

Capstone Submission



**DATA SCIENCE WITH PYTHON
BY SKILL ACADEMY**

BEHERA PITER

Used Car Price Prediction



THIS CAPSTONE PROJECT REVOLVES AROUND EXPLORING AND PREDICTING THE PRICES OF OLD CARS. BY INITIALLY ANALYZING THE DATASET IN EXCEL, I'VE IDENTIFIED NUMEROUS COLUMNS DIRECTLY IMPACTING THE CAR PRICES. NOW, I'M EMBARKING ON THE EXPLORATION AND MODEL-BUILDING PHASE USING PYTHON TO DELVE DEEPER INTO THESE INSIGHTS AND DEVELOP ACCURATE PRICE PREDICTION MODELS."

Analyzing Dataset

```
df=pd.read_csv('CAR DETAILS.csv')
df.sample(10)
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
2988	Tata Manza Club Class Quadrajet90 LS	2014	200000	100000	Diesel	Individual	Manual	First Owner
1211	Tata Nano Cx BSIV	2012	75000	35000	Petrol	Individual	Manual	Second Owner
1476	Honda Brio VX	2017	390000	45000	Petrol	Dealer	Manual	First Owner
640	Maruti Eeco 5 Seater Standard BSIV	2019	380000	5000	Petrol	Individual	Manual	First Owner
1264	Toyota Etios GD	2012	300000	124000	Diesel	Individual	Manual	First Owner
1343	Maruti Alto LX	2010	110000	88000	Petrol	Individual	Manual	Second Owner
1812	Ford Ikon 1.8 D	2005	114999	92645	Diesel	Dealer	Manual	Fourth & Above Owner
3746	Maruti Wagon R VXI BS IV	2017	430000	17000	Petrol	Individual	Manual	First Owner
584	Hyundai i10 Magna	2012	229999	49824	Petrol	Dealer	Manual	First Owner
329	Renault Duster 110PS Diesel RxZ	2012	350000	110000	Diesel	Individual	Manual	Second Owner

```
#Checking DataFrame Shape
df.shape
```

```
(4340, 8)
```

Handling Duplicates and Null Values

```
#Checking Null Values  
df.isnull().sum()
```

[4]

```
name          0  
year          0  
selling_price 0  
km_driven     0  
fuel          0  
seller_type   0  
transmission  0  
owner         0  
dtype: int64
```

```
#Checking Duplicates  
df.duplicated().sum()  
print('Duplicates Value %',763/4339*100)
```

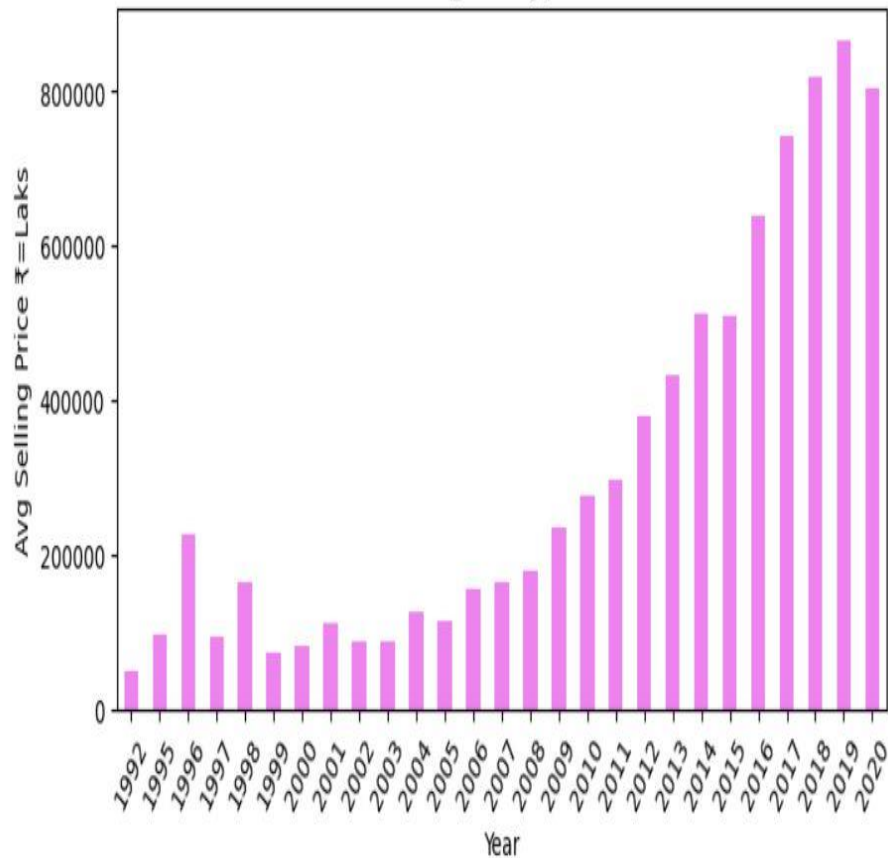
[5]

```
Duplicates Value % 17.5846969347776
```

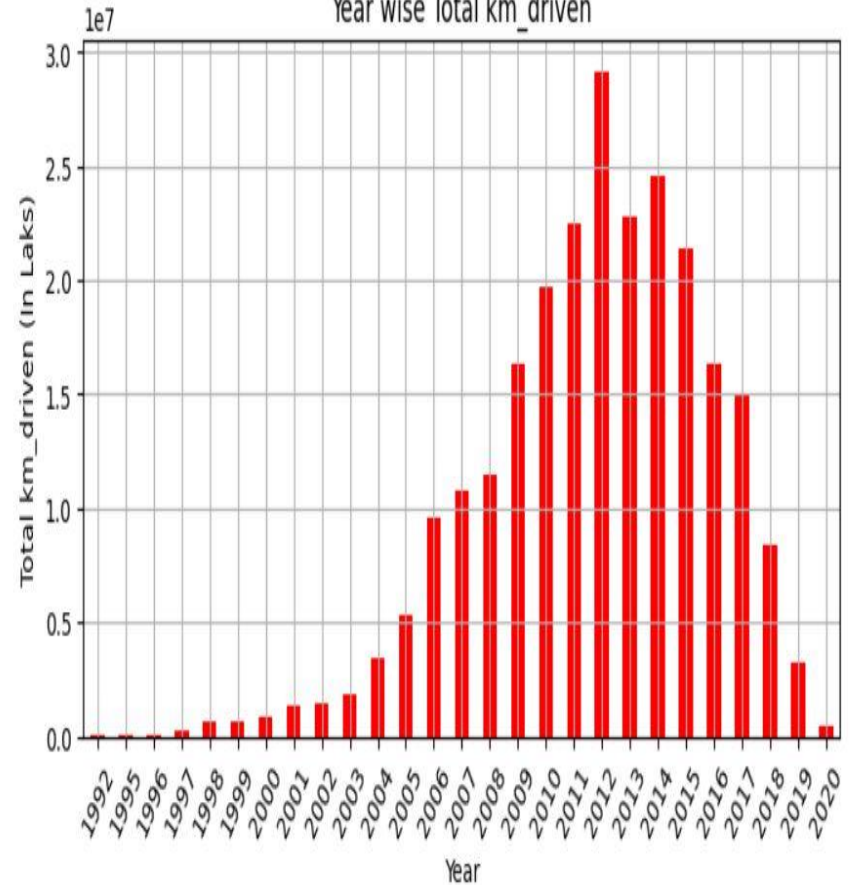
```
#Dropping Duplicates.(here are so many duplicates values.)  
df.drop_duplicates(inplace=True)
```

Inference: Here, we clearly see that over the years, the total kilometers driven by cars being resold vary. Before 2012, the usage was not as high, but it peaked in 2012. After that, the trend slowed down, and by 2020, cars arriving for resale had driven less than 50,000 kilometers.

Year wise Avg Selling price ₹=Laks

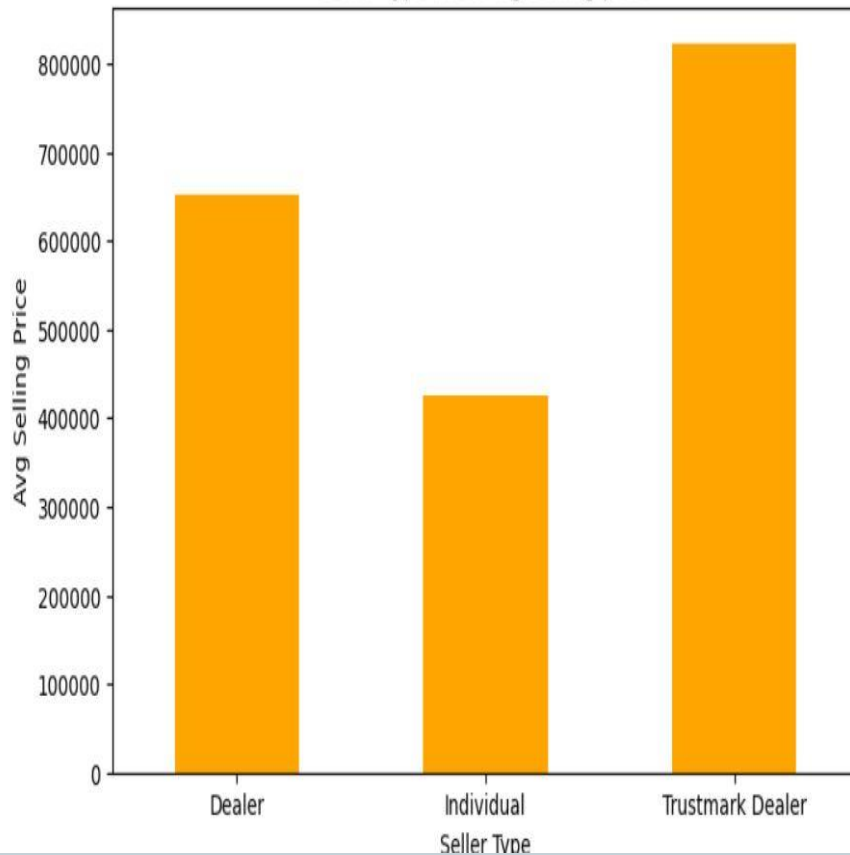


Year wise Total km_driven

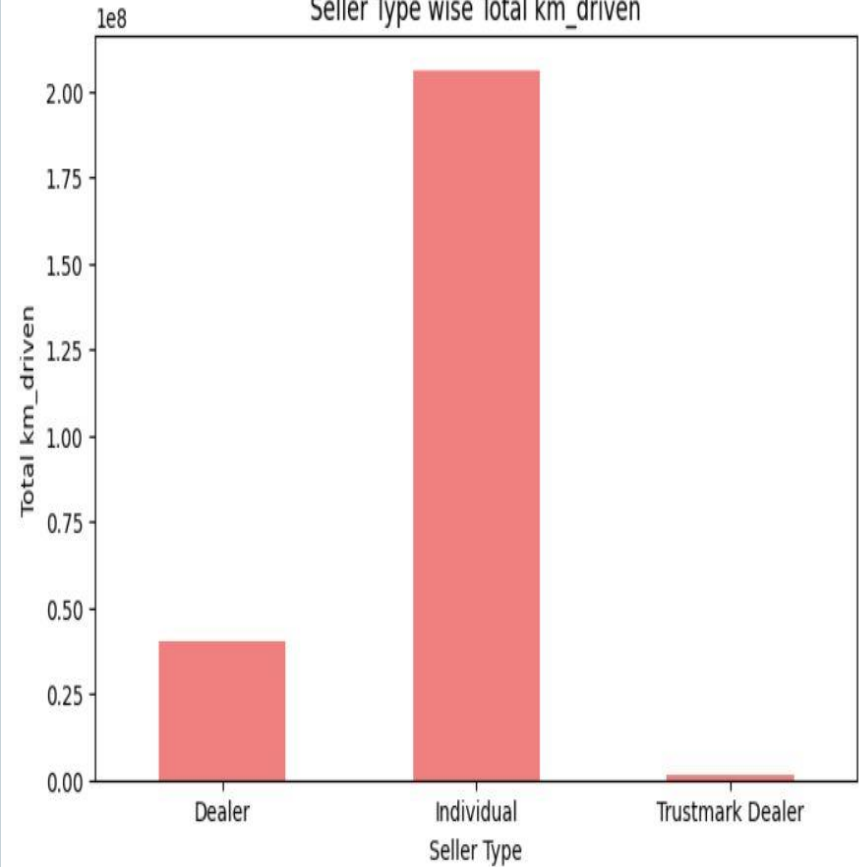


Inference: The selling price is higher in the Trustmaker Dealer category with a lower rate of distance driven by cars. Conversely, individual selling prices are lower when the kilometers driven are higher.

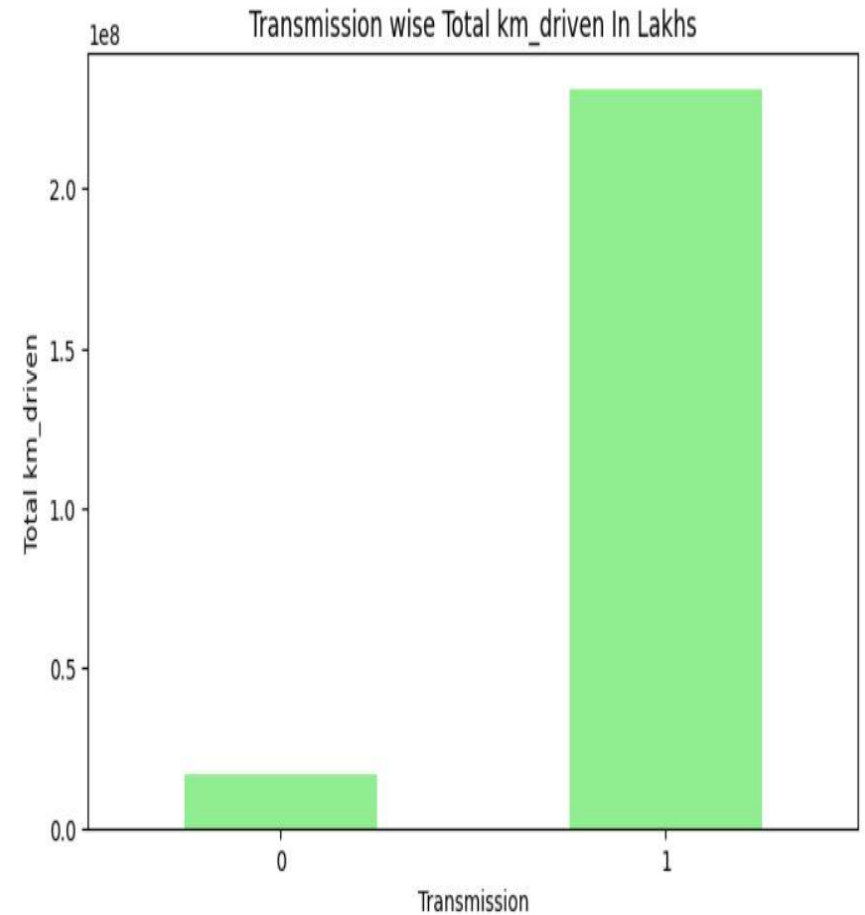
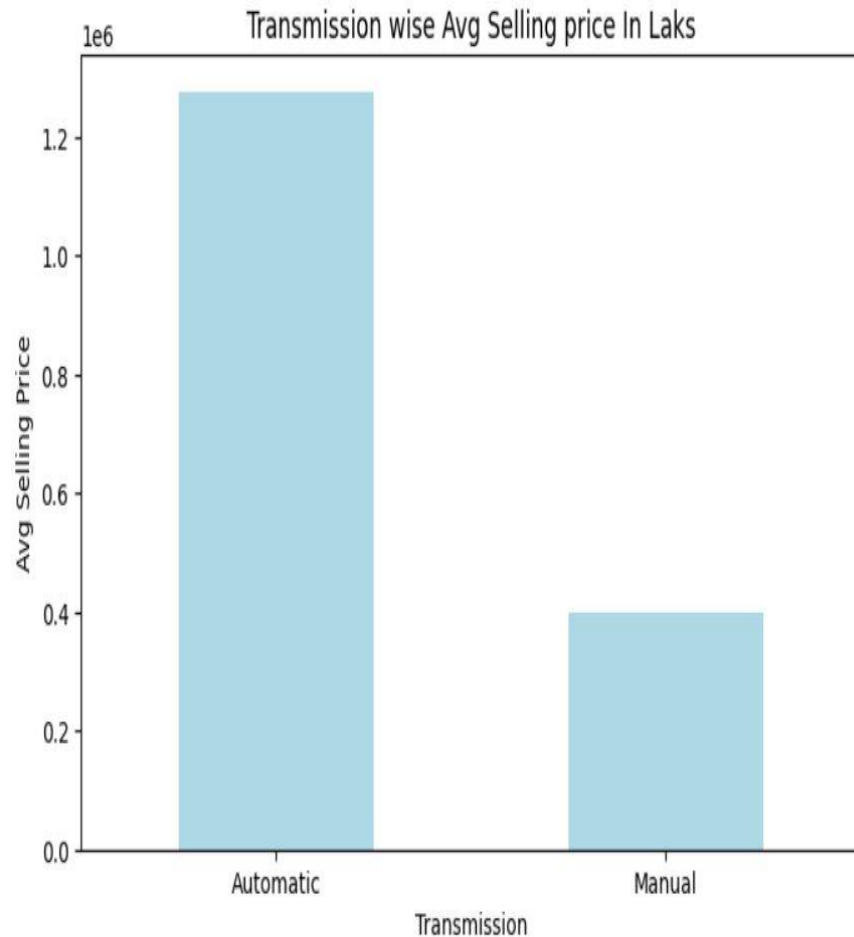
Seller Type wise Avg Selling price



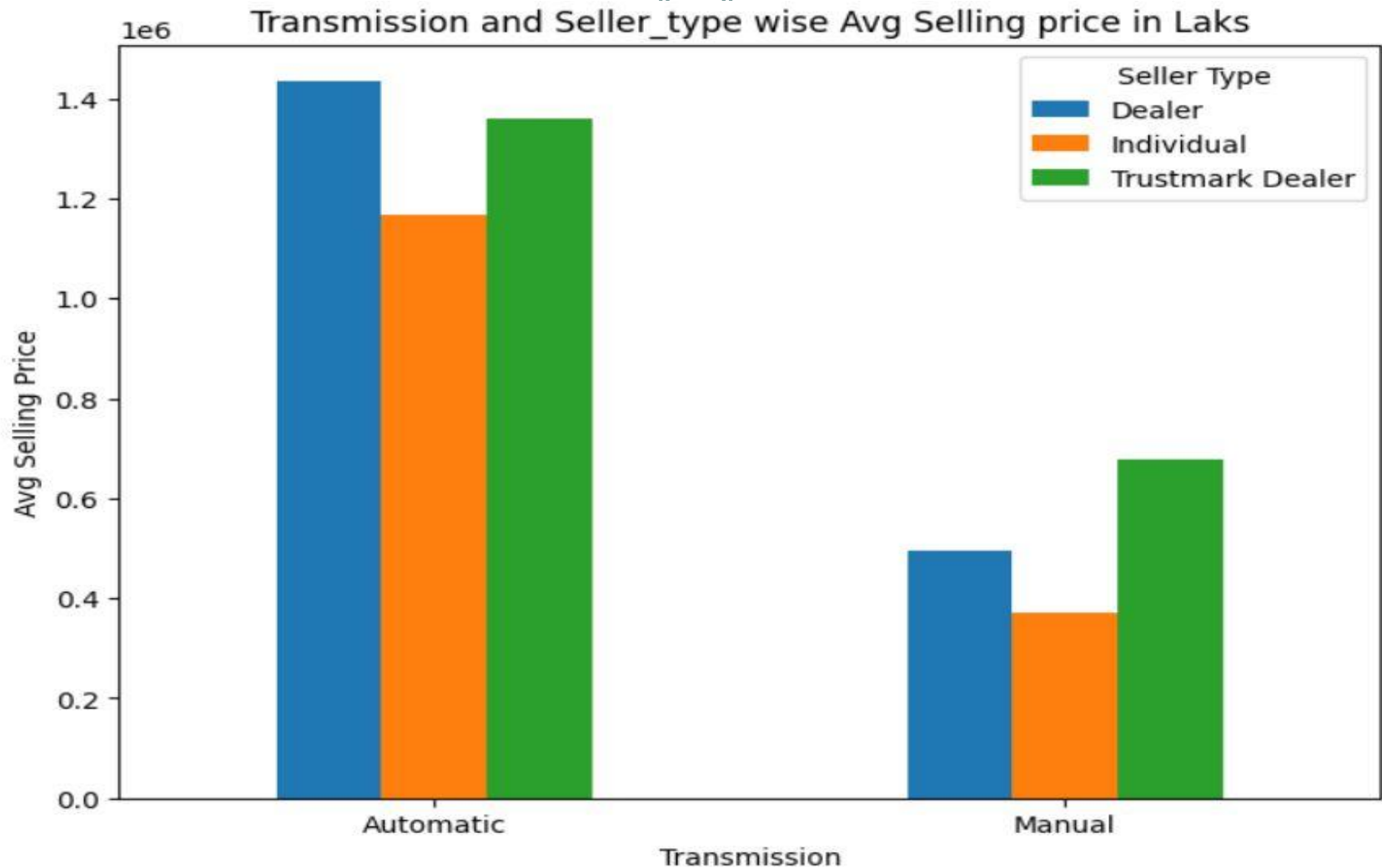
Seller Type wise Total km_driven



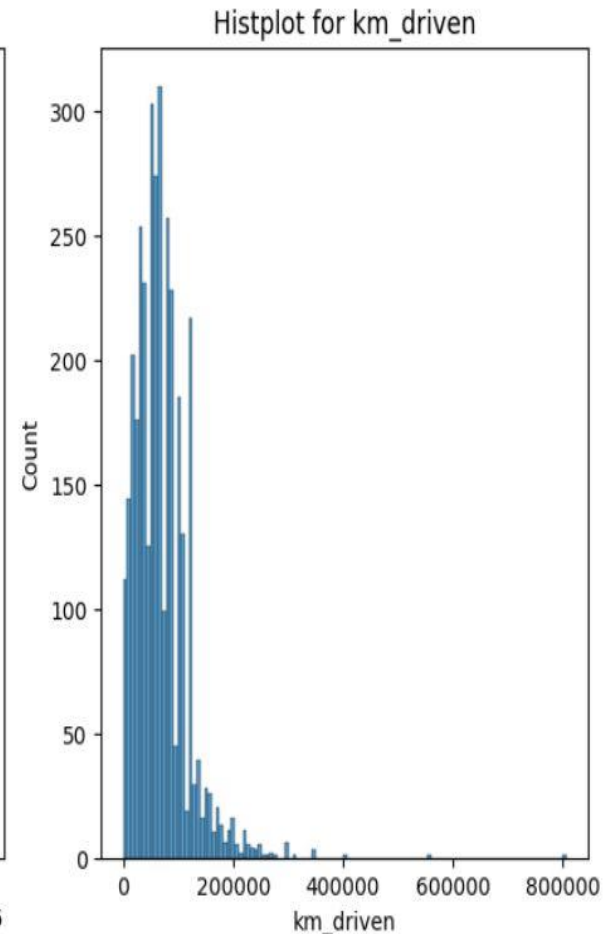
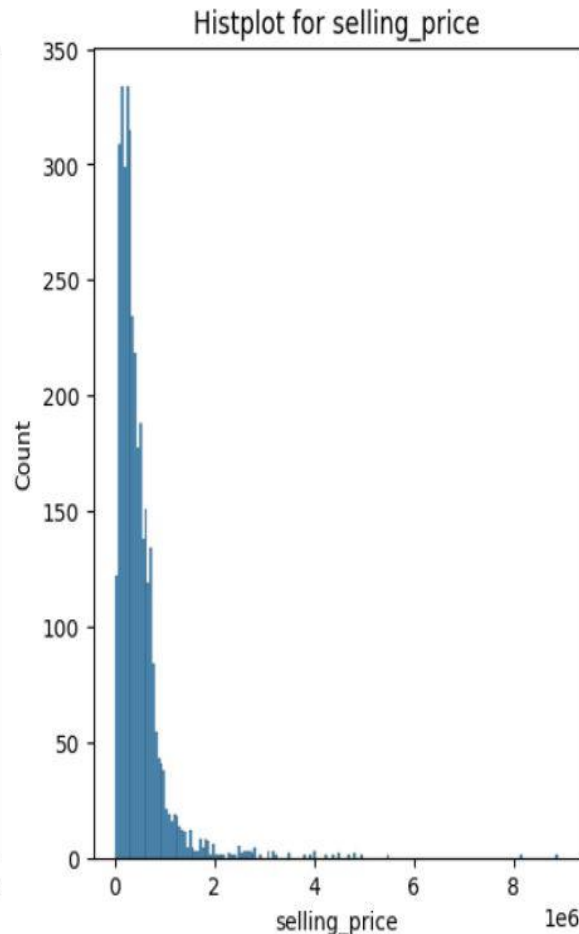
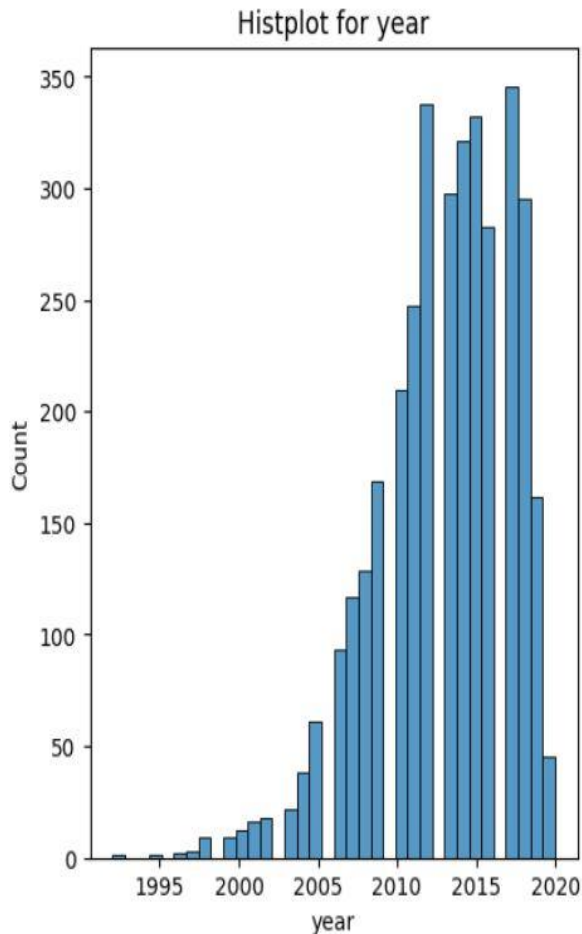
Inference: In the automatic transmission segment, prices are higher than in the manual vehicle segment. Conversely, automatic vehicles tend to be driven less compared to manual vehicles.



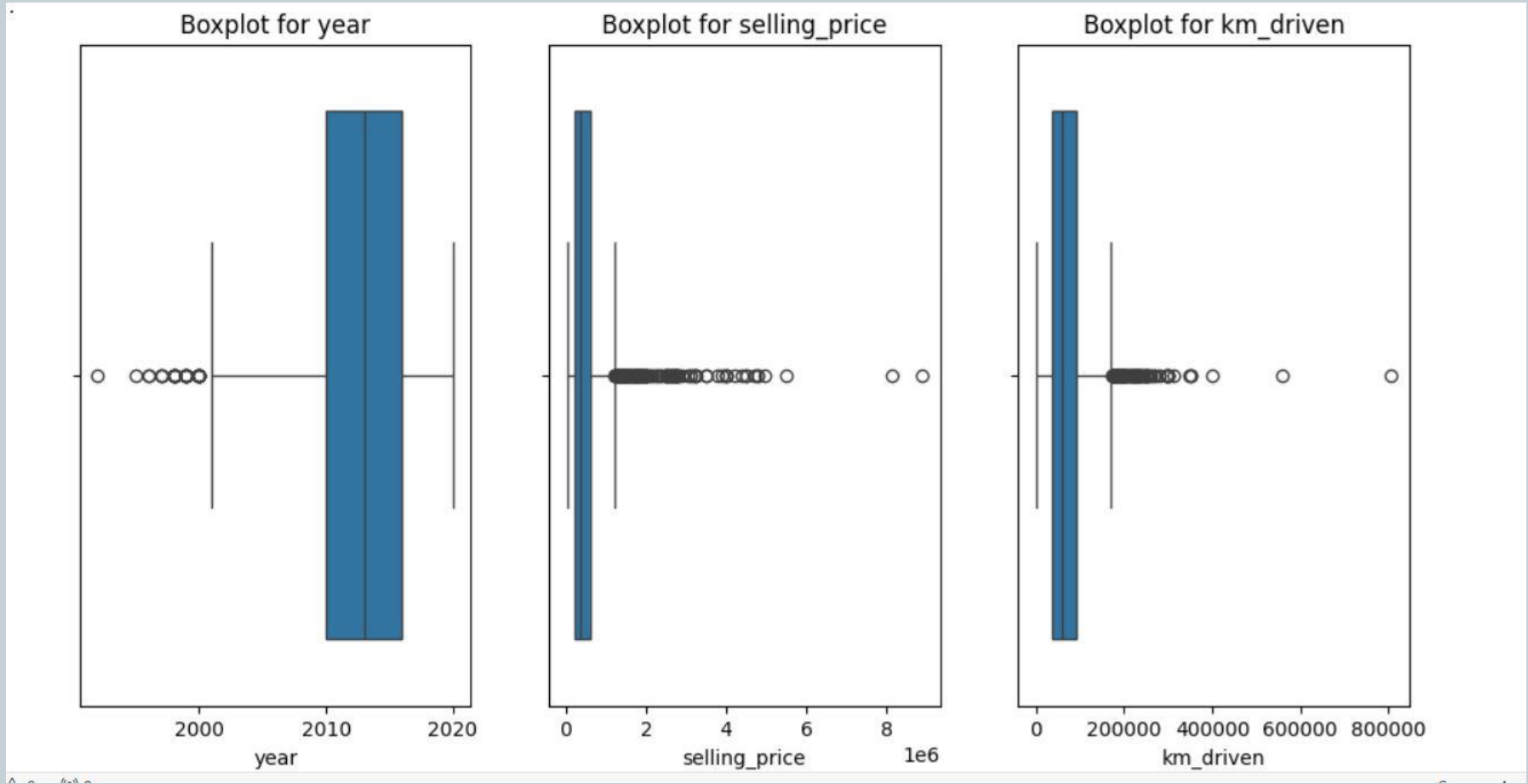
Inference: In the automatic segment, dealer prices are higher than those of Trustmark Dealers and individuals. However, in the manual segment, Trustmark Dealers have higher prices than the others.



Inference: Examining patterns over the years reveals trends in used car prices, kilometers driven, and the quantity of cars available.



Outlier Detection And Treatment



Extracting Important Features



```
# Extract Maker, model, and type features
df[['Car_Name', 'Model', 'Type']] = df['name'].str.split(n=2, expand=True)
df.head(10)
```

194] ✓ 0.1s

...

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Car_Name	Model	Type
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti	800	AC
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti	Wagon R LXI Minor	
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai	Verna	1.6 SX
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner	Datsun	RediGO	T Option
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda	Amaze	VX i-DTEC
5	Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner	Maruti	Alto	LX BSIII
6	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner	Hyundai	Xcent	1.2 Kappa S
7	Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner	Tata	Indigo	Grand Petrol
8	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner	Hyundai	Creta	1.6 VTVT S
9	Maruti Celerio Green VXI	2017	365000	78000	CNG	Individual	Manual	First Owner	Maruti	Celerio	Green VXI

Label Encoding for Model Building



```
df1=df.copy()
```

[214] ✓ 0.3s

```
#Label Encoding
```

```
cols_for_encoding = df[['year', 'seller_type', 'transmission', 'Car_Name', 'Model', 'Type', 'owner', 'fuel']]
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lb=LabelEncoder()
```

[215] ✓ 0.1s

```
for i in cols_for_encoding:  
    df[i]=lb.fit_transform(df[i])
```

[216] ✓ 0.2s

Selecting and Splitting Data



```
#Selecting x and y
x=df.drop('selling_price',axis=1)
y=df['selling_price']
print(x.shape)
print(y.shape)
```

19] ✓ 0.1s

```
.. (3576, 9)
   (3576,)
```

```
#Data Splitting
from sklearn.model_selection import train_test_split
```

20] ✓ 0.1s

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=80)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

21] ✓ 0.1s

```
.. (2503, 9)
   (1073, 9)
   (2503,)
```

Creating eval Function & Model importing



```
#Function to evaluate model performance
from sklearn.metrics import *
```

2] ✓ 0.0s

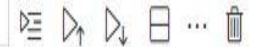
```
def eval_model(model,x_train,x_test,y_train,y_test,mname):
    model.fit(x_train,y_train)
    ypred=model.predict(x_test)
    train_score=model.score(x_train,y_train)
    test_score=model.score(x_test,y_test)
    test_mse=mean_squared_error(y_test,ypred)
    test_rmse=np.sqrt(test_mse)
    test_mae=mean_absolute_error(y_test,ypred)
    res=pd.DataFrame({'Train_Score':train_score,'Test_Score':test_score,'Test_MAE':test_mae,'Test_MSE':test_mse,
                      'Test_RMSE':test_rmse},index=[mname])
    return res,ypred
```

3] ✓ 0.0s

```
from sklearn.linear_model import LinearRegression,Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

4] ✓ 0.0s

Model Comparison



```
#Combining the results
```

```
all_res = pd.concat([Lr_res, Ri_res, DT_res, RF_res])
```

```
all_res
```

245] ✓ 0.0s

Pyt

	Train_Score	Test_Score	Test_MAE	Test_MSE	Test_RMSE
Lin_Reg	0.435565	0.472058	219581.700977	1.099936e+11	331652.800570
Ridge_Reg	0.435565	0.471936	219580.688343	1.100190e+11	331691.098879
DT_reg	0.692104	0.645613	171505.343870	7.383440e+10	271724.851856
RF_reg	0.796795	0.734676	144271.477206	5.527875e+10	235114.333128

Inference: Here, we clearly see that the Random Forest model scores are better than the other three models. Therefore, it is the best model among them.

Finalizing Model



```
RF_final=RF
RF.fit(x,y)
```

0] ✓ 1.7s

▼ RandomForestRegressor
RandomForestRegressor(max_depth=8, min_samples_split=7, n_estimators=125)

```
RF.predict(x_test)
```

1] ✓ 0.0s

```
array([744397.34317541, 656564.08563555, 927103.21603234, ...,
       484185.41270952, 118847.1589904 , 114410.70343448])
```


Generating Random Data Points and Model Testing



```
#Generating random data points
random_sample_20 = df.sample(n=20)
random_sample_20.head()
```

[6] ✓ 0.0s

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Car_Name	Model	Type
4249	5	850000	52000	1	0	1	2	12	13	7
2775	5	650000	70000	1	1	1	2	3	17	7
176	11	282000	40000	0	1	1	0	5	1	7
3911	8	900000	15000	4	1	0	0	9	13	7
1799	8	715000	25000	1	1	0	0	5	16	7

```
#Generating Predictions
print(random_sample_20.shape)
random_sample_20 = df.drop(columns=['selling_price'])
predictions = RF.predict(random_sample_20)
print(predictions)
```

[7] ✓ 0.2s

```
(20, 10)
[ 72481.23903216 117487.81664762 435639.28024352 ... 168136.63633489
 621164.29135684 328891.43822205]
```

Loading Model



```
#DataFrame and Model Saving
```

```
import pickle
```



```
pickle.dump(RF_final,open('RF_final_03-05-2024.pkl','wb'))
```


```
pickle.dump(df1,open('df1_03-05-2024.pkl','wb'))
```

```
2] ✓ 0.1s
```


File Uploading in Github Repo.




 VirendraR2107 / Capstone_Virendra_R







Type  to search

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 **Capstone_Virendra_R** Public Pin Unwatch 1

main 1 Branch 0 Tags t Add file Code

 **VirendraR2107** Add files via upload f4565ee · 2 minutes ago 17 Commits

 1.py	Add files via upload	2 minutes ago
 Capstone.ipynb	Add files via upload	8 hours ago
 RF_final_03-05-2024.pkl	Add files via upload	2 minutes ago
 df1.csv	Add files via upload	8 hours ago
 df1_03-05-2024.pkl	Add files via upload	2 minutes ago
 requirements.txt	Add files via upload	8 hours ago

Streamlit App Interface

Link:-Fill the used car details to predict the price



Used Car Price Prediction

Fill The Used Car Details to Predict The Price

Car_Name

Toyota



Model

Indigo



Type

Other



year

2017



KM(between 1.0 - 806599.0)

5000.00



GitHub



THANK YOU!