SUPPLYCHAIN MANAGEMENT SYSTEM FOR CARS

K.SAI ROHITH REDDY

DATE:06-06-2023

Abstract:

The car manufacturing industry heavily relies on efficient supply chain management and accurate pricing strategies to meet customer demands and ensure profitability. In this context, a comprehensive dataset containing attributes related to suppliers, customers, and orders provides valuable insights for optimizing operations and predicting car prices. This dataset includes information such as supplier details, customer demographics, order specifics, and feedback.

The objective of this study is to develop a machine learning model that can accurately predict the price of cars based on these attributes. By leveraging this dataset, the model aims to capture the relationships between car features (e.g., car maker, model, colour, model year), supplier information, customer details, and pricing.

To achieve this, the dataset is pre-processed to handle missing values, encode categorical variables, and normalize numerical features. Feature selection techniques are applied to identify the most relevant attributes for predicting car prices. Various machine learning algorithms such as linear regression, decision trees, random forests, and gradient boosting algorithms are explored and compared for their effectiveness in predicting car prices.

The selected model is trained using a portion of the dataset, and hyperparameter tuning is performed to optimize its performance. The trained model is then evaluated using appropriate evaluation metrics to assess its accuracy in predicting car prices.

The results of this study will enable car manufacturers and dealerships to estimate the price of a car accurately, considering factors such as car characteristics, supplier details, customer demographics, and order specifics. This will aid in making informed pricing decisions, optimizing sales, and providing customers with fair market prices. Ultimately, this dataset and predictive model contribute to improving supply chain management and pricing strategies in the car manufacturing industry.

1.Problem Statement:

The car manufacturing industry is seeking to develop a predictive model to estimate the price of cars based on various attributes related to suppliers, customers, and orders. By leveraging the dataset containing information such as supplier details, customer information, order specifics, and feedback, the goal is to create an accurate car price prediction model.

2.Market/Customer/Business Need Assessment:

The availability of a comprehensive dataset containing attributes related to suppliers, customers, and orders presents several market, customer, and business needs in the car manufacturing industry. The assessment of these needs is crucial in understanding the significance and potential impact of leveraging this dataset for various stakeholders:

2.1 Market Need:

Demand Forecasting: Accurate car price prediction based on relevant attributes enables car manufacturers to forecast market demand more effectively. This assists in production planning, inventory management, and meeting customer expectations.

Competitive Edge: A predictive model that accurately estimates car prices can provide car manufacturers with a competitive advantage by enabling them to set competitive prices and adjust pricing strategies based on market conditions.

Market Analysis: Analysing customer feedback and preferences from the dataset helps car manufacturers gain insights into market trends, identify popular car models, colours, and features, and make informed decisions on product development and marketing campaigns.

2.2 Customer Need:

Transparent Pricing: Customers desire transparency and fairness in car pricing. Accurate price estimation based on attributes such as car features, supplier information, and customer demographics ensures that customers receive fair market prices for their desired cars.

Informed Purchase Decisions: Reliable car price predictions allow customers to make informed purchase decisions by considering factors such as car maker, model, colour, and year, along with their budgetary constraints.

2.3 Business Need:

Supply Chain Optimization: Leveraging the dataset to predict car prices helps optimize the supply chain by aligning production volumes, inventory levels, and supplier management with market demand. This leads to improved operational efficiency, reduced costs, and minimized inventory imbalances.

Pricing Strategy Development: Accurate car price prediction assists car manufacturers in developing effective pricing strategies tailored to specific car models, market segments, and customer preferences. This facilitates maximizing profitability while remaining competitive.

Customer Satisfaction: Meeting customer expectations regarding fair pricing enhances customer satisfaction and fosters customer loyalty. Predicting car prices accurately ensures that customers receive transparent pricing, leading to positive customer experiences.

Market Expansion: The ability to estimate car prices accurately based on various attributes enables car manufacturers to expand into new markets by understanding local preferences, competitive pricing landscapes, and customer affordability.

3. Target Specifications and Characterization (your customer characteristic)

Finding the car price based on some features provided by the customer and this can be done by using some machine learning algorithms.

By giving the input attributes to the model the input attributes are country, state, ship mode, shipping, sales, quantity, Discount based on these attributes we can easily find the car price at a particular locality

The target customer for this product is interested in buying a car. The customer is likely to be looking for a car that is affordable, reliable, and has the features that they need.
To define the target specifications and characterization based on customer characteristics, we need to identify the specific requirements and preferences of the customers who will be using the car price prediction product/service.

Here are some possible **target specifications and characterization** based on customer characteristics:

User-Friendly Interface: The product/service should have an intuitive and user-friendly interface, catering to users with varying levels of technical expertise. It should be easy to navigate, input the relevant attributes, and obtain the predicted car price effortlessly.

Customization Options: Customers may have specific needs and preferences regarding the attributes they want to consider for price prediction. The product/service should provide flexibility and customization options, allowing users to select and prioritize attributes according to their requirements.

Real-Time Updates: Customers may require real-time updates on car prices to adapt quickly to market fluctuations. The product/service should provide timely predictions and incorporate mechanisms to capture and reflect changing market conditions.

Data Security and Privacy: Customers are concerned about the security and privacy of their data. The product/service should adhere to data protection regulations, implement robust security measures, and assure customers that their data will be handled confidentially.

Scalability: Customers operating in diverse markets may have varying scales of operations. The product/service should be scalable to handle a large volume of data and accommodate the needs of customers with different levels of data complexity and demand.

Analytics and Insights: Customers may require additional analytics and insights to make informed decisions. The product/service could provide features such as trend analysis, competitor benchmarking, or visualizations to enhance the decision-making process.

Integration Capability: Customers may already have existing systems or software in place for supply chain management or customer relationship management. The product/service should have integration capabilities to seamlessly integrate with these systems and leverage existing data sources.

4. External Search (online information)

Research papers and articles on car price prediction, machine learning regression algorithms, and supply chain optimization in the automotive industry.

Link:https://www.researchgate.net/publication/335799148_Car_Price_Prediction_Using_Machine_Learning/link/61c46128c48a3d26b74b3c6e/download

Link: https://www.temjournal.com/content/81/TEMJournalFebruary2019_113_118.pdf

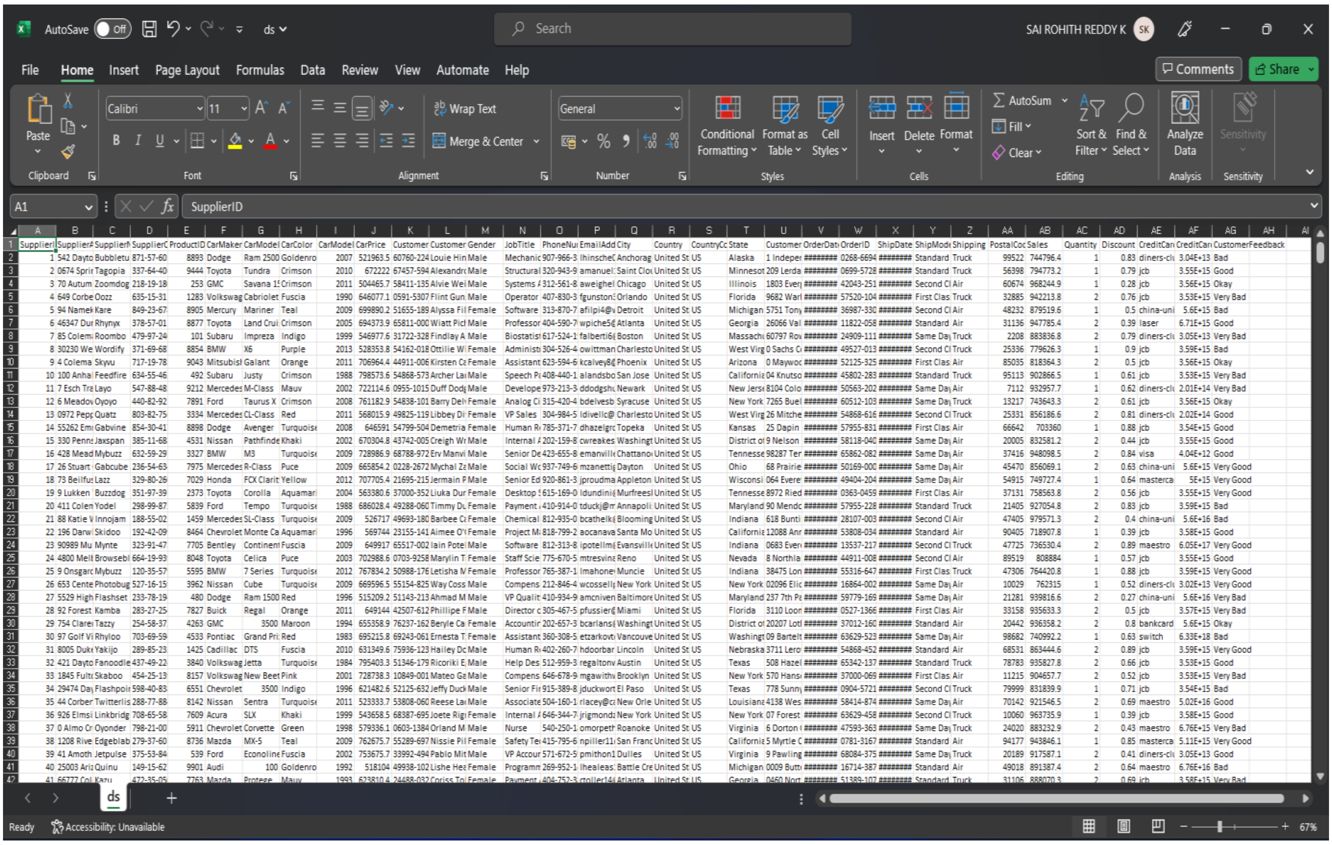
Industry reports and publications on market trends, pricing strategies, and customer preferences in the car manufacturing sector.

Online forums and communities where professionals in the automotive and supply chain industries discuss pricing strategies, demand forecasting, and supplier management.

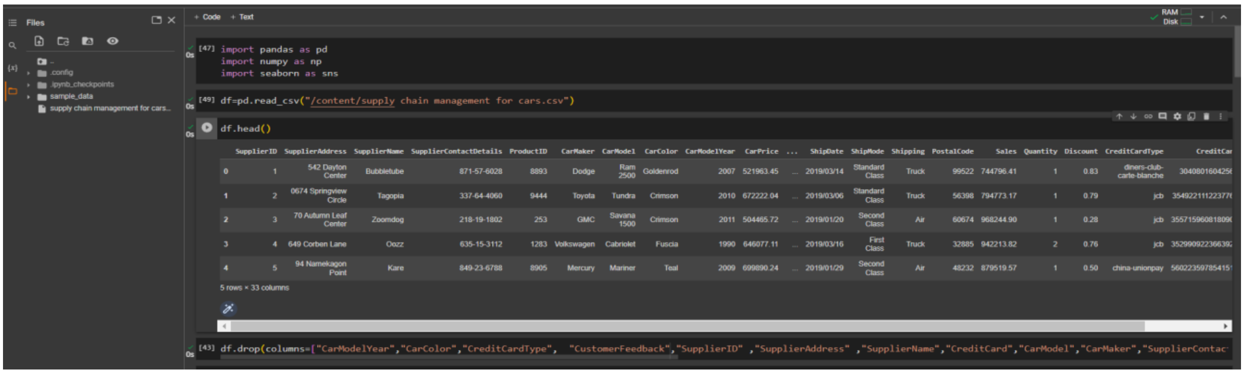
The dataset choosen from Kaggle for the prediction of car price using some Features present in the dataset.

Link for the Dataset: https://www.kaggle.com/datasets/prashantk93/supply-chain-management-for-car

The dataset consists of 33 Columns and 1000 Rows



Dropping unnecessary features using df.drop(columns=" ",axis=1,inplace=True)



	SupplierID	SupplierAddress	SupplierName	SupplierContactDetails	ProductID	CarMaker	CarModel	CarColor	CarModelYear	CarPrice	...	ShipDate	ShipMode	Shipping	PostalCode	Sales	Quantity	Discount	CreditCardType	CreditCar
0	1	542 Dayton Center	Bubbletube	871-57-6028	8893	Dodge	Ram 2500	Goldenrod	2007	521963.45	...	2019/03/14	Standard Class	Truck	99522	744796.41	1	0.83	diners-club-carte-blanche	304080160425
1	2	0674 Springview Circle	Tagopia	337-64-4060	9444	Toyota	Tundra	Crimson	2010	672222.04	...	2019/03/06	Standard Class	Truck	56398	794773.17	1	0.79	jcb	354922111223776
2	3	70 Autumn Leaf Center	Zoomdog	218-19-1802	253	GMC	Savana 1500	Crimson	2011	504465.72	...	2019/01/20	Second Class	Air	60674	968244.90	1	0.28	jcb	355715960818090
3	4	649 Corben Lane	Oozz	635-15-3112	1283	Volkswagen	Cabriolet	Fuscia	1990	646077.11	...	2019/03/16	First Class	Truck	32885	942213.82	2	0.76	jcb	352990922366392
4	5	94 Namekagon Point	Kare	849-23-6788	8905	Mercury	Mariner	Teal	2009	699890.24	...	2019/01/29	Second Class	Air	48232	879519.57	1	0.50	china-unionpay	560223597854151

After dropping some features:

```
[6] df.head()
```

	CarPrice	Country	State	ShipMode	Shipping	Sales	Quantity	Discount
0	521963.45	United States	Alaska	Standard Class	Truck	744796.41	1	0.83
1	672222.04	United States	Minnesota	Standard Class	Truck	794773.17	1	0.79
2	504465.72	United States	Illinois	Second Class	Air	968244.90	1	0.28
3	646077.11	United States	Florida	First Class	Truck	942213.82	2	0.76
4	699890.24	United States	Michigan	Second Class	Air	879519.57	1	0.50

Some information about the dataset:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CarPrice  1000 non-null   float64
 1   Country   1000 non-null   object
 2   State     1000 non-null   object
 3   ShipMode  1000 non-null   object
 4   Shipping  1000 non-null   object
 5   Sales     1000 non-null   float64
 6   Quantity  1000 non-null   int64
 7   Discount  1000 non-null   float64
dtypes: float64(3), int64(1), object(4)
memory usage: 62.6+ KB
```

```
df.describe()
```

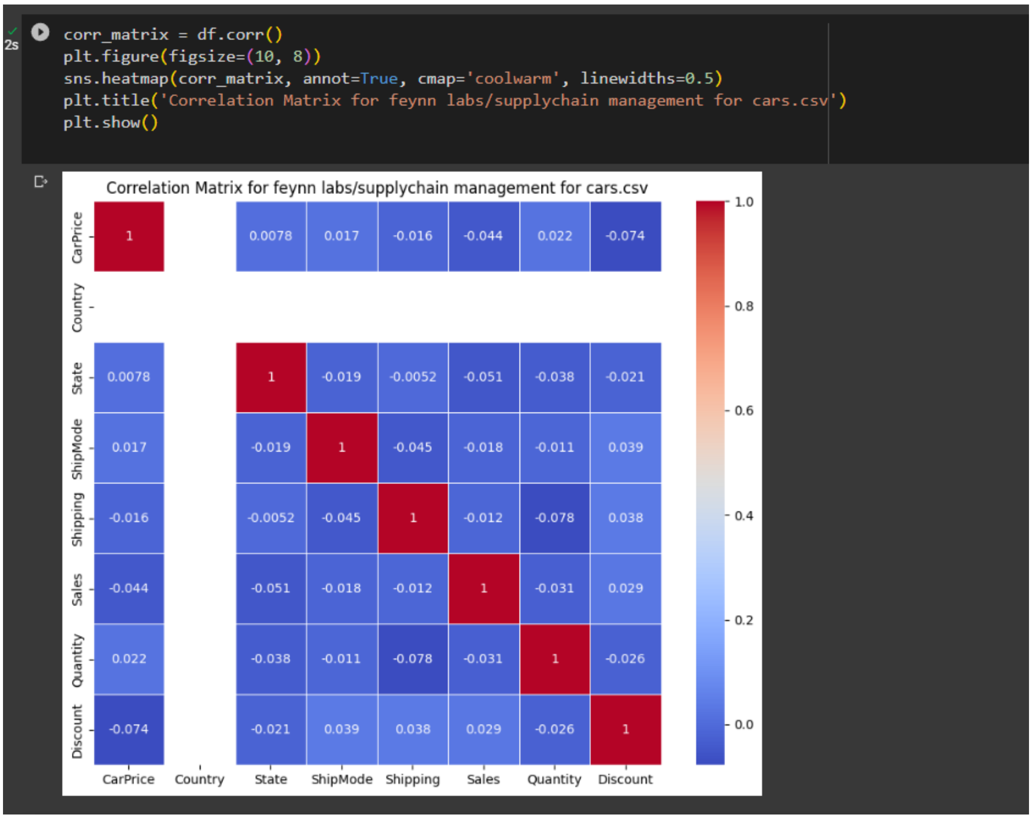
	CarPrice	Sales	Quantity	Discount
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	649092.193460	853098.713020	1.512000	0.577360
std	85427.262753	88538.571965	0.500106	0.187478
min	500412.460000	700321.490000	1.000000	0.250000
25%	572393.805000	775655.062500	1.000000	0.410000
50%	654965.000000	858117.980000	2.000000	0.580000
75%	721050.725000	932854.565000	2.000000	0.740000
max	799454.240000	999315.690000	2.000000	0.900000

```
[38] df.shape

(1000, 8)
```

5. Bench marking alternate products (comparison with existing products/services)

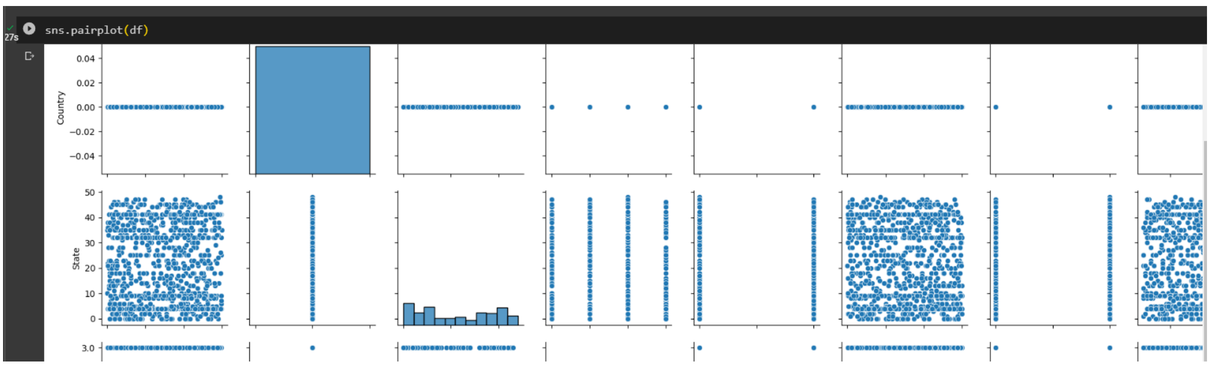
To benchmark the car price prediction product/service, a comparison can be made with existing products or services that offer similar functionalities. This involves evaluating their features, accuracy, scalability, user interface, and any additional benefits they provide to car manufacturers and dealerships.

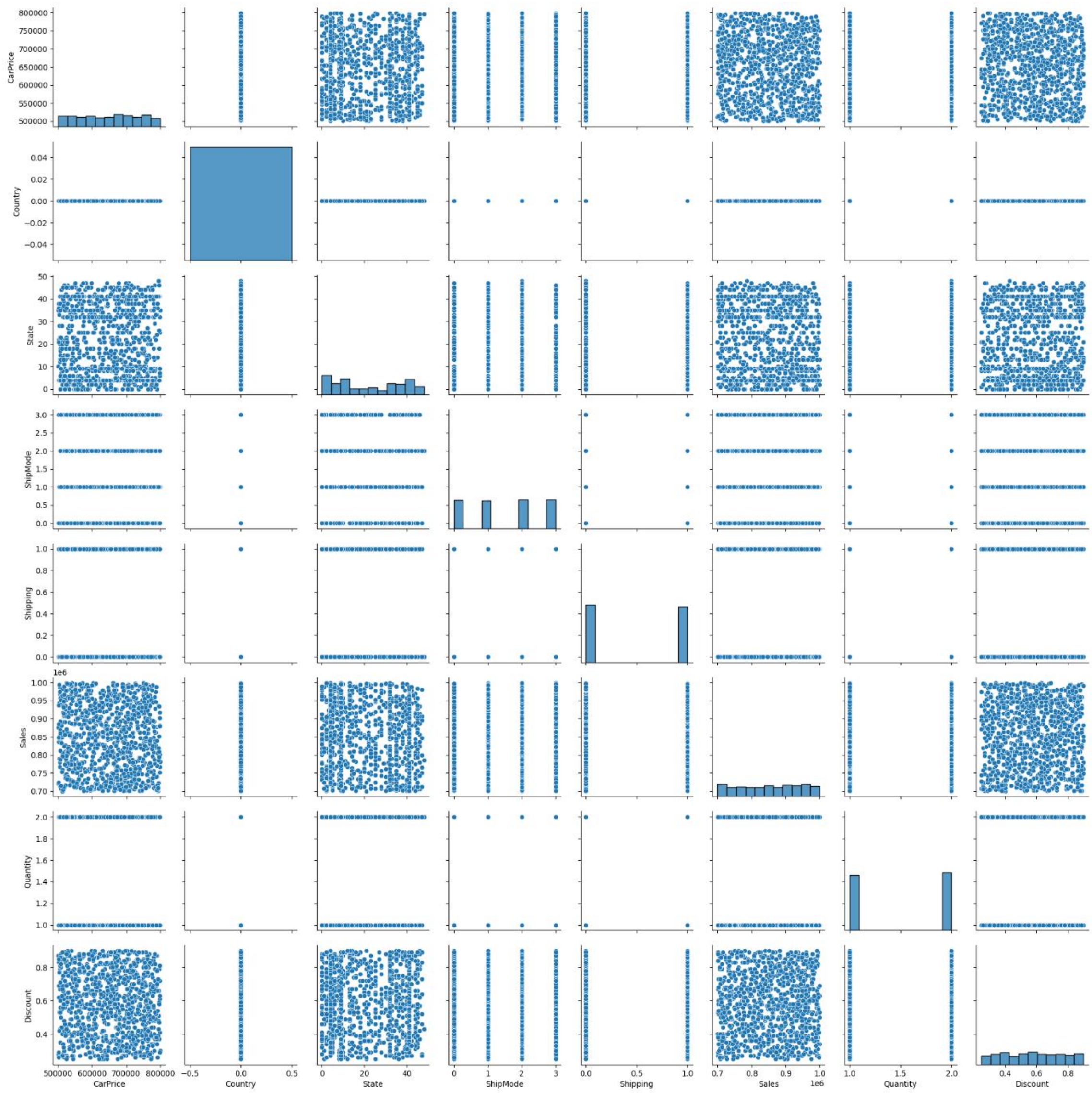
Heat Map



we can find the Correlation of all the columns. I use the matplotlib to resize the output of the image and using seaborn heatmap we can find a correlation between each of the columns

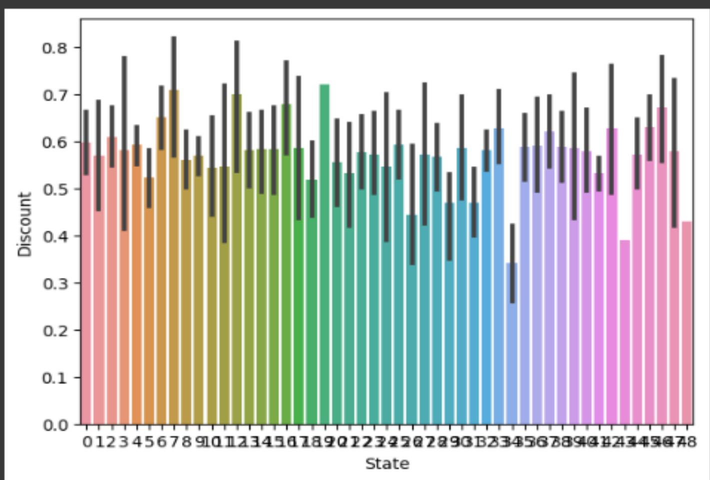
Pair plot:





Bar Plot

```
sns.barplot(data=df,x=df['State'],y=df['Discount'])
plt.show()
```



7. Applicable Regulations (government and environmental regulations imposed by countries)

There are several government and environmental regulations that apply to this product. These regulations include safety regulations, emissions regulations, and fuel efficiency regulations.

Pricing Transparency Regulations: Some countries or jurisdictions may have regulations or guidelines regarding pricing transparency and fairness. It is important to ensure that the car price prediction product/service adheres to these regulations, providing transparent and accurate price predictions based on relevant attributes.

Consumer Protection Regulations: Consumer protection laws and regulations vary across countries. These regulations aim to ensure fair business practices, prevent deceptive pricing strategies, and protect consumers from unfair or discriminatory treatment. Compliance with these regulations is essential to avoid any legal issues or reputational damage.

Advertising and Marketing Regulations: If the product/service involves any marketing or advertising components, it is important to comply with applicable regulations regarding truthful advertising, disclosure requirements, and fair marketing practices.

Customization Options: Customers may have specific needs and preferences regarding the attributes they want to consider for price prediction. The product/service should provide flexibility and customization options, allowing users to select and prioritize attributes according to their requirements.

Real-Time Updates: Customers may require real-time updates on car prices to adapt quickly to market fluctuations. The product/service should provide timely predictions and incorporate mechanisms to capture and reflect changing market conditions.

8. Applicable Constraints (need for space, budget, expertise)

Considerations regarding space, budget, and expertise should be considered. This includes the availability of computational resources, financial constraints for development and implementation, and the required expertise in machine learning, data analysis, and software engineering.

Space: The model will need to be developed in a small space. This is because the model is a software program, and software programs can be developed in a relatively small space.

Budget: The budget for the model will be limited. This is because the model is a startup, and startups typically have limited budgets.

Expertise: The model will require expertise in several areas, including:

> **Data science**: The model will need to use data science techniques to predict car prices.
>
> **Software development:** The model will need to be developed using software development tools and techniques.
>
> **Machine learning:** The model will need to use machine learning techniques to learn from the data and improve its predictions over time.

By considering the applicable constraints, you can increase the chances of success for your model.

9.Business Model (Monetization Idea)

The business model for this product is to sell it to car buyers. The product will be sold through a website and through car dealerships.

Business Model: The monetization idea for the car price prediction product/service can be based on subscription models, where car manufacturers and dealerships pay a recurring fee for access to the prediction tool. Alternatively, a licensing model can be considered, where the product/service is sold as a one-time purchase or with a usage-based pricing model. Additional revenue streams can include providing premium features or customized analytics reports.

Concept Generation (process of coming up with Idea): The concept for this product came from the idea that there is a need for a product that can help car buyers find the right car for their needs.

Concept Development (Brief summary of Product/Service will be developed): The product will be a website that allows car buyers to compare different cars based on their specifications and prices. The website will also provide car buying guides and car reviews.

10. Concept Generation (process of coming up with Idea)

The concept generation process involves brainstorming and ideation to generate innovative ideas for the car price prediction product/service. This can include exploring different machine learning algorithms, considering integration with existing supply chain management systems, and identifying potential value-added features such as market trend analysis or competitor benchmarking.

1.Includes Data pre-processing

2.Splitting the data into x and y variables

+ Code + Text

```
#choosing the dependent and independent values accross x and y
x=df.drop(columns="CarPrice",axis=1)
x.head()
```

	Country	State	ShipMode	Shipping	Sales	Quantity	Discount
0	0	1	3	1	744796.41	1	0.83
1	0	23	3	1	794773.17	1	0.79
2	0	13	2	0	968244.90	1	0.28
3	0	9	0	1	942213.82	2	0.76
4	0	22	2	0	879519.57	1	0.50

```
y=df[["CarPrice"]]
y.head()
```

	CarPrice
0	521963.45
1	672222.04
2	504465.72
3	646077.11
4	699890.24

3.normailzing the values to a scale of 0 to 1

```
[18] #normalizing the values and bringing them to the scale of o and 1
     from sklearn.preprocessing import MinMaxScaler
     scale=MinMaxScaler()
     x_scaled=pd.DataFrame(scale.fit_transform(x),columns = x.columns)
     y_scaled= pd.DataFrame(scale.fit_transform(y),columns = y.columns)
```

```
x_scaled.head()
```

	Country	State	ShipMode	Shipping	Sales	Quantity	Discount
0	0.0	0.020833	1.000000	1.0	0.148748	0.0	0.892308
1	0.0	0.479167	1.000000	1.0	0.315898	0.0	0.830769
2	0.0	0.270833	0.666667	0.0	0.896082	0.0	0.046154
3	0.0	0.187500	0.000000	1.0	0.809020	1.0	0.784615
4	0.0	0.458333	0.666667	0.0	0.599336	0.0	0.384615

```
[20] y_scaled.head()
```

	CarPrice
0	0.072067
1	0.574534
2	0.013554
3	0.487105
4	0.667057

4.Splitting the data into training and testing

```
[21] #splitting the data into training and testing data
     from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x_scaled,y_scaled,test_size=0.2,random_state=0)

[22] x_train.shape,x_test.shape,y_train.shape,y_test.shape

     ((800, 7), (200, 7), (800, 1), (200, 1))
```

5.Model fitting

Algorithms performed (Linear Regression, ANN Regressor, Decision tree regressor, Random Forest Regressor, XG Boost regressor)

LINEAR REGRESSION:

```
[23] #fitting linear regression model
     from sklearn.linear_model import LinearRegression
     model = LinearRegression()
     model.fit(x_train,y_train)
     y_pred=model.predict(x_test)

[24] #evaluation metrics
     #MEAN SQUARD ERROR
     from sklearn.metrics import mean_squared_error
     mse = mean_squared_error(y_test, y_pred)
     print('Mean Squared Error:', mse)
     #ROOT MEAN SQUARED ERROR
     import numpy as np
     from sklearn.metrics import mean_squared_error
     mse = mean_squared_error(y_test, y_pred)
     rmse = np.sqrt(mse)
     print('Root Mean Squared Error:', rmse)
     #MEAN ABSOLUTE ERROR
     from sklearn.metrics import mean_absolute_error
     mae = mean_absolute_error(y_test, y_pred)
     print('Mean Absolute Error:', mae)

     Mean Squared Error: 0.0880007761178696
     Root Mean Squared Error: 0.29664924762734457
     Mean Absolute Error: 0.25903561918606316
```

ANN REGRESSION:

```
[25] # ANN REGRESSION MODEL
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
```

```
[26] reg_model=Sequential()
     #Adding the input layer
     reg_model.add(Dense(7,activation="relu"))
     #Adding four hidden layers
     reg_model.add(Dense(128,activation="relu"))
     reg_model.add(Dense(64,activation="relu"))
     reg_model.add(Dense(32,activation="relu"))
     reg_model.add(Dense(16,activation="relu"))
     #Adding the output layer
     reg_model.add(Dense(1,activation="linear"))
```

```
[27] #compiling the model
     reg_model.compile(optimizer="adam",loss="mse",metrics=['mse','mae'])
```

fitting the model with an epoch of 40

```
[28] #fitting the ann regression model to the training data
     reg_model.fit(x_train,y_train,epochs=40,batch_size=2,validation_data=(x_test,y_test))
400/400 [==============================] - 1s 2ms/step - loss: 0.0796 - mse: 0.0796 - mae: 0.2411 - val_loss: 0.0893 - val_mse: 0.0893 - val_mae: 0.2628
Epoch 13/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0791 - mse: 0.0791 - mae: 0.2417 - val_loss: 0.0892 - val_mse: 0.0892 - val_mae: 0.2624
Epoch 14/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0791 - mse: 0.0791 - mae: 0.2426 - val_loss: 0.0884 - val_mse: 0.0884 - val_mae: 0.2594
Epoch 15/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0789 - mse: 0.0789 - mae: 0.2413 - val_loss: 0.0883 - val_mse: 0.0883 - val_mae: 0.2601
Epoch 16/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0782 - mse: 0.0782 - mae: 0.2403 - val_loss: 0.0896 - val_mse: 0.0896 - val_mae: 0.2639
Epoch 17/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0785 - mse: 0.0785 - mae: 0.2403 - val_loss: 0.0930 - val_mse: 0.0930 - val_mae: 0.2621
Epoch 18/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0775 - mse: 0.0775 - mae: 0.2376 - val_loss: 0.0901 - val_mse: 0.0901 - val_mae: 0.2613
Epoch 19/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0783 - mse: 0.0783 - mae: 0.2393 - val_loss: 0.0887 - val_mse: 0.0887 - val_mae: 0.2598
Epoch 20/40
400/400 [==============================] - 1s 3ms/step - loss: 0.0783 - mse: 0.0783 - mae: 0.2405 - val_loss: 0.0892 - val_mse: 0.0892 - val_mae: 0.2624
Epoch 21/40
400/400 [==============================] - 1s 3ms/step - loss: 0.0781 - mse: 0.0781 - mae: 0.2402 - val_loss: 0.0890 - val_mse: 0.0890 - val_mae: 0.2613
Epoch 22/40
400/400 [==============================] - 1s 3ms/step - loss: 0.0779 - mse: 0.0779 - mae: 0.2395 - val_loss: 0.0905 - val_mse: 0.0905 - val_mae: 0.2635
Epoch 23/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0782 - mse: 0.0782 - mae: 0.2399 - val_loss: 0.0893 - val_mse: 0.0893 - val_mae: 0.2608
Epoch 24/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0780 - mse: 0.0780 - mae: 0.2394 - val_loss: 0.0903 - val_mse: 0.0903 - val_mae: 0.2600
Epoch 25/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0769 - mse: 0.0769 - mae: 0.2369 - val_loss: 0.0898 - val_mse: 0.0898 - val_mae: 0.2628
Epoch 26/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0775 - mse: 0.0775 - mae: 0.2382 - val_loss: 0.0902 - val_mse: 0.0902 - val_mae: 0.2602
Epoch 27/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0764 - mse: 0.0764 - mae: 0.2365 - val_loss: 0.0898 - val_mse: 0.0898 - val_mae: 0.2621
Epoch 28/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0769 - mse: 0.0769 - mae: 0.2373 - val_loss: 0.0902 - val_mse: 0.0902 - val_mae: 0.2610
Epoch 29/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0764 - mse: 0.0764 - mae: 0.2361 - val_loss: 0.0897 - val_mse: 0.0897 - val_mae: 0.2625
Epoch 30/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0765 - mse: 0.0765 - mae: 0.2368 - val_loss: 0.0899 - val_mse: 0.0899 - val_mae: 0.2599
Epoch 31/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0768 - mse: 0.0768 - mae: 0.2381 - val_loss: 0.0909 - val_mse: 0.0909 - val_mae: 0.2642
Epoch 32/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0767 - mse: 0.0767 - mae: 0.2377 - val_loss: 0.0922 - val_mse: 0.0922 - val_mae: 0.2648
Epoch 33/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0772 - mse: 0.0772 - mae: 0.2378 - val_loss: 0.0881 - val_mse: 0.0881 - val_mae: 0.2608
Epoch 34/40
400/400 [==============================] - 1s 3ms/step - loss: 0.0758 - mse: 0.0758 - mae: 0.2338 - val_loss: 0.0906 - val_mse: 0.0906 - val_mae: 0.2648
Epoch 35/40
400/400 [==============================] - 1s 4ms/step - loss: 0.0766 - mse: 0.0766 - mae: 0.2361 - val_loss: 0.0898 - val_mse: 0.0898 - val_mae: 0.2629
Epoch 36/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0758 - mse: 0.0758 - mae: 0.2356 - val_loss: 0.0912 - val_mse: 0.0912 - val_mae: 0.2610
Epoch 37/40
400/400 [==============================] - 1s 2ms/step - loss: 0.0758 - mse: 0.0758 - mae: 0.2357 - val_loss: 0.0901 - val_mse: 0.0901 - val_mae: 0.2628
Epoch 38/40
```

```
[29] #evaluation
     y_pred_1=reg_model.predict(x_test)

7/7 [==============================] - 0s 2ms/step
```

```
[30] # Evaluate the model on test data using MSE
     from tensorflow import keras
     #MEAN SQUARD ERROR
     from sklearn.metrics import mean_squared_error
     mse_1 = mean_squared_error(y_test, y_pred)
     print('Mean Squared Error:', mse_1)
     #ROOT MEAN SQUARED ERROR
     import numpy as np
     from sklearn.metrics import mean_squared_error
     mse_1 = mean_squared_error(y_test, y_pred)
     rmse_1 = np.sqrt(mse_1)
     print('Root Mean Squared Error:', rmse_1)
     #MEAN ABSOLUTE ERROR
     from sklearn.metrics import mean_absolute_error
     mae_1 = mean_absolute_error(y_test, y_pred)
     print('Mean Absolute Error:', mae_1)

Mean Squared Error: 0.0880007761178696
Root Mean Squared Error: 0.29664924762734457
Mean Absolute Error: 0.25903561918606316
```

Taking the random input and testing the ANN regressor model

```
[31] df.head()
```

	CarPrice	Country	State	ShipMode	Shipping	Sales	Quantity	Discount
0	521963.45	0	1	3	1	744796.41	1	0.83
1	672222.04	0	23	3	1	794773.17	1	0.79
2	504465.72	0	13	2	0	968244.90	1	0.28
3	646077.11	0	9	0	1	942213.82	2	0.76
4	699890.24	0	22	2	0	879519.57	1	0.50

```
[32] #TESTING
     Testing=reg_model.predict([[4,2,5,3,879865.236894,2,0.566]])
     Testing

1/1 [==============================] - 0s 97ms/step
array([[10563.955]], dtype=float32)
```

RANDOM FOREST REGRESSOR

```
#Random forest regressor
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train,y_train)
```

```
<ipython-input-36-40f095ccc9ac>:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(x_train,y_train)
▾ RandomForestRegressor
RandomForestRegressor()
```

```
y_pred_2=rf.predict(x_test)
#MEAN SQUARD ERROR
from sklearn.metrics import mean_squared_error
mse_2 = mean_squared_error(y_test, y_pred_2)
print('Mean Squared Error:', mse_2)
#ROOT MEAN SQUARED ERROR
import numpy as np
from sklearn.metrics import mean_squared_error
mse_2= mean_squared_error(y_test, y_pred_2)
rmse_2 = np.sqrt(mse_2)
print('Root Mean Squared Error:', rmse_2)
#MEAN ABSOLUTE ERROR
from sklearn.metrics import mean_absolute_error
mae_2 = mean_absolute_error(y_test, y_pred_2)
print('Mean Absolute Error:', mae_2)
```

```
Mean Squared Error: 0.09758921887021015
Root Mean Squared Error: 0.3123927317819833
Mean Absolute Error: 0.27131428746177205
```

DECISION TREE REGRESSOR

```
#DECISION TREE REGRESSOR
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
y_pred_3 = dt.predict(x_test)
```

```
#MEAN SQUARD ERROR
from sklearn.metrics import mean_squared_error
mse_3 = mean_squared_error(y_test, y_pred_3)
print('Mean Squared Error:', mse_3)
#ROOT MEAN SQUARED ERROR
import numpy as np
from sklearn.metrics import mean_squared_error
mse_3= mean_squared_error(y_test, y_pred_3)
rmse_3 = np.sqrt(mse_3)
print('Root Mean Squared Error:', rmse_3)
#MEAN ABSOLUTE ERROR
from sklearn.metrics import mean_absolute_error
mae_3 = mean_absolute_error(y_test, y_pred_3)
print('Mean Absolute Error:', mae_3)
```

```
Mean Squared Error: 0.1893094507504212
Root Mean Squared Error: 0.4350970589999675
Mean Absolute Error: 0.3609103988746991
```

XG BOOST REGRESSOR

```
[40] #xg boost regressor
     import xgboost as xgb
     xgbr = xgb.XGBRegressor()
     xgbr.fit(x_train, y_train)
     y_pred_4= xgbr.predict(x_test)
```

```
#MEAN SQUARD ERROR
from sklearn.metrics import mean_squared_error
mse_4 = mean_squared_error(y_test, y_pred_4)
print('Mean Squared Error:', mse_4)
#ROOT MEAN SQUARED ERROR
import numpy as np
from sklearn.metrics import mean_squared_error
mse_4= mean_squared_error(y_test, y_pred_4)
rmse_4 = np.sqrt(mse_4)
print('Root Mean Squared Error:', rmse_4)
#MEAN ABSOLUTE ERROR
from sklearn.metrics import mean_absolute_error
mae_4= mean_absolute_error(y_test, y_pred_4)
print('Mean Absolute Error:', mae_4)
```

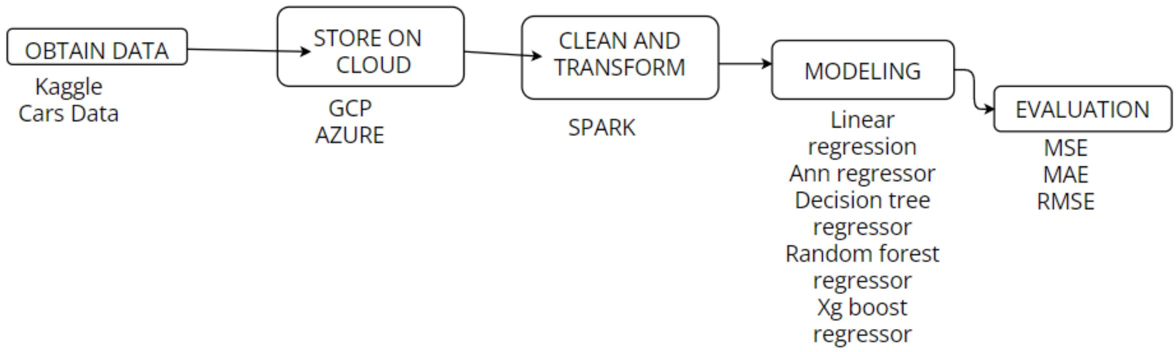
```
Mean Squared Error: 0.11000080223198404
Root Mean Squared Error: 0.33166368844355576
Mean Absolute Error: 0.2848840068834364
```

By analysing the evaluation metrics of all models which we had applied to the dataset we can know that decision tree regressor is giving the good mean squared error, root mean squared error, mean absolute error.

11. Concept Development

The concept development for the car price prediction system using the given dataset involves creating a machine learning model that can predict the price of a car based on various attributes such as car maker, car model, car colour, car model year, customer details, order details, and more. The goal is to develop a system that can assist customers and sellers in determining the estimated price of a car based on its features and other relevant factors.

STEPS:



12.Final Product Prototype (abstract) with Schematic Diagram

The final product prototype will be a website that allows car buyers to compare different cars based on their specifications and prices. The website will also provide car buying guides and car reviews. The website will be developed using a variety of technologies, including:

Web development: The website will be developed using HTML, CSS, and JavaScript.

Data science: The car price prediction model will be developed using data science techniques.

Machine learning: The car price prediction model will be developed using machine learning algorithms.

The website will be hosted on a cloud server. The website will be accessed by car buyers using a variety of devices, including:

Desktop computers: Car buyers can access the website using a desktop computer.

Laptop computers: Car buyers can access the website using a laptop computer.

Mobile phones: Car buyers can access the website using a mobile phone.

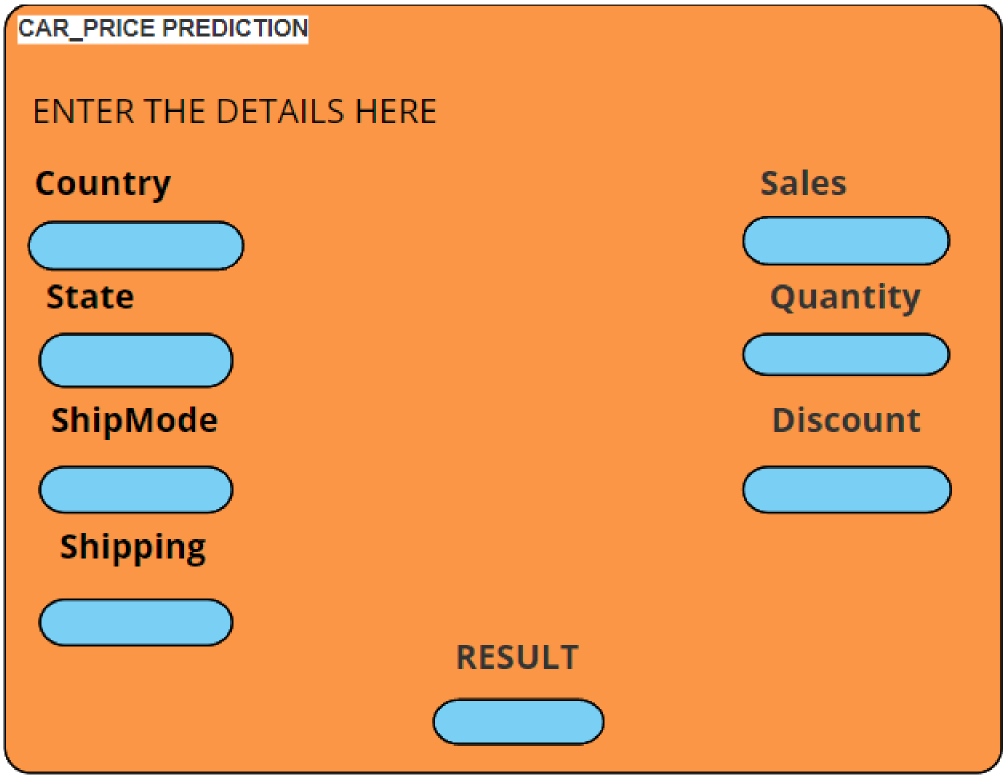
The website will be marketed to car buyers using a variety of methods, including:

Search engine optimization: The website will be optimized for search engines so that it can be easily found by car buyers.

Pay-per-click advertising: The website will be advertised using pay-per-click advertising so that it can be seen by car buyers who are searching for information about cars.

Social media marketing: The website will be marketed on social media so that it can be seen by car buyers who are active on social media.

The website will be updated regularly with new information about cars. The website will also be improved over time to make it more user-friendly and to provide car buyers with more information about cars.



13. Product details

13.1 how does it work.

After training the model we pass some parameters to the model so that the model predicts the output as car price prediction based on the input parameters.

13.2 Data Sources

The dataset is taken from Kaggle.

https://www.kaggle.com/datasets/prashantk93/supply-chain-management-for-car

13.3 Algorithms, frameworks, software etc.

Algorithms used: Linear regression, Ann regressor, Decision tree regressor, Random forest regressor, xgboost regressor

Framework: TensorFlow for Ann regressor

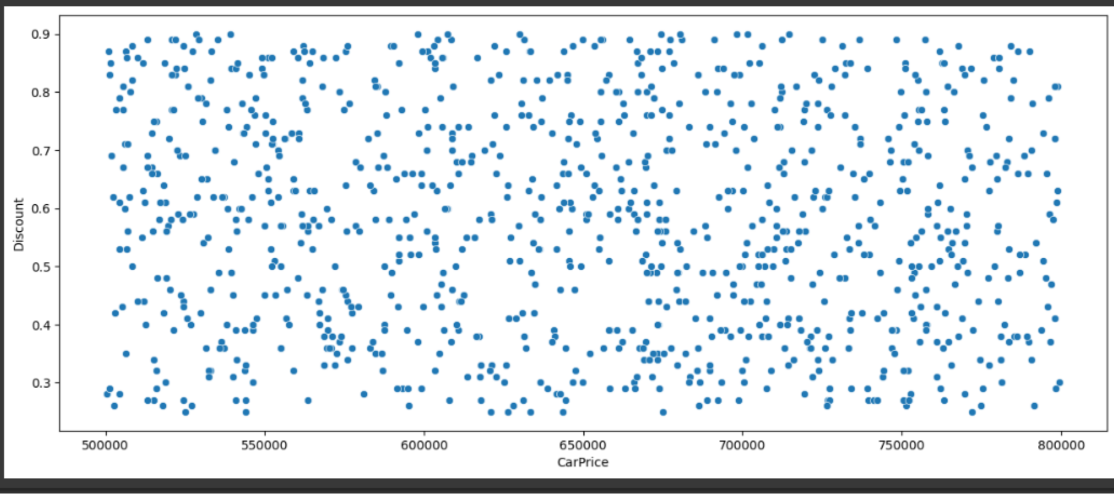
Software: Google collab

14. Code Implementation/Validation on Small Scale

Some Basic Visualizations on Real World or Augmented Data

Scatter plot→**Bivariate Analysis**

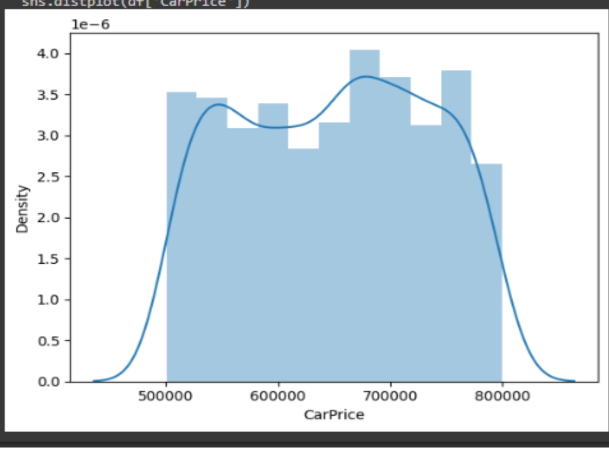
```
plt.figure(figsize=(15,6))
sns.scatterplot(x=df['CarPrice'],y=df['Discount'])
plt.show()
```



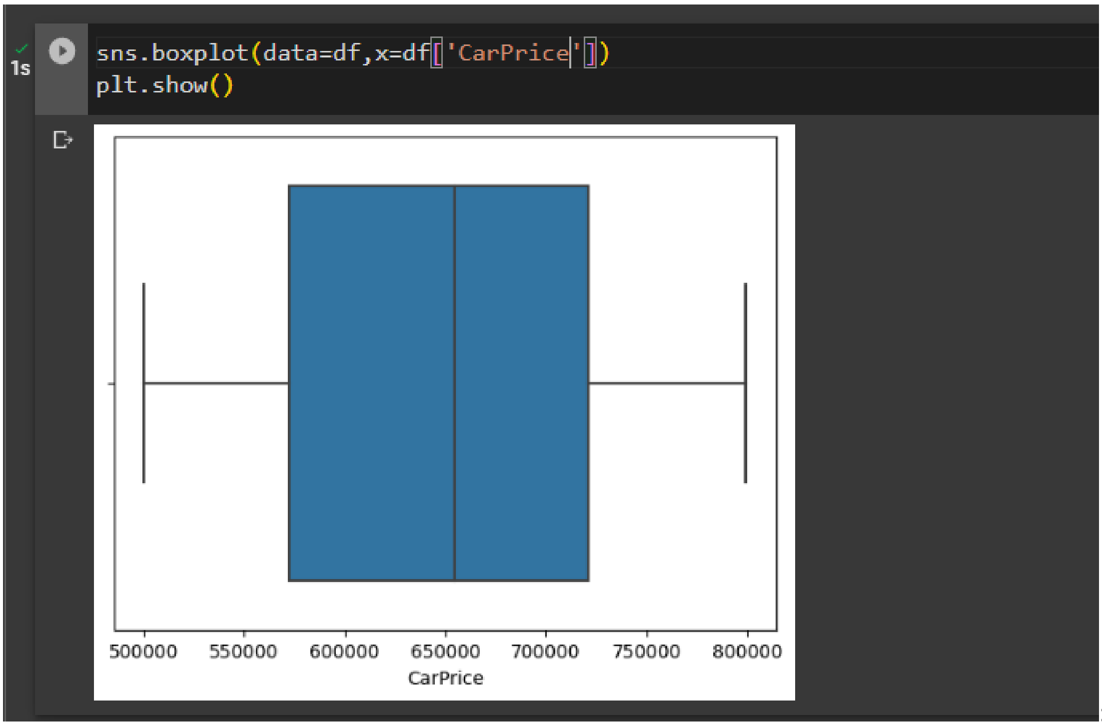
Dist plot→**univariate analysis.**

```
sns.distplot(df['CarPrice'])
plt.show()
```

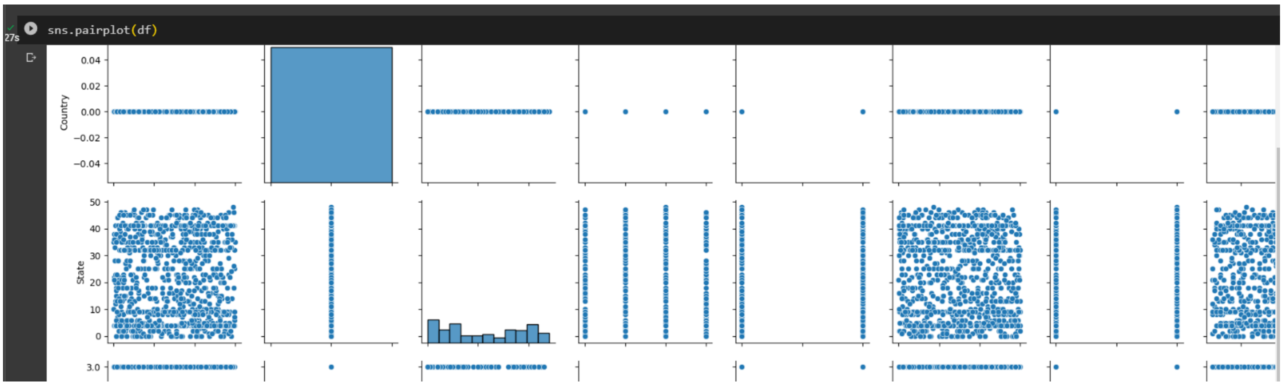
```
<ipython-input-42-89e9e6d079f2>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['CarPrice'])
```

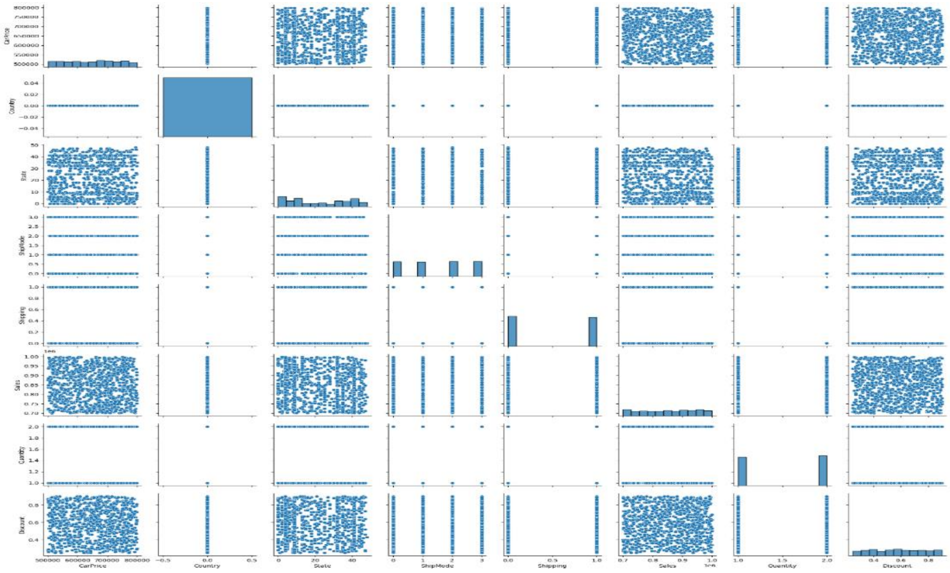


Box plot



Pair plot→**Multivariate analysis**





EDA

Data cleaning is also an application on EDA

```
[1] #importing required libraries
    import pandas as pd
    import numpy as np
    import seaborn as sns
```

```
[2] df=pd.read_csv("/content/supply chain management for cars.csv")
```

```
df.head()
```

	SupplierID	SupplierAddress	SupplierName	SupplierContactDetails	ProductID	CarMaker	CarModel	CarColor	CarModelYear	CarPrice	...	ShipDate	ShipMode	Shipping	PostalCode	Sales	Quantity	Discount	CreditCardType	CreditCar
0	1	542 Dayton Center	Bubbletube	871-57-6028	8893	Dodge	Ram 2500	Goldenrod	2007	521963.45	...	2019/03/14	Standard Class	Truck	99522	744796.41	1	0.83	diners-club-carte-blanche	3040801604256
1	2	0674 Springview Circle	Tagopia	337-64-4060	9444	Toyota	Tundra	Crimson	2010	672222.04	...	2019/03/06	Standard Class	Truck	56398	794773.17	1	0.79	jcb	354922111223776
2	3	70 Autumn Leaf Center	Zoomdog	218-19-1802	253	GMC	Savana 1500	Crimson	2011	504465.72	...	2019/01/20	Second Class	Air	60674	968244.90	1	0.28	jcb	35571596081809(
3	4	649 Corben Lane	Oozz	635-15-3112	1283	Volkswagen	Cabriolet	Fuscia	1990	646077.11	...	2019/03/16	First Class	Truck	32885	942213.82	2	0.76	jcb	35299092236639
4	5	94 Namekagon Point	Kare	849-23-6788	8905	Mercury	Mariner	Teal	2009	699890.24	...	2019/01/29	Second Class	Air	48232	879519.57	1	0.50	china-unionpay	56022359785415

5 rows × 33 columns

Dropping columns

```
[4] #dropping unnecessary columns
df.drop(columns=["CarModelYear","CarColor","CreditCardType",  "CustomerFeedback","SupplierID" ,"SupplierAddress" ,"SupplierName","CreditCard","CarModel","CarMaker","SupplierContac
```

```
[5] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CarPrice  1000 non-null   float64
 1   Country   1000 non-null   object
 2   State     1000 non-null   object
 3   ShipMode  1000 non-null   object
 4   Shipping  1000 non-null   object
 5   Sales     1000 non-null   float64
 6   Quantity  1000 non-null   int64
 7   Discount  1000 non-null   float64
dtypes: float64(3), int64(1), object(4)
memory usage: 62.6+ KB
```


```
[5] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CarPrice  1000 non-null   float64
 1   Country   1000 non-null   object
 2   State     1000 non-null   object
 3   ShipMode  1000 non-null   object
 4   Shipping  1000 non-null   object
 5   Sales     1000 non-null   float64
 6   Quantity  1000 non-null   int64
 7   Discount  1000 non-null   float64
dtypes: float64(3), int64(1), object(4)
memory usage: 62.6+ KB
```


```
[6] df.describe()
```

	CarPrice	Sales	Quantity	Discount
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	649092.193460	853098.713020	1.512000	0.577360
std	85427.262753	88538.571965	0.500106	0.187478
min	500412.460000	700321.490000	1.000000	0.250000
25%	572393.805000	775655.062500	1.000000	0.410000
50%	654965.000000	858117.980000	2.000000	0.580000
75%	721050.725000	932854.565000	2.000000	0.740000
max	799454.240000	999315.690000	2.000000	0.900000

```
[7] #shape of the dataset
df.shape
```

```
(1000, 8)
```

Checking for null values

```
[9] #checking for null values
df.isnull().sum()
```

```
CarPrice    0
Country     0
State       0
ShipMode    0
Shipping    0
Sales       0
Quantity    0
Discount    0
dtype: int64
```

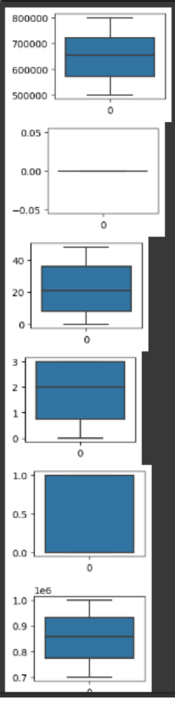
Using label encoder for categorical

```
[10] #replacing categorical variables with numeric values using labelencoder
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Country=le.fit_transform(df.Country)
df.State=le.fit_transform(df.State)
df.ShipMode=le.fit_transform(df.ShipMode)
df.Shipping=le.fit_transform(df.Shipping)
```

Outlier detection checking

```
[12] names=["CarPrice", "Country", "State", "ShipMode" ,"Shipping" ,"Sales", "Quantity" ,"Discount"]
```

```
#outlier detection
import matplotlib.pyplot as plt
for i,j in enumerate(names):
  plt.figure(figsize=(12,15))
  plt.subplot(8,5,i+1)
  sns.boxplot(df[j])
```



Normalizing values using MinMaxScaler technique

```
[14] #choosing the dependent and independent values accross x and y
x=df.drop(columns="CarPrice",axis=1)
x.head()
```

	Country	State	ShipMode	Shipping	Sales	Quantity	Discount
0	0	1	3	1	744796.41	1	0.83
1	0	23	3	1	794773.17	1	0.79
2	0	13	2	0	968244.90	1	0.28
3	0	9	0	1	942213.82	2	0.76
4	0	22	2	0	879519.57	1	0.50

```
y=df[["CarPrice"]]
y.head()
```

	CarPrice
0	521963.45
1	672222.04
2	504465.72
3	646077.11
4	699890.24

```
[16] #normalizing the values and bringing them to the scale of o and 1
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
x_scaled=pd.DataFrame(scale.fit_transform(x),columns = x.columns)
y_scaled= pd.DataFrame(scale.fit_transform(y),columns = y.columns)
```

GITHUB LINK: https://github.com/rohithreddy999/Feynn-lab

Conclusion:

It benefits in the automotive sector for both buyers and sellers. The system can precisely estimate the price of a car by utilising machine learning techniques and the extensive dataset that contains parameters like car maker, car model, car colour, customer details, and order information.

A regression model was trained and improved during the concept development process utilising methods like XGBoost, Random Forest, or Decision Tree Regressor. To assure the model's accuracy and performance, it was assessed using a variety of assessment measures, including mean squared error (MSE), root mean square error (RMSE), mean absolute error (MAE)

The user-friendly interface of the automobile price prediction system enables users to enter the pertinent car parameters and obtain an estimated price based on the trained model. This empowers customers to make informed decisions when buying or selling cars and provides sellers with valuable insights into pricing strategies.

Continuous improvement is crucial for the car price prediction system. Regular updates and refinements based on new data and customer feedback help enhance the accuracy and performance of the model over time. By monitoring the model's performance and gathering feedback, the system can be fine-tuned to better meet the needs of users and adapt to changing market dynamics.

-- END--