Car Price Prediction Report

Submitted By:-Sujay Nimbalkar

Submitted To:-Swati Mahaseth(Flip Robo Technologies)

1. Introduction

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

2. Loading Data and Explanation of Features

In [4]:
```
df=pd.DataFrame(data=da)
df
```

Out[4]:

	Unnamed: 0	Name	Model	Transmission	Km_driven	Owner	Fuel	Discount	Discount_price	Sale_price
0	0	2020 Renault Kwid	1.0 RXT Opt AT Automatic	Automatic	3,252	1st Owner	Petrol	₹5,000	4,31,699	4,36,699
1	1	2013 Maruti Alto 800	LXI Manual	Manual	13,807	1st Owner	Petrol	₹11,000	2,16,899	2,27,899
2	2	2020 Maruti New Wagon-R	VXI 1.0 Manual	Manual	3,865	1st Owner	Petrol	₹5,000	5,16,399	5,21,399
3	3	2019 Maruti Swift	VXI Manual	Manual	3,362	1st Owner	Petrol	₹5,000	6,20,699	6,25,699
4	4	2020 Maruti Baleno	SIGMA 1.2 K12 Manual	Manual	5,645	1st Owner	Petrol	₹6,000	5,75,299	5,81,299
...	...	...	...	...	...	...	...	...	...	...
5335	5335	2016 Maruti Wagon R 1.0	VXI	Manual	33,463	1st Owner	Petrol	₹6,000	3,88,699	3,94,699
5336	5336	2015 Hyundai Eon	SPORTZ	Manual	15,346	3rd Owner	Petrol	₹20,000	3,01,299	3,21,299
5337	5337	2012 Maruti Wagon R 1.0	VXI	Manual	38,553	1st Owner	Petrol	₹12,000	3,15,199	3,27,199
5338	5338	2018 Tata Tiago	XT 1.2 REVOTRON	Manual	11,253	1st Owner	Petrol	₹7,000	4,86,799	4,93,799
5339	5339	2017 Maruti Alto 800	VXI	Manual	21,239	1st Owner	Petrol	₹37,000	3,20,999	3,57,999

5340 rows × 10 columns

Here we have imported the data set we have 5340 rows and 10 columns as we can see above

3. Exploratory Data Analysis (EDA)

Here we have sorted data types, Columns, sales price, and below is mentioned the info

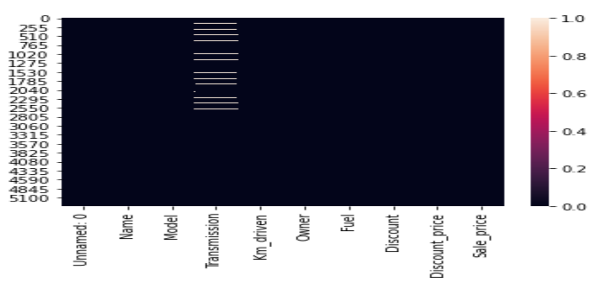
In [11]:
```
df.info
```

Out[11]:
```
<bound method DataFrame.info of       Unnamed: 0                     Name                    Model  \
0              0        2020 Renault Kwid  1.0 RXT Opt AT Automatic
1              1     2013 Maruti Alto 800               LXI Manual
2              2  2020 Maruti New Wagon-R           VXI 1.0 Manual
3              3        2019 Maruti Swift               VXI Manual
4              4       2020 Maruti Baleno     SIGMA 1.2 K12 Manual
...          ...                      ...                      ...
5335        5335  2016 Maruti Wagon R 1.0                      VXI
5336        5336         2015 Hyundai Eon                   SPORTZ
5337        5337  2012 Maruti Wagon R 1.0                      VXI
5338        5338          2018 Tata Tiago          XT 1.2 REVOTRON
5339        5339     2017 Maruti Alto 800                      VXI

     Transmission Km_driven      Owner    Fuel Discount Discount_price  \
0       Automatic     3,252  1st Owner  Petrol   ₹5,000       4,31,699
1          Manual    13,807  1st Owner  Petrol  ₹11,000       2,16,899
2          Manual     3,865  1st Owner  Petrol   ₹5,000       5,16,399
3          Manual     3,362  1st Owner  Petrol   ₹5,000       6,20,699
4          Manual     5,645  1st Owner  Petrol   ₹6,000       5,75,299
...           ...       ...        ...     ...      ...            ...
5335       Manual    33,463  1st Owner  Petrol   ₹6,000       3,88,699
5336       Manual    15,346  3rd Owner  Petrol  ₹20,000       3,01,299
5337       Manual    38,553  1st Owner  Petrol  ₹12,000       3,15,199
5338       Manual    11,253  1st Owner  Petrol   ₹7,000       4,86,799
5339       Manual    21,239  1st Owner  Petrol  ₹37,000       3,20,999

     Sale_price
0      4,36,699
```


We have also Checked the missing values where we can see that there are missing values in the transmission column.



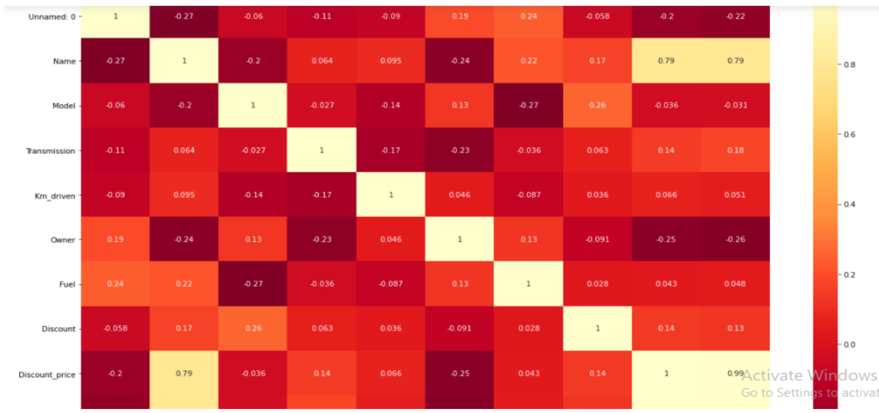
4 Unvairate Analysis

We have plotted the box plots with each factor for Unvairate Analysis

5.Bivariate Analysis

We have used strip plot for Bivariate Variate analysis to see the relation between the target columns

6. Multivariate Analysis



Major Observations Done

Now we can clearly indentify the correlation of independent variable with the target variable"SaLe_Price".

Light shades are highly positively correlated.

Sale price is negatively correlated with Unnamed: o and Owner column.

Sale price is positively correlated with discount price and name column

7. Data Cleaning

```
# Data Cleaning

In [54]: # Dropping the highly correlated columns.
df.drop('Unnamed: 0',axis=1,inplace=True)
df.drop('Owner',axis=1,inplace=True)

In [55]: df.head()
Out[55]:
```

	Name	Model	Transmission	Km_driven	Fuel	Discount	Discount_price	Sale_price
0	32	0	0	22	1	14	24	23
1	6	9	1	10	1	1	0	0
2	31	20	1	26	1	14	31	32
3	27	24	1	23	1	14	35	36
4	30	13	1	36	1	15	34	35

Here we have dropped the Unnamed: 0 and owner column due to its highly negative correlation with the sale_price column.These columns impacts our data negatively.

Here we have deleted the unnecessary owner column as it is highly negative corelatiob with the sale price as a result it hampers the data in high range

Then we have also separated the it into train test split and then we are finding the best algorithm using the best models Altogether

```
score: LinearRegression()
0.9834959852057372
score: DecisionTreeRegressor()
1.0
score: SVR()
0.9999070429857579
score: KNeighborsRegressor()
1.0
score: Lasso(alpha=0.0001)
0.9834956938771806
score: Ridge(alpha=0.0001)
0.9834959848507028
```

Above we can clearly see that we are getting the accuracy with r2_score with the decision Tree and KNeighbors Regressior

Accuracy can also be due to over fitting so we will check for cross validation.

```
In [80]: from sklearn.model_selection import cross_val_score

In [81]: for m in model:
             score=cross_val_score(m,x,y,cv=5)
             print('score:',m)
             print(score)
             print(score.mean())
score: LinearRegression()
[0.9787249  0.97854213 0.98284689 0.97999383 0.97984505]
0.9799905583407849
score: DecisionTreeRegressor()
[1. 1. 1. 1. 1.]
1.0
score: SVR()
[0.99993329 0.99993311 0.99992386 0.99973016 0.99971868]
0.9998478220473132
score: KNeighborsRegressor()
[1. 1. 1. 1. 1.]
1.0
score: Lasso(alpha=0.0001)
[0.978725   0.97854226 0.98284691 0.97999582 0.97984702]
0.9799914032992613
score: Ridge(alpha=0.0001)
[0.9787249  0.97854213 0.98284689 0.97999383 0.97984505]
0.9799905588040844
```

From the cross validation score we have seen that DecisionTreeRegressor() and KNeighborsRegressor() has least cross validation score.

Hyper parameter tuning for DecisionTreeRegressor

```
In [90]: dtr=DecisionTreeRegressor()
         grid_param={'criterion':['mse','mae','friedman_mse','poisson']}
         gd_sr=GridSearchCV(dtr,
                            scoring='accuracy',
                            param_grid=grid_param,
                            cv=5)
         gd_sr.fit(x_train,y_train)
         best_parameters=gd_sr.best_params_
         print(best_parameters)

         {'criterion': 'mse'}

In [91]: dtr1=DecisionTreeRegressor(criterion= 'mse')
         dtr1.fit(x_train,y_train)

Out[91]: DecisionTreeRegressor()

In [92]: dtr1.score(x_test,y_test)

Out[92]: 1.0
```

Here we can clearly see that hyper parameter tuning with decision tree regressor is giving 100% accuracy score which is best

Hyper parameter tuning for KNeighborsRegressor

```
In [94]: param={'n_neighbors':np.arange(1,12,2),
                'weights':['uniform','distance']}
         knr=KNeighborsRegressor()
         gscv=GridSearchCV(knr,param,cv=5)
         gscv.fit(x_train,y_train)

         best_parameters=gscv.best_params_
         print(best_parameters)

         {'n_neighbors': 1, 'weights': 'uniform'}

In [95]: gscv.best_score_

Out[95]: 1.0

In [96]: knr1=KNeighborsRegressor(n_neighbors=11, weights= 'distance')
         knr1.fit(x_train,y_train)

Out[96]: KNeighborsRegressor(n_neighbors=11, weights='distance')

In [97]: knr1.score(x_test,y_test)

Out[97]: 1.0
```

Here also with KNeighborsRegressor we are getting the accuracy score of 100% which is again best,So we are selecting DecisionTreeRegressor as our best model

Saving the Best Model

```
In [99]: import joblib
         joblib.dump(dtr1, "dtr1carfile.obj")

         dtr1_from_joblib =joblib.load("dtr1carfile.obj")

In [ ]:
```