# Implementation Documentation

## Car Price Prediction System - Implementation Guide

---

## Table of Contents

1. [System Architecture](#system-architecture)
2. [Implementation Steps](#implementation-steps)
3. [Key Code Snippets](#key-code-snippets)
4. [Deployment](#deployment)

---

## � System Architecture

### \*\*Car Price Prediction System Workflow\*\*

┌─────────────────────────────────────────────────────────────────────────┐  
│ CAR PRICE PREDICTION SYSTEM │  
├─────────────────────────────────────────────────────────────────────────┤  
│ │  
│ TechStack ┌──────────────────┐ ┌─────────────────────────┐ │  
│ ├─ Streamlit │ Data Sources │ │ Machine Learning │ │  
│ ├─ Pandas │ │ │ │ │  
│ ├─ Scikit-learn │ • car\_dataset │────▶│ • Data Preprocessing │ │  
│ ├─ Plotly │ • launch\_years │ │ • Feature Engineering │ │  
│ ├─ Joblib │ • encoders.pkl │ │ • Model Training │ │  
│ └─ NumPy └──────────────────┘ │ • Gradient Boosting │ │  
│ └─────────────────────────┘ │  
│ │ │ │  
│ ▼ ▼ │  
│ ┌─────────────────┐ ┌─────────────────────┐ │  
│ │ Streamlit App │ │ Model Inference │ │  
│ │ │ │ │ │  
│ │ ├─ Home │◀────────│ • Price Prediction │ │  
│ │ ├─ Filtering │ │ • Input Validation │ │  
│ │ ├─ Analysis │ │ • Feature Encoding │ │  
│ │ ├─ Prediction │ │ • Result Formatting│ │  
│ │ └─ Comparison │ └─────────────────────┘ │  
│ └─────────────────┘ │ │  
│ │ │ │  
│ ▼ ▼ │  
│ ┌─────────────────────────┐ ┌─────────────────────────┐ │  
│ │ User Interface │ │ Data Storage │ │  
│ │ │ │ │ │  
│ │ • Interactive Forms │ │ • Model Files (.pkl) │ │  
│ │ • Real-time Charts │ │ • Dataset (CSV) │ │  
│ │ • Glassmorphism UI │ │ • Launch Year Data │ │  
│ │ • Responsive Design │ │ • Label Encoders │ │  
│ └─────────────────────────┘ └─────────────────────────┘ │  
│ │  
└─────────────────────────────────────────────────────────────────────────┘

### \*\*Data Flow Pipeline\*\*

Raw Data → Data Cleaning → Feature Engineering → Model Training → Web App → Predictions  
 ↓ ↓ ↓ ↓ ↓ ↓  
• 4000+ cars • Remove • Car age • Gradient • Streamlit • Price estimation  
• CSV format • outliers • Brand encoding • Boosting • Multi-page• Confidence interval   
• Mixed types • Fill missing • Derived metrics • 87% R² • Interactive• Market insights

---

## Implementation Steps

### \*\*Step 1: Environment Setup\*\*

python -m venv myenv  
source myenv/bin/activate  
pip install -r requirements.txt

### \*\*Step 2: Data Processing Pipeline\*\*

def data\_cleaning\_pipeline(df):  
 # Remove duplicates and outliers  
 df = df.drop\_duplicates()  
 Q1, Q3 = df['selling\_price'].quantile([0.25, 0.75])  
 IQR = Q3 - Q1  
 df = df[(df['selling\_price'] >= Q1-1.5\*IQR) & (df['selling\_price'] <= Q3+1.5\*IQR)]  
   
 # Handle missing values  
 df['km\_driven'].fillna(df['km\_driven'].median(), inplace=True)  
 df['fuel\_type'].fillna(df['fuel\_type'].mode()[0], inplace=True)  
   
 return df

### \*\*Step 3: Feature Engineering\*\*

def feature\_engineering(df, current\_year=2024):  
 # Most important feature: car age  
 df['car\_age'] = current\_year - df['year']  
   
 # Performance metrics  
 df['power\_to\_engine\_ratio'] = df['max\_power'] / df['engine']  
 df['efficiency\_score'] = df['mileage'] / df['engine'] \* 1000  
   
 return df

### \*\*Step 4: Model Training\*\*

from sklearn.ensemble import GradientBoostingRegressor  
  
# Train the main model  
model = GradientBoostingRegressor(  
 n\_estimators=150,  
 learning\_rate=0.1,  
 max\_depth=6,  
 random\_state=42  
)  
  
model.fit(X\_train, y\_train)  
joblib.dump(model, 'GradientBoost\_model.pkl')

### \*\*Step 5: Streamlit Application\*\*

# Main.py - Application entry point  
import streamlit as st  
  
st.set\_page\_config(page\_title="Car Price Predictor", layout="wide")  
  
# Navigation  
menu = st.sidebar.radio("Dashboard", [  
 "🏠 Home", "🔍 Filtering", "📊 Analysis",   
 "💰 Prediction", "📉 Comparison"  
])  
  
# Route to pages  
if menu == "💰 Prediction":  
 Prediction.app()

---

## Key Code Snippets

### \*\*1. Core Prediction Engine\*\*

class CarPricePredictionEngine:  
 def \_\_init\_\_(self):  
 self.model = joblib.load("GradientBoost\_model.pkl")  
 self.encoders = joblib.load("label\_encoders.pkl")  
   
 def predict\_price(self, features):  
 # Validate inputs  
 errors = self.validate\_inputs(features)  
 if errors:  
 return None, errors  
   
 # Encode categorical features  
 encoded\_features = self.encode\_features(features)  
   
 # Create feature vector  
 feature\_vector = np.array([  
 encoded\_features['brand'], encoded\_features['model'],  
 encoded\_features['car\_age'], encoded\_features['km\_driven'],  
 encoded\_features['engine'], encoded\_features['max\_power'],  
 encoded\_features['mileage'], encoded\_features['fuel\_type'],  
 encoded\_features['transmission'], encoded\_features['seats']  
 ]).reshape(1, -1)  
   
 # Make prediction  
 prediction = self.model.predict(feature\_vector)[0]  
 final\_price = max(0.5, prediction)  
   
 return {'predicted\_price': final\_price}, None

### \*\*2. Interactive UI Components\*\*

# Prediction form  
def create\_prediction\_form():  
 col1, col2 = st.columns(2)  
   
 with col1:  
 brand = st.selectbox("🚘 Brand", brands)  
 model = st.selectbox("🚗 Model", models)  
 year = st.selectbox("📅 Year", range(2024, 1979, -1))  
 km\_driven = st.number\_input("🛣️ KM Driven", 0, 500000, 25000)  
   
 with col2:  
 fuel\_type = st.selectbox("⛽ Fuel Type", fuel\_types)  
 transmission = st.selectbox("🔄 Transmission", transmissions)  
 engine = st.number\_input("⚙️ Engine (CC)", 500, 5000, 1200)  
 max\_power = st.number\_input("🔋 Power (bhp)", 30.0, 1000.0, 85.0)  
   
 return {  
 'brand': brand, 'model': model, 'year': year,  
 'km\_driven': km\_driven, 'fuel\_type': fuel\_type,  
 'transmission': transmission, 'engine': engine,  
 'max\_power': max\_power  
 }

### \*\*3. Data Visualization\*\*

import plotly.express as px  
  
def create\_price\_analysis():  
 # Brand distribution  
 fig = px.pie(df['brand'].value\_counts(),   
 title="Brand Distribution")  
 st.plotly\_chart(fig)  
   
 # Age vs Price correlation  
 fig = px.scatter(df, x='car\_age', y='selling\_price',  
 color='fuel\_type', trendline="ols")  
 st.plotly\_chart(fig)

### \*\*4. Model Performance Monitoring\*\*

def evaluate\_model\_performance(model, X\_test, y\_test):  
 predictions = model.predict(X\_test)  
   
 metrics = {  
 'r2\_score': r2\_score(y\_test, predictions),  
 'mae': mean\_absolute\_error(y\_test, predictions),  
 'rmse': np.sqrt(mean\_squared\_error(y\_test, predictions))  
 }  
   
 return metrics  
  
# Results: R² = 0.87, MAE = 1.89 lakhs, RMSE = 2.67 lakhs

---

## Deployment

### \*\*Local Development\*\*

# Run the application  
streamlit run Main.py

### \*\*Production Deployment\*\*

# Dockerfile  
FROM python:3.9-slim  
WORKDIR /app  
COPY requirements.txt .  
RUN pip install -r requirements.txt  
COPY . .  
EXPOSE 8501  
CMD ["streamlit", "run", "Main.py"]

### \*\*Performance Metrics\*\*

* \*\*Model Accuracy\*\*: 87% R² score
* \*\*Response Time\*\*: <2 seconds
* \*\*Data Processing\*\*: 4,000+ car records
* \*\*Concurrent Users\*\*: 50+ supported

---

Implementation Guide - Car Price Prediction System

Author: Argus-66 | Date: September 30, 2025