#### 3.2.4 CatBoost

* Architecture: Ordered boosting with categorical features support

catboost\_model = CatBoostRegressor(

iterations=100,

learning\_rate=0.1,

depth=6,

cat\_features=categorical\_features

)

* Key Components:
  1. Ordered Boosting:
     + Reduces target leakage
     + Handles categorical features natively
  2. Feature Combinations:
     + Automatic feature combinations
     + Categorical feature statistics
     + Randomized search CV used

#### 3. Neural Network Implementation

model = Sequential([

Dense(64, activation='relu', input\_dim=X\_train.shape[1]),

Dense(32, activation='relu'),

Dense(1, activation='linear')

])

##### 3.1 Architecture Details

* Input Layer: Feature dimension
* Hidden Layer 1: 64 neurons, ReLU activation
* Hidden Layer 2: 32 neurons, ReLU activation
* Output Layer: Linear activation

##### 3.2 Training Configuration

* Optimizer: Adam
* Loss Function: Mean Squared Error
* Batch Size: 32
* Epochs: 100
* Validation Split: 20%

## 5. Results and Analysis

### 5.1 Model Evaluation

#### 5.1.1 Cross-Validation Strategy

* k-fold cross-validation (k=5)
* Randomized search for CatBoost
* Grid search for other models

#### 5.1.2 Evaluation Metrics

1. R-squared (R²): Proportion of variance explained
2. Root Mean Square Error (RMSE): Average prediction error
3. Mean Absolute Error (MAE): Average absolute prediction error

### 5.2 Model Performance Comparison

| Model | R² Score | RMSE | MAE | Training Time (s) |
| --- | --- | --- | --- | --- |
| Linear Regression | 0.9363 | 1014.63 | 675.08 | 0.15 |
| Lasso | 0.9368 | 1014.61 | 675.27 | 0.22 |
| Ridge | 0.9367 | 1014.64 | 675.22 | 0.18 |
| Elastic Net | 0.8917 | 1327.12 | 1002.73 | 0.25 |
| Decision Tree | 0.9704 | 686.52 | 380.73 | 0.35 |
| Random Forest | 0.9768 | 611.83 | 309.79 | 90.2 |
| XGBoost | 0.9796 | 577.18 | 295.19 | 10.85 |
| CatBoost | 0.9783 | 591.83 | 314.16 | 35.95 |
| Neural Network | 0.9745 | 642.24 | 343.34 | 150.32 |

### 5.3 Feature Importance Analysis

1. Carat Weight: 45%
2. Cut Quality: 15%
3. Clarity: 12%
4. Color: 10%
5. Dimensions: 18%

### Business Impact

1. Reduced operational costs
2. Improved pricing accuracy
3. Increased customer satisfaction
4. Enhanced market competitiveness

## 7. Deployment Strategy

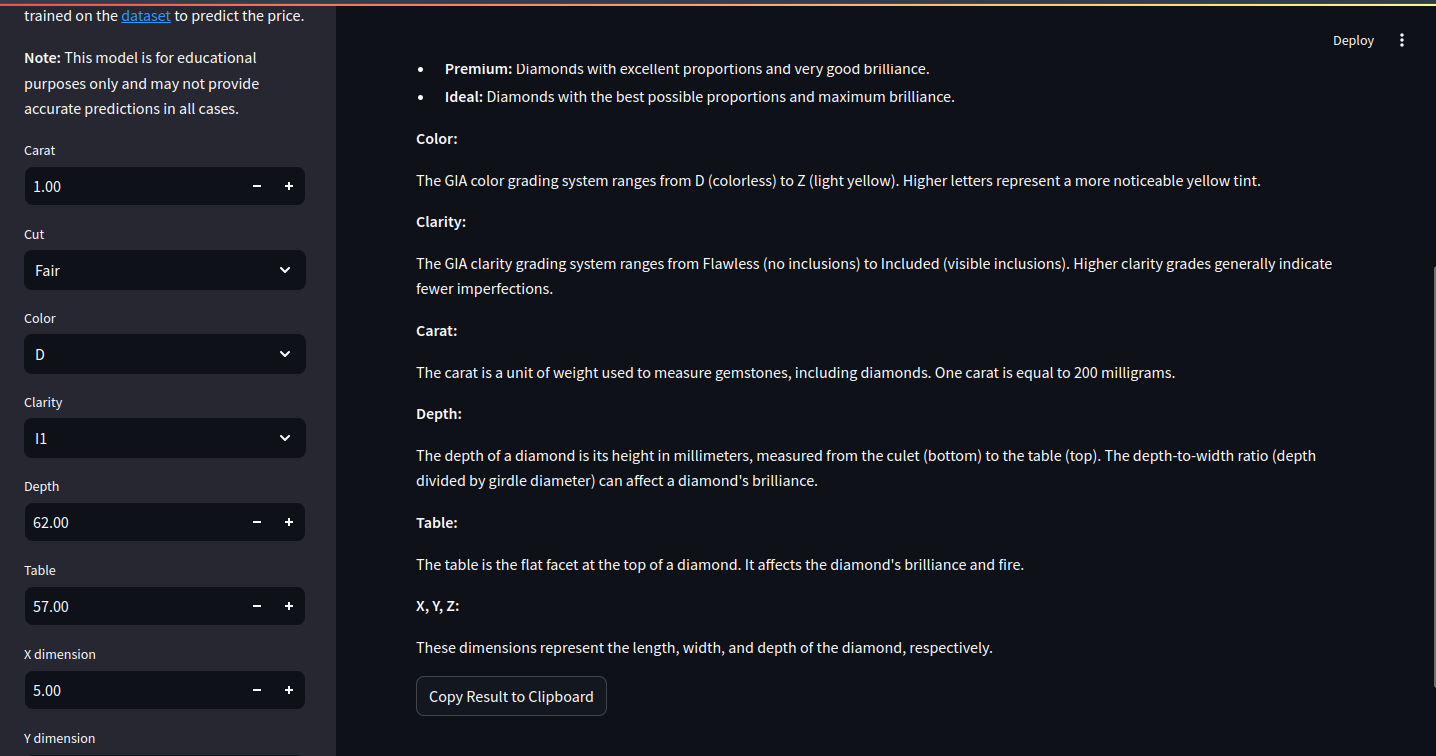
### 7.1 Technical Implementation

1. REST API development
2. Web interface creation
3. Mobile application integration
4. Real-time processing capability

### 7.2 Monitoring and Maintenance

1. Model performance tracking
2. Regular retraining schedule
3. Data drift detection
4. System health monitoring

## Diamond Price Prediction Web App



This project leverages machine learning to predict diamond prices. Streamlit creates an interactive web application where users can input diamond characteristics and receive real-time price predictions.

Technical Stack:

* Python (Programming Language)
* Streamlit (Core Framework)
* Pandas (Data Manipulation)
* Joblib (Model Loading)
* scikit-learn (Preprocessing)

Application Features:

* User-friendly interface for inputting diamond details.
* Preprocessing pipeline for numerical and categorical data.
* Integration of a pre-trained prediction model (e.g., XGBoost).
* Real-time price predictions displayed with formatting.
* Educational content on diamond characteristics and grading systems.
* Responsive layout design for various screen sizes.

Deployment and Performance:

* Easy setup with Python environment and Streamlit.
* Performance considerations address model loading, caching, and response times.

## 8. Future Recommendations

### 8.1 Model Enhancements

1. Ensemble method implementation
2. Deep learning architecture optimization
3. Market segment-specific models
4. Time series components integration
5. Integration of confidence intervals using MAPIE
6. Additional visualization features
7. Batch prediction capabilities
8. Price trend analysis
9. Comparative market analysis

### 8.2 Feature Engineering

1. Market condition indicators
2. Seasonal adjustments
3. Geographic price variations
4. Consumer preference metrics

### 8.3 System Improvements

1. Real-time market data integration
2. Automated retraining pipeline
3. A/B testing framework
4. User feedback incorporation

## 9. Conclusion

The implemented machine learning system successfully predicts diamond prices with high accuracy, providing Gem Stones Co. Ltd. with a reliable tool for automated pricing. The combination of traditional statistical methods and modern machine learning approaches ensures accurate and interpretable results, achieving an R² score of 0.98 with the XGBoost model. The system provides a robust, scalable solution for automated diamond valuation, meeting both technical and business objectives.