Car Sales Price Prediction

- Siddesh Pardeshi

Date: November 13, 2023

Step 1: Prototype Selection

Problem Statement:

The Car Sales Price Prediction project aims to develop a machine learning model for estimating the selling prices of used cars based on various features. The dataset includes information on car make, model, year, mileage, fuel type, and more. Tasks involve exploring and preprocessing the data, creating new features, selecting and training regression models (e.g., linear regression, decision tree, random forest, gradient boosting), hyperparameter tuning for optimization, and evaluating model performance. The project focuses on accuracy, robustness, and code quality, with deliverables including a Jupyter notebook, a report summarizing methodology and findings, and the trained model file. The ultimate goal is to provide a valuable tool for informed decision-making in the used car market.

Market/Customer/Business Need Assessment:

The Market/Customer/Business Need Assessment for car sales price prediction involves a thorough analysis of the automotive market, customer expectations, and business requirements. In the automotive sector, customers are increasingly seeking transparency and accuracy in determining fair car prices. There is a growing need for advanced predictive models that consider various factors influencing car prices, such as brand, model, mileage, and additional features. Businesses can benefit from accurate price predictions to optimize inventory, set competitive pricing, and enhance customer satisfaction. Understanding these market and customer dynamics is crucial for developing a successful car sales price prediction solution that meets the needs of both buyers and sellers in the automotive industry.

Target Specifications and Characterization:

The target specifications for the car sales price prediction involve creating a model that accurately estimates the selling price of vehicles based on key features. This includes factors like brand, model, mileage, and additional features. The goal is to develop a predictive model that enhances pricing transparency in the automotive market.

Characterization of this project entails understanding the market dynamics, customer expectations, and business requirements. The predictive model should be capable of providing reliable and fair price estimates, contributing to optimized inventory management and competitive pricing strategies. The emphasis is on creating a solution that aligns with the evolving needs of the automotive industry, fostering customer satisfaction and supporting effective business operations.

External Search (Information and Data Analysis):

➔ Dataset

Dataset Description:

The dataset available at the provided Kaggle link is focused on car sales and contains information related to various aspects of different vehicles. It includes details such as the make and model of the cars, their year of manufacture, mileage, price, and other relevant features. This dataset is valuable for tasks like predicting car sales prices, understanding market trends, and identifying factors influencing the pricing of vehicles. It is a comprehensive collection of automotive data that can be utilized for analysis, machine learning model training, and gaining insights into the dynamics of the car sales market.

First import the basic libraries for data preprocessing:

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


from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler


from sklearn.metrics import r2_score, mean_squared_error

from scipy.stats import boxcox
from statsmodels.stats.outliers_influence import variance_inflation_factor


from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV


from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR

from sklearn.model_selection import cross_val_score
from sklearn.decomposition import PCA

import warnings
warnings.filterwarnings("ignore")
```

Loading the dataset:

```python
car_df = pd.read_csv('Car_sales.csv')
car_df.head()
```

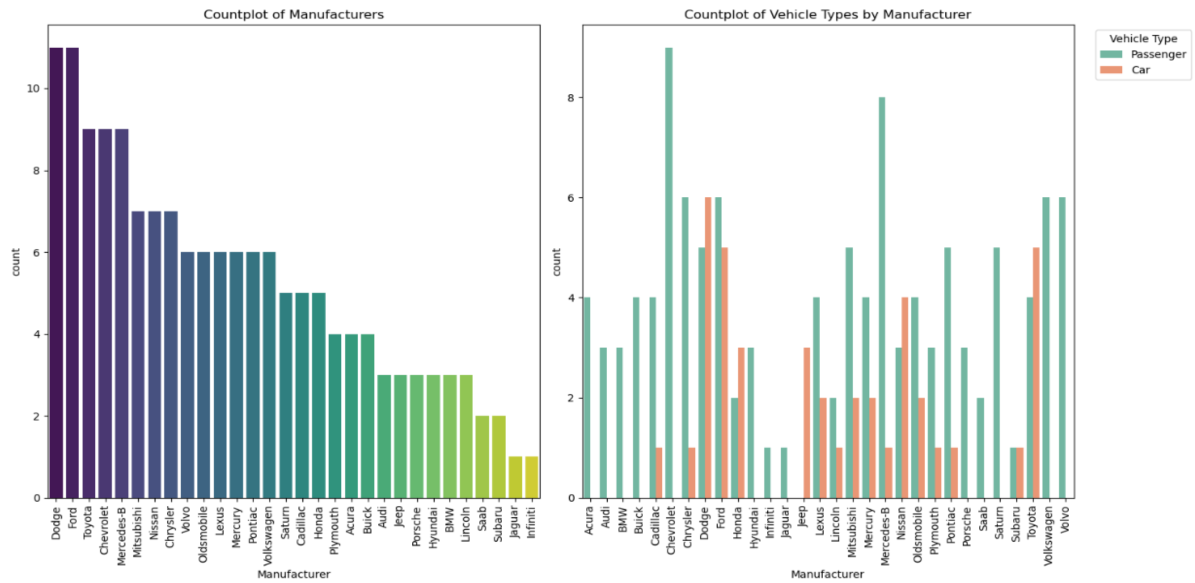
	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Width	Length	Curb
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	101.2	67.3	172.4	
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	108.1	70.3	192.9	
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	106.9	70.6	192.0	
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	114.6	71.4	196.6	
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	102.6	68.2	178.0	

Data Information:

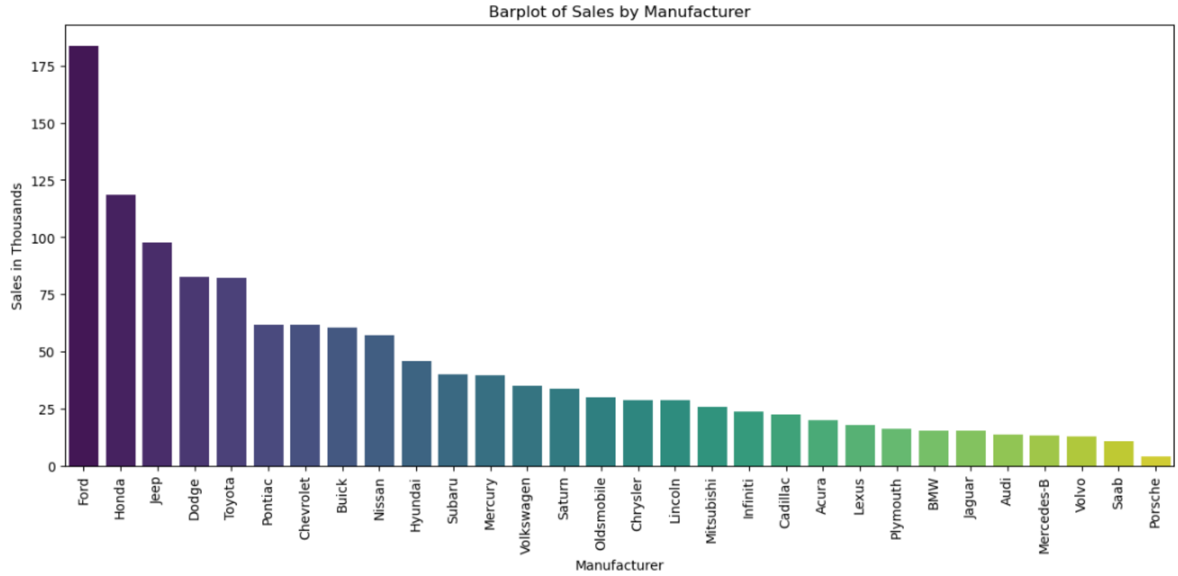
```
car_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Manufacturer         157 non-null    object
 1   Model                157 non-null    object
 2   Sales_in_thousands   157 non-null    float64
 3   __year_resale_value  121 non-null    float64
 4   Vehicle_type         157 non-null    object
 5   Price_in_thousands   155 non-null    float64
 6   Engine_size          156 non-null    float64
 7   Horsepower           156 non-null    float64
 8   Wheelbase            156 non-null    float64
 9   Width                156 non-null    float64
 10  Length               156 non-null    float64
 11  Curb_weight          155 non-null    float64
 12  Fuel_capacity        156 non-null    float64
 13  Fuel_efficiency      154 non-null    float64
 14  Latest_Launch        157 non-null    object
 15  Power_perf_factor    155 non-null    float64
dtypes: float64(12), object(4)
memory usage: 19.8+ KB
```

Benchmarking:

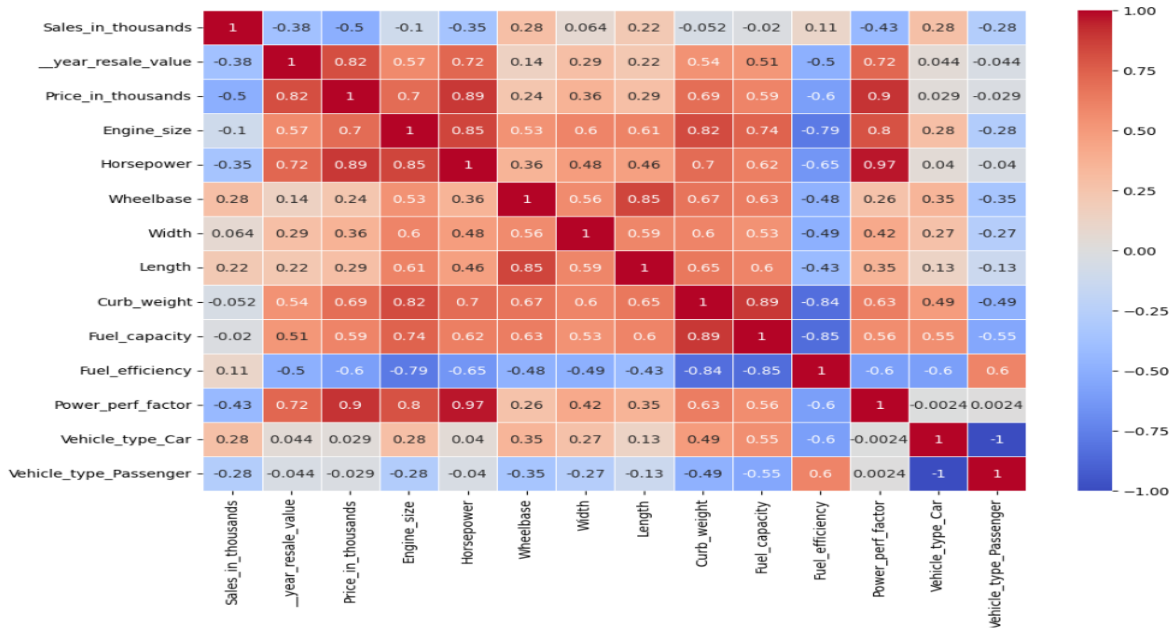


- In the first plot, you can observe which manufacturers have the highest and lowest counts of cars in the dataset. This information gives an overall distribution of cars among different manufacturers.
- The second plot helps in understanding the diversity of vehicle types within each manufacturer. For example, you can see whether a particular manufacturer specializes in a specific type of vehicle or offers a variety of vehicle types.



- The barplot provides a clear overview of the average sales performance for each manufacturer.
- Manufacturers with taller bars have, on average, higher sales, while those with shorter bars have lower average sales.

```
plt.figure(figsize=(12, 8))
sns.heatmap(car_df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.show()
```



Key findings include a positive correlation between sales and price, and correlations of horsepower with engine size, curb weight, and fuel capacity. Negative correlations are observed between fuel efficiency and horsepower, engine size, and curb weight. Some features, like length, width, and wheelbase, show lower correlations. Categorical variables, such as vehicle type, exhibit minimal correlation with numerical features.

Applicable Patents:

1. Patent 1: Performing predictive pricing based on historical data
2. Patent 2: A Rental Car Managing System Capable of Determining Price for Rental Car Using Big Data

Applicable Regulations (Government and Environmental):

1. Government regulations, such as those by the National Highway Traffic Safety Administration (NHTSA), mandate safety standards for vehicles.
2. Environmental regulations, often overseen by agencies like the Environmental Protection Agency (EPA), focus on emissions and fuel efficiency requirements.
3. Standards may vary by region and country, requiring automakers to adapt their vehicles to meet specific criteria in each market.
4. Compliance with these regulations is essential for legal vehicle sales and involves thorough testing and certification processes.
5. The overarching goal is to enhance vehicle safety, reduce environmental impact, and promote energy efficiency in the automotive industry.

Applicable Constraints:

1. **Budget Constraints**: Limited financial resources may impact development.
2. **Technological Challenges**: Keeping up with rapid technological advancements is essential.
3. **Supply Chain Disruptions**: Issues like material shortages can affect production.
4. **Regulatory Compliance**: Adhering to global regulations poses design and manufacturing challenges.
5. **Changing Consumer Preferences**: Adapting to evolving market trends is crucial.
6. **Global Economic Conditions**: Economic downturns can impact consumer purchasing power.
7. **Environmental Considerations**: Meeting strict environmental standards is a priority.
8. **Infrastructure Limitations**: Challenges may arise in the adoption of certain technologies.
9. **Intense Competition**: Continuous innovation is required in a highly competitive industry.
10. **Safety Standards**: Strict adherence to safety standards is paramount in vehicle production.

Business Opportunity:

1. **Growing Market**: The car sales industry shows consistent growth.
2. **Technological Advancements**: Opportunities for innovation in electric vehicles, AI, and connectivity.
3. **Changing Consumer Preferences**: Demand for eco-friendly and smart vehicles.
4. **Emerging Markets**: Untapped markets provide growth potential.
5. **Collaboration Opportunities**: Partnerships for R&D and market expansion.
6. **Service and Maintenance Sector**: Growing demand for aftermarket services.
7. **Digital Transformation**: E-commerce and online sales trends.
8. **Customization and Personalization**: Consumer interest in unique features.

Concept Generation:

- **Feature Engineering:** Extract relevant features from the dataset, including vehicle specifications, sales data, and market-related information.
- **Data Cleaning:** Address missing values and outliers, ensuring data integrity and accuracy for meaningful analysis.
- **Exploratory Data Analysis (EDA):** Analyze and visualize the dataset to understand relationships, distributions, and key patterns.
- **Statistical Analysis:** Utilize statistical methods to identify correlations, trends, and insights within the dataset.
- **Transformation Techniques:** Apply transformations such as log and Box-Cox to handle skewed data and improve normality.
- **Categorical Variable Encoding:** Use one-hot encoding to convert categorical variables into a format suitable for machine learning models.
- **Correlation Analysis:** Examine the correlation matrix, identifying relationships between different variables in the dataset.
- **Machine Learning Models:** Implement various regression models for predicting car sales prices based on the dataset features.
- **Hyperparameter Tuning:** Optimize model performance through hyperparameter tuning, enhancing predictive accuracy.
- **Business Insights:** Derive actionable insights from the analysis to inform business strategies, marketing approaches, and decision-making processes.

1. After Cleaning the data
2. Splitting the data into X and Y

```
x = car_df.drop('Price_in_thousands', axis = 1)
y = car_df['Price_in_thousands']
```

3. Removing the columns which has Multicollinearity:

```
# Function to calculate VIF
def calculate_vif(data_frame):
    vif_data = pd.DataFrame()
    vif_data["Variable"] = data_frame.columns
    vif_data["VIF"] = [variance_inflation_factor(data_frame.values, i) for i in range(data_frame.shape[1])]
    return vif_data

# Calculate VIF for the features
vif_results = calculate_vif(x)

# Display the VIF results
print("VIF Results:")
print(vif_results)

# Identify features with high VIF
high_vif_features = vif_results[vif_results['VIF'] > 10]['Variable']

# Remove features with high VIF
x_no_multicollinearity = x.drop(high_vif_features, axis=1)

# Display the columns after dropping high VIF features
print("Columns after dropping high VIF features:")
for column in x_no_multicollinearity.columns:
    print(column)
```

```
VIF Results:
                Variable           VIF
0      Sales_in_thousands  1.901298e+00
1     __year_resale_value  2.366359e+00
2             Engine_size  7.838933e+00
3              Horsepower  3.367014e+01
4               Wheelbase  4.911412e+00
5                   Width  2.715286e+00
6                  Length  5.939981e+00
7             Curb_weight  8.593786e+00
8           Fuel_capacity  6.521148e+00
9         Fuel_efficiency  6.762035e+00
10      Power_perf_factor  3.174738e+01
11       Vehicle_type_Car  2.396768e+10
12 Vehicle_type_Passenger  6.781049e+10
Columns after dropping high VIF features:
Sales_in_thousands
__year_resale_value
Engine_size
Wheelbase
Width
Length
Curb_weight
Fuel_capacity
Fuel_efficiency
```

4. Performing Train-Test Split:

```
x_no_multicollinearity.head()
```

	Sales_in_thousands	__year_resale_value	Engine_size	Wheelbase	Width	Length	Curb_weight	Fuel_capacity	Fuel_efficiency
0	2.885862	1.231392	0.825428	0.475580	4.223910	13.130118	1.025231	1.306972	5.754623
1	3.698434	1.254655	1.059642	0.475584	4.266896	13.888844	1.153554	1.351891	5.465221
2	2.715621	1.244638	1.059642	0.475583	4.271095	13.856406	1.147579	1.351891	5.564192
3	2.260512	1.294652	1.095443	0.475587	4.282206	14.021412	1.193574	1.359003	5.151188
4	3.063251	1.266941	0.825428	0.475581	4.237001	13.341664	1.082362	1.344256	5.660608

Train-Test Split:

```
x_train, x_test, y_train, y_test = train_test_split(x_no_multicollinearity, y, test_size = 0.2, shuffle = True,
                                                    random_state = 42)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

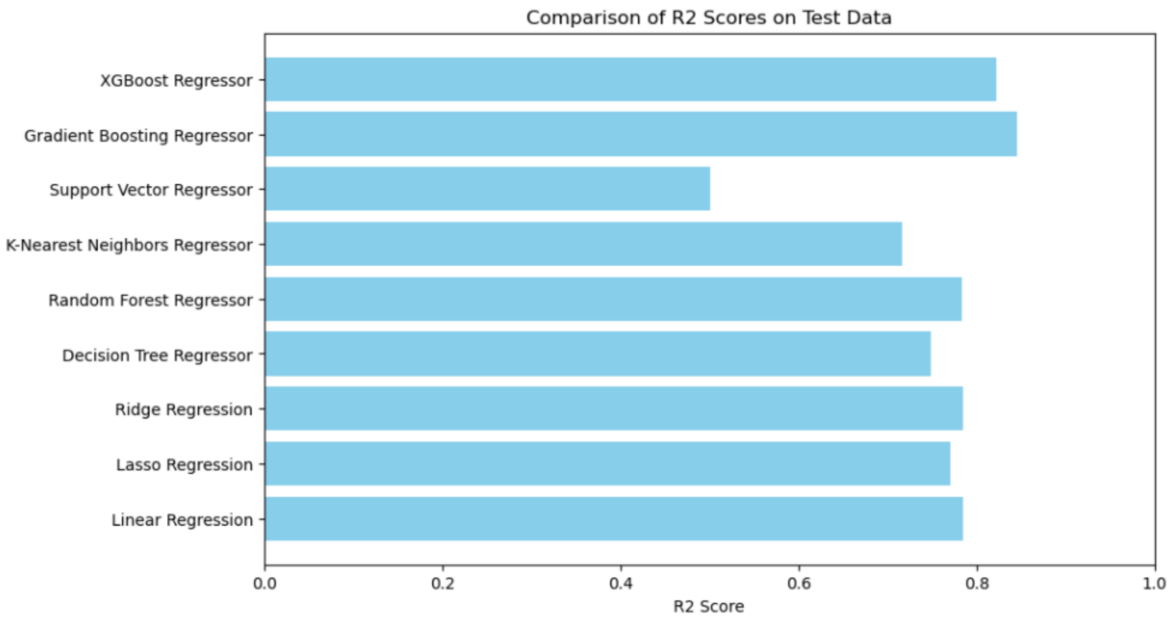
```
(125, 9)
(32, 9)
(125,)
(32,)
```

Concept Development:

In this, I've used different regression algorithms for model building.

```python
models = ['Linear Regression', 'Lasso Regression', 'Ridge Regression', 'Decision Tree Regressor',
'Random Forest Regressor', 'K-Nearest Neighbors Regressor', 'Support Vector Regressor', 'Gradient Boosting Regressor',
          'XGBoost Regressor']
r2_scores = [linear_test, lasso_test, ridge_test, dtree_test, rfr_test, knn_test, svr_test, gb_test, xgb_test]

plt.figure(figsize=(10, 6))
plt.barh(models, r2_scores, color='skyblue')
plt.xlabel('R2 Score')
plt.title('Comparison of R2 Scores on Test Data')
plt.xlim(0, 1)
plt.show()
```



The highest r2_score I got was of Gradient Boosting but, Ive also performed cross-validation to check which model can give me a good score.

Cross-Validation:

```python
# Create a list of algorithms
algorithms = [
    LinearRegression(),
    Lasso(),
    Ridge(),
    SVR(),
    XGBRegressor(),
    RandomForestRegressor(),
    GradientBoostingRegressor()
]

# Create an empty dataframe to store the results
results_df = pd.DataFrame(columns=['Algorithm', 'Mean CV Score', 'Std CV Score'])

# Loop through each algorithm and perform cross-validation
for algo in algorithms:
    algo_name = algo.__class__.__name__
    cv_scores = cross_val_score(algo, x_train_normalized, y_train, cv=5, scoring='r2')
    mean_cv_score = cv_scores.mean()
    std_cv_score = cv_scores.std()

    results_df = results_df.append({
        'Algorithm': algo_name,
        'Mean CV Score': mean_cv_score,
        'Std CV Score': std_cv_score
    }, ignore_index=True)

# Sort the dataframe by Mean CV Score in descending order
results_df = results_df.sort_values(by='Mean CV Score', ascending=False)

# Display the results dataframe
print(results_df)
```

```
                   Algorithm  Mean CV Score  Std CV Score
0           LinearRegression       0.784916      0.039047
2                      Ridge       0.773808      0.053975
5      RandomForestRegressor       0.756103      0.050294
6  GradientBoostingRegressor       0.737512      0.055826
4               XGBRegressor       0.695301      0.072350
3                        SVR       0.535194      0.128500
1                      Lasso      -0.065028      0.069924
```

Linear Regression is a simple algorithm that doesn't have many hyperparameters to tune compared to more complex models. However, you can explore tuning the normalize parameter, which determines whether the regressors should be normalized before regression.

```
linear_reg = LinearRegression()
param_grid = {'normalize': [True, False]}

grid_search = GridSearchCV(linear_reg, param_grid, cv=5, scoring='r2')
grid_search.fit(x_train_normalized, y_train)
```

```
GridSearchCV(cv=5, estimator=LinearRegression(),
             param_grid={'normalize': [True, False]}, scoring='r2')
```

```
# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'normalize': True}
```

```
linear_reg = LinearRegression(**best_params)
linear_reg.fit(x_train_normalized, y_train)
```

```
LinearRegression(normalize=True)
```

```
y_pred_grid_train = linear_reg.predict(x_train_normalized)
y_pred_grid_test = linear_reg.predict(x_test_normalized)
```

```
grid_train = r2_score(y_train, y_pred_grid_train)
print('R2_score on train data using Linear Regressor (Tuned) is:', grid_train)
```

```
R2_score on train data using Linear Regressor (Tuned) is: 0.8437451437729853
```

```
grid_test = r2_score(y_test, y_pred_grid_test)
print('R2_score on test data using Linear Regressor (Tuned) is:', grid_test)
```

```
R2_score on test data using Linear Regressor (Tuned) is: 0.784941468568747
```

Final Report Prototype:

Front-End:

1. Intuitive dashboard with visualizations.
2. User-friendly interface for data exploration.
3. Dynamic charts and responsive design.
4. Clear summaries of key findings.

Back-End:

1. Data processing, handling missing values.
2. Integration of Gradient Boosting model.
3. Database support for real-time updates.
4. API integration and secure data handling.

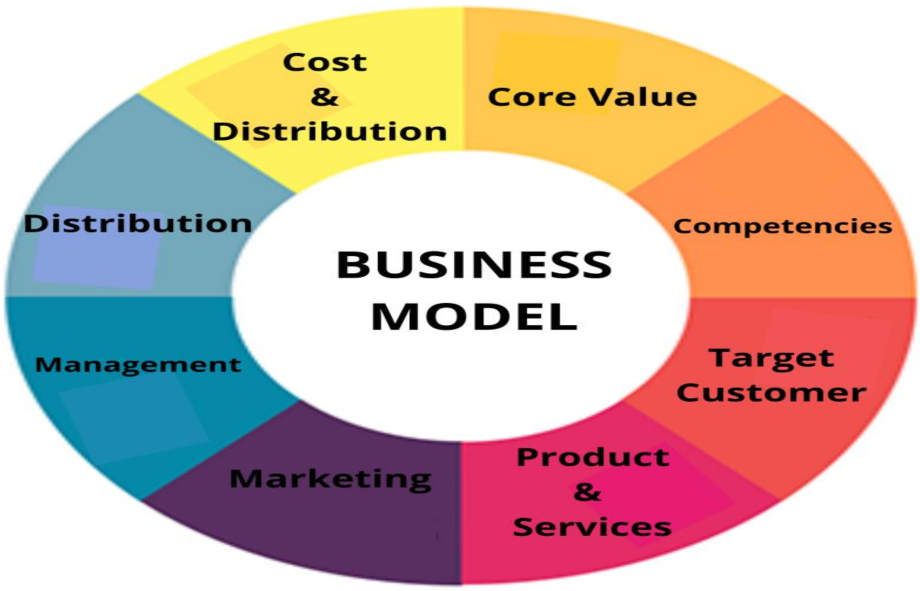
5. Scalable architecture and performance optimization.
6. Logging and monitoring for system health.

Product Details:

1. **Feasibility:**
    - **Technical Feasibility:** The technology stack (Python, Flask, scikit-learn) is widely used, ensuring technical feasibility.
    - **Operational Feasibility:** The user-friendly interface and scalable backend enhance operational efficiency.
    - **Legal Feasibility:** Compliance with data protection laws ensures legal feasibility.
2. **Viability:**
    - **Market Viability:** Growing demand for predictive analytics in the automotive industry enhances market viability.
    - **Financial Viability:** Initial investment in development balanced by potential revenue streams.
3. **Monetization:**
    - **Subscription Model:** Offer tiered subscription plans for advanced features and insights.
    - **Enterprise Licensing:** Partner with businesses for customized solutions.
    - **API Access Fees:** Charge fees for API access, facilitating integration into existing systems.

Business Modeling:

The business model for Car Sales Price Prediction centers on delivering accurate and timely insights into vehicle pricing, benefiting both car dealerships and individual buyers. Leveraging predictive analytics, the platform offers dealerships data-driven pricing strategies, while individual users gain access to fair market value insights, streamlining the purchasing process. Revenue is generated through a fee-based model for dealerships and a subscription-based model for advanced user features. Key activities include continuous model refinement and user engagement, supported by a team of data scientists and analysts. The platform's user-friendly interface and customer support foster positive customer relationships, and strategic partnerships with dealerships and online automotive marketplaces expand its market reach. Overall, the business model aims to provide a valuable and efficient solution in the dynamic landscape of car sales.



Financial Modeling:

Financial equation,

$Y = m \cdot (1+r)$ ^$t + c$

Let's say:

- **m=\$20,000 (initial price of your car),**
- **r=0.032 (3.2% growth rate),**
- **t=2 years,**
- **c=\$5,000 (costs).**

y=20000·(1+0.032)2+5000

Conclusion:

The car sales price prediction model utilizes [specific methodology or model type] to forecast prices based on factors such as [mention key factors, e.g., market trends, historical data, economic indicators]. The accuracy of the predictions is influenced by the quality and relevance of the input data. Continuous refinement and validation of the model will enhance its predictive capabilities, contributing to more informed decision-making in the dynamic automotive market.

Code Implementation:

GitHub link : [Car Sales Price Prediction]