FLIP ROBO

USED CAR PRICE PREDICTION PROJECT

SUBMITTED BY –GOPABANDHU SAHOO

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Mohd Kashif Sir as well as Flip Robo Technologies who gave me the opportunity to do this project on Used Car Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things especially the data collection part.

Also, I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION
Business Problem Framing

Impact of COVID-19 on Indian automotive sector: The Indian automotive sector was already struggling in FY20. before the Covid-19 crisis. It saw an overall degrowth of nearly 18 per cent. This situation was worsened by the onset of the Covid-19 pandemic and the ongoing lockdowns across India and the rest of the world. These two years (FY20 and FY21) are challenging times for the Indian automotive sector on account of slow economic growth, negative consumer sentiment, BS-VI transition, changes to the axle load norms, liquidity crunch, low-capacity utilisation and potential bankruptcies. The return of daily life and manufacturing activity to near normalcy in China and South Korea, along with extended lockdown in India, gives hope for a U-shaped economic recovery. Our analysis indicates that the Indian automotive sector will start to see recovery in the third quarter of FY21. We expect the industry demand to be down 15-25 per cent in FY21. With such degrowth, OEMs, dealers and suppliers with strong cash reserves and better access to capital will be better positioned to sail through. Auto sector has been under pressure due to a mix of demand and supply factors. However, there are also some positive outcomes, which we shall look at.

• With India's GDP growth rate for FY21 being downgraded from 5% to 0% and later to (-5%), the auto sector will take a hit. Auto demand is highly sensitive to job creation and income levels and both have been impacted. CII has estimated the revenue impact at
$2 billion on a monthly basis across the auto industry in India.

• Supply chain could be the worst affected. Even as China recovers, supply chain disruptions are likely to last for some more time. The problems on the Indo-China border at Ladakh are not helping matters. Domestic suppliers are chipping in but they will face an inventory surplus as demand remains tepid. • The Unlock 1.0 will coincide with the implementation of the BS-VI norms and that would mean heavier discounts to dealers and also to customers. Even as auto companies are managing costs, the impact of discounts on profitability is going to be fairly steep.

• The real pain could be on the dealer end with most of them struggling with excess inventory and lack of funding options in the post COVID-19 scenario. The BS-VI price increases are also likely to hit auto demand. There are two positive developments emanating from COVID-19. The China supply chain shock is forcing major investments in the "Make in India" initiative. The COVID-19 crisis has exposed chinks in the automobile business model and it could catalyse a big move towards electric vehicles (EVs). That could be the big positive for auto sector.

Conceptual Background of the Domain Problem

The growing world of e-commerce is not just restricted to buying electronics and clothing but everything that you expect in a general store. Keeping the general store perspective aside and looking at the bigger picture, every day there are thousands or perhaps millions of deals happening in the digital marketplace. One of the most booming markets in the digital space is that of the automobile industry wherein the buying and selling of used cars take place. Sometimes we need to walk up to the dealer or individual sellers to get a used car price quote. However, buyers and sellers face a major stumbling block when it comes to their used car valuation or say their second-hand car valuation. Traditionally, you would go to a showroom and get your vehicle inspected before learning about the price. So instead of doing all these stuffs we can build a

machine learning model using different features of the used cars to predict the exact and valuable car price.

Analytical Problem Framing
Mathematical/ Analytical Modeling of the Problem

In our scrapped dataset, our target variable "Used Car Price " is a continuous variable. Therefore, we will be handling this modelling problem as regression.
This project is done in two parts:
- Data Collection phase
- Model Building phase

Data Collection phase:
You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you. More the data better the model. In this section You need to scrape the data of used cars from websites (OLX, OLA, Car Dekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data. Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback.
Model Building phase:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the below steps mentioned:

1. Data Cleaning
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing and Visualisation
4. Model Building
5. Model Evaluation
6. Selecting the best model

- **Data Sources and their formats**

In [3]: df

Out[3]:

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price
0	Hyundai	2017	2,200 km	Petrol	Manual	5,25,000
1	Hyundai	2013	91,500 km	Diesel	Manual	5,95,000
2	Ford	2017	36,000 km	Diesel	Manual	7,75,000
3	Honda	2015	90,000 km	Diesel	Manual	4,00,000
4	Maruti Suzuki	2010	40,000 km	Petrol	Manual	2,30,000
...	...	...	...	...	...	...
9995	Hyundai	2012	65,000 km	Petrol	Manual	3,25,000
9996	Maruti Suzuki	2018	85,000 km	CNG & Hybrids	Manual	2,90,000
9997	Maruti Suzuki	2010	72,000 km	Petrol	Manual	3,20,000
9998	Tata	2012	70,000 km	Diesel	Manual	1,85,000
9999	Ford	2018	53,764 km	Diesel	Manual	8,75,000

10000 rows × 6 columns

The dataset is in the form of CSV (Comma Separated Value) format and consists of 6 columns (5 features and 1 label) with 10000 number of records as explained below:
- Used Car Model - This shows the car model names

- Year of Manufacture - Gives us the year in which the car was made

- Kilometres Driven - Number of kilometres the car the driven reflecting on the Odometer

- Fuel Type - Shows the fuel type used by the vehicle

- Transmission Type - Gives us the manual or automatic gear

Shifting mechanism

- Used Car Price - Lists the selling price of the used cars

We can see our dataset includes a target label "Used Car Price" column and the remaining feature columns can be used to determine or help in predicting the price of the used cars. Since price is a continuous value it makes this to be a Regression problem!

Data Preprocessing Done

```
In [8]: missingno.bar(df, figsize = (25,5), color="tab:red")
Out[8]: <AxesSubplot:>
```
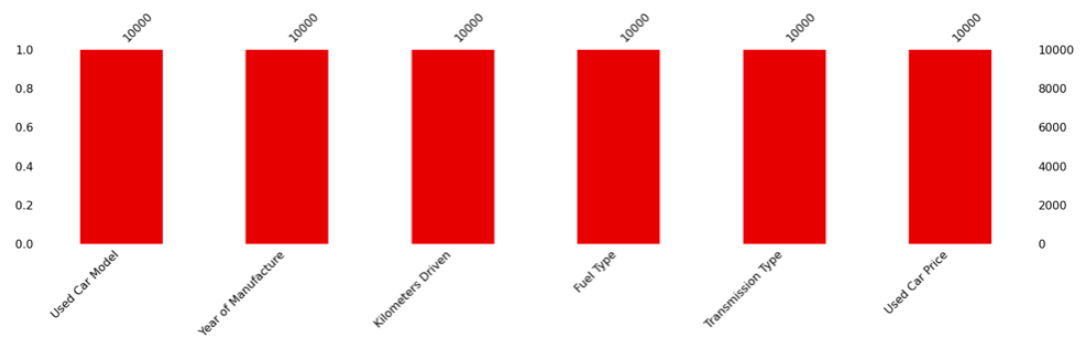
For the data pre-processing step, I checked through the dataframe for missing values, imputed records with "-" using various imputing techniques to handle them.

Checked the datatype details for each column to understand the numeric ones and its further conversion process.

```
In [9]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Used Car Model     10000 non-null  object
 1   Year of Manufacture 10000 non-null  object
 2   Kilometers Driven  10000 non-null  object
 3   Fuel Type          10000 non-null  object
 4   Transmission Type  10000 non-null  object
 5   Used Car Price     10000 non-null  object
dtypes: object(6)
memory usage: 468.9+ KB
```

I also took a look at all the unique value present in each of the columns and then decided to dealt with the imputation part accordingly.

```
In [16]: df.nunique().sort_values().to_frame("Unique Values")
Out[16]:
```

	Unique Values
Transmission Type	2
Fuel Type	5
Year of Manufacture	30
Used Car Price	751
Kilometers Driven	979
Used Car Model	2055

The various data imputation performed on our data set are shown below with the code.

I then used the "describe" method to check the count, mean, standard deviation, minimum, maximum, 25%, 50% and 75%

quartile   data.
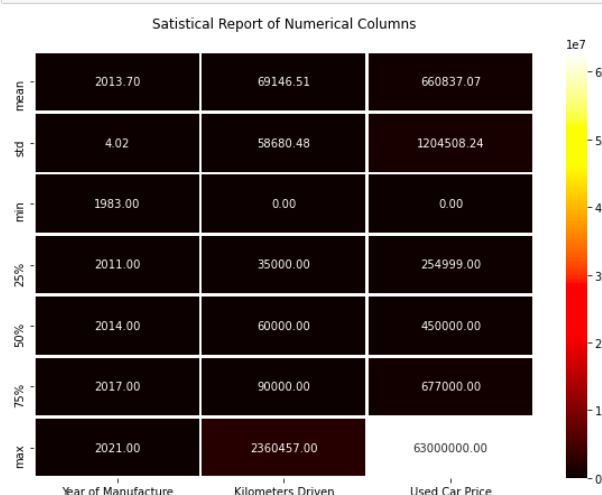
```
In [19]: df.describe(include="all")
```

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price
count	10000	10000.0000	1.000000e+04	10000	10000	1.000000e+04
unique	2055	NaN	NaN	5	2	NaN
top	Maruti Suzuki	NaN	NaN	Diesel	Manual	NaN
freq	602	NaN	NaN	5345	8598	NaN
mean	NaN	2013.69860	6.914651e+04	NaN	NaN	6.608371e+05
std	NaN	4.02124	5.868048e+04	NaN	NaN	1.204508e+06
min	NaN	1983.00000	0.000000e+00	NaN	NaN	0.000000e+00
25%	NaN	2011.00000	3.500000e+04	NaN	NaN	2.549990e+05
50%	NaN	2014.00000	6.000000e+04	NaN	NaN	4.500000e+05
75%	NaN	2017.00000	9.000000e+04	NaN	NaN	6.770000e+05
max	NaN	2021.00000	2.360457e+06	NaN	NaN	6.300000e+07

Took a visual on just the numeric part as well and saw just the maximum value for Used Car Price

```
In [21]: # visualizing the statistical description of numeric datatype columns

plt.figure(figsize = (10,7))
sns.heatmap(round(df.describe()[1:],2), linewidth = 2, annot= True, fmt = ".2f", cmap="hot")
plt.title("Satistical Report of Numerical Columns\n")
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.show()
```

Satistical Report of Numerical Columns

	Year of Manufacture	Kilometers Driven	Used Car Price
mean	2013.70	69146.51	660837.07
std	4.02	58680.48	1204508.24
min	1983.00	0.00	0.00
25%	2011.00	35000.00	254999.00
50%	2014.00	60000.00	450000.00
75%	2017.00	90000.00	677000.00
max	2021.00	2360457.00	63000000.00

• **Data Inputs- Logic- Output Relationships**

The input data were initially all object datatype so had to clean the data by removing unwanted information like "km" from Kilometres Driven column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.
Code:

Made use of Z score method to remove outliers that were present on our dataset.

```
In [36]: z = np.abs(zscore(df))
         threshold = 3
         df1 = df[(z<3).all(axis = 1)]

         print ("Shape of the dataframe before removing outliers: ", df.shape)
         print ("Shape of the dataframe after removing outliers: ", df1.shape)
         print ("Percentage of data loss post outlier removal: ", (df.shape[0]-df1.shape[0])/df.shape[0]*100)

         df=df1.copy() # reassigning the changed dataframe name to our original dataframe name

         Shape of the dataframe before removing outliers:  (10000, 6)
         Shape of the dataframe after removing outliers:  (9660, 6)
         Percentage of data loss post outlier removal:  3.4000000000000004
```

To handle the skewness, I made use of Log transformation technique ensuring that at least a bell shape curve closer to normal distribution is achieved.

• Testing of Identified Approaches (Algorithms)

```
In [1]: import warnings
        warnings.simplefilter("ignore")
        warnings.filterwarnings("ignore")
        import joblib

        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        import missingno
        from sklearn import metrics
        from scipy.stats import zscore
        from sklearn.preprocessing import OrdinalEncoder
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split

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

        from sklearn.metrics import r2_score
        from sklearn.metrics import mean_squared_error
        from sklearn.model_selection import cross_val_score
        from sklearn.model_selection import GridSearchCV
```

Libraries and Machine Learning Regression models that were used in this project are shown below.

All the regression machine learning algorithms used are:

• **Run and Evaluate selected models**
I created a Regression Machine Leaning Model function incorporating the evaluation metrics so that we can get the required

data   for   all   the   above   models.

```python
In [46]: # Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=251)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

- **Key Metrics for success in solving problem under consideration**

RMSE Score:

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

R2 Score:

The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

Cross Validation Score:

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are common tactics that you can use to select the value of k for your dataset (I have used 5-fold validation in this project). There

arecommonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.

Hyper Parameter Tuning:

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

```
In [57]: fmod_param = {'n_estimators' : [100, 200, 300],
                        'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
                        'n_jobs' : [-2, -1, 1],
                        'random_state' : [42, 251, 340]
                       }

         GSCV = GridSearchCV(ExtraTreesRegressor(), fmod_param, cv=5)
         GSCV.fit(X_train,Y_train)

Out[57]: GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
                       param_grid={'criterion': ['squared_error', 'mse', 'absolute_error',
                                                 'mae'],
                                   'n_estimators': [100, 200, 300], 'n_jobs': [-2, -1, 1],
                                   'random_state': [42, 251, 340]})
```

- **Visualizations**

I used the pandas profiling feature to generate an initial detailed report on my dataframe values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

```
In [30]: import pandas_profiling

In [31]: pandas_profiling.ProfileReport(df)
```

|		
Summarize dataset: 100%		35/35 [00:06<00:00, 6.42it/s, Completed]
Generate report structure: 100%		1/1 [00:02<00:00, 2.74s/it]
Render HTML: 100%		1/1 [00:01<00:00, 1.11s/it]

Pandas Profiling Report   Overview   Variables   Interactions   Correlations   Missing values   Sample   Duplicate rows

pandas-profiling is an open-source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and

customizations for the report generated. In short, what pandas-profiling does is save us all the work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available.

I generated count plots, bar plots, pair plots, heatmap and others to visualise the datapoint present in our column records.

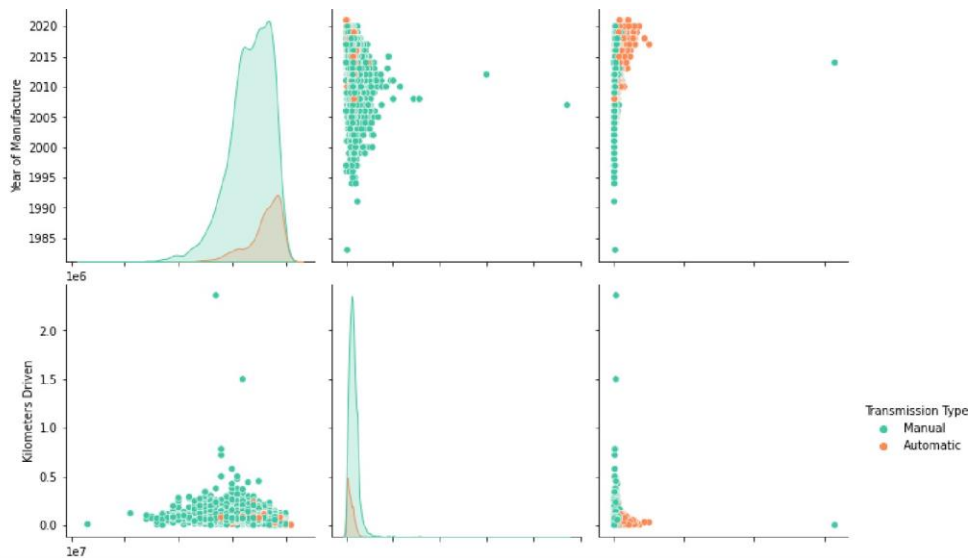Count Plot for Transmission Type column
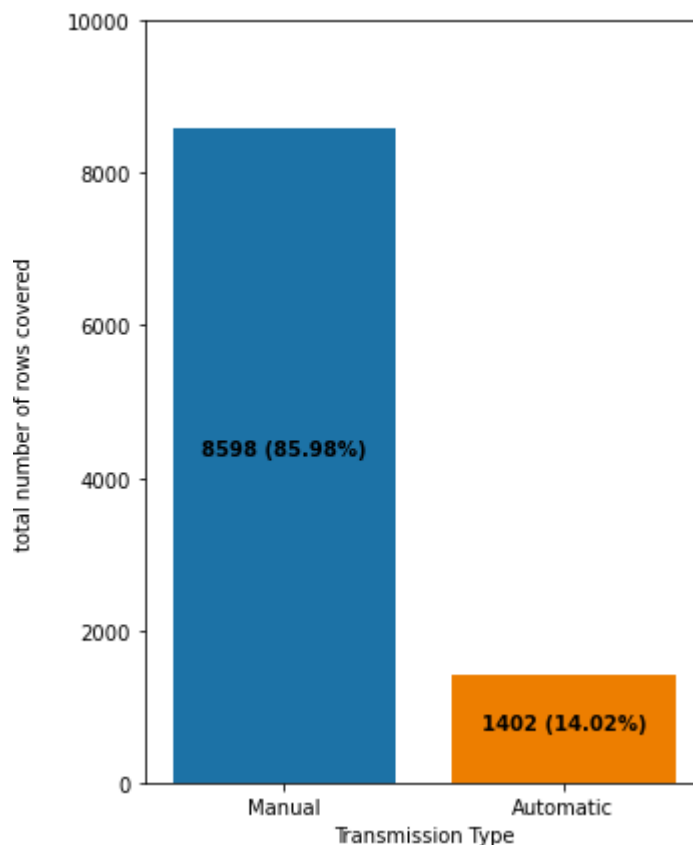
Barplot for Year of Manufacture column vs Transmission Type column
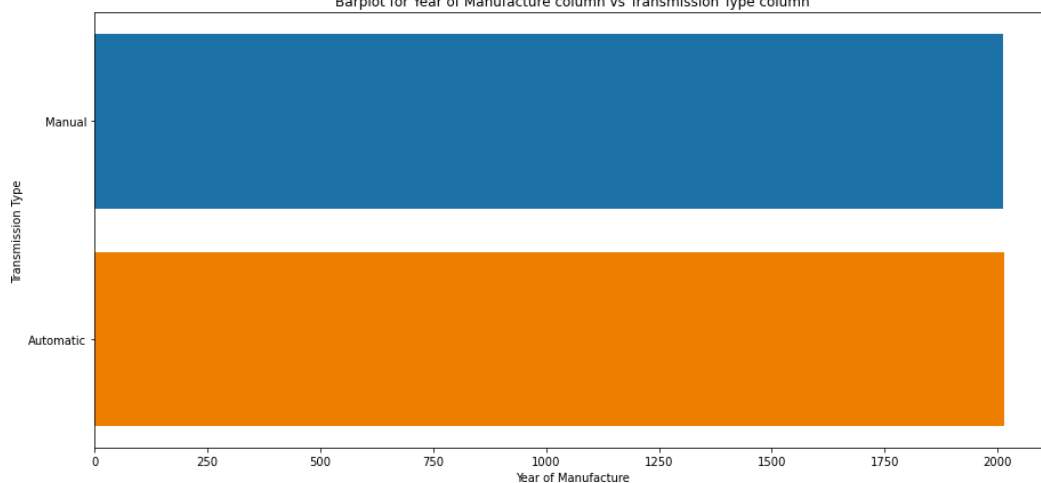
Pair P \ot uith Tran s nlss ion Type legend



```
In [27]:  Manual = df[df['Transmission Type']=='Manual']
          Automatic = df[df['Transmission Type']=='Automatic']

          print('Manual transmission type used car fuel details')
          sns.pairplot(Manual, hue='Fuel Type', diag_kind="kde", kind="scatter", palette="tab10", height=3.5)
          plt.show()

          print('Automatic transmission type used car fuel details')
          sns.pairplot(Automatic, hue='Fuel Type', diag_kind="kde", kind="scatter", palette="hls", height=3.5)
          plt.show()
```
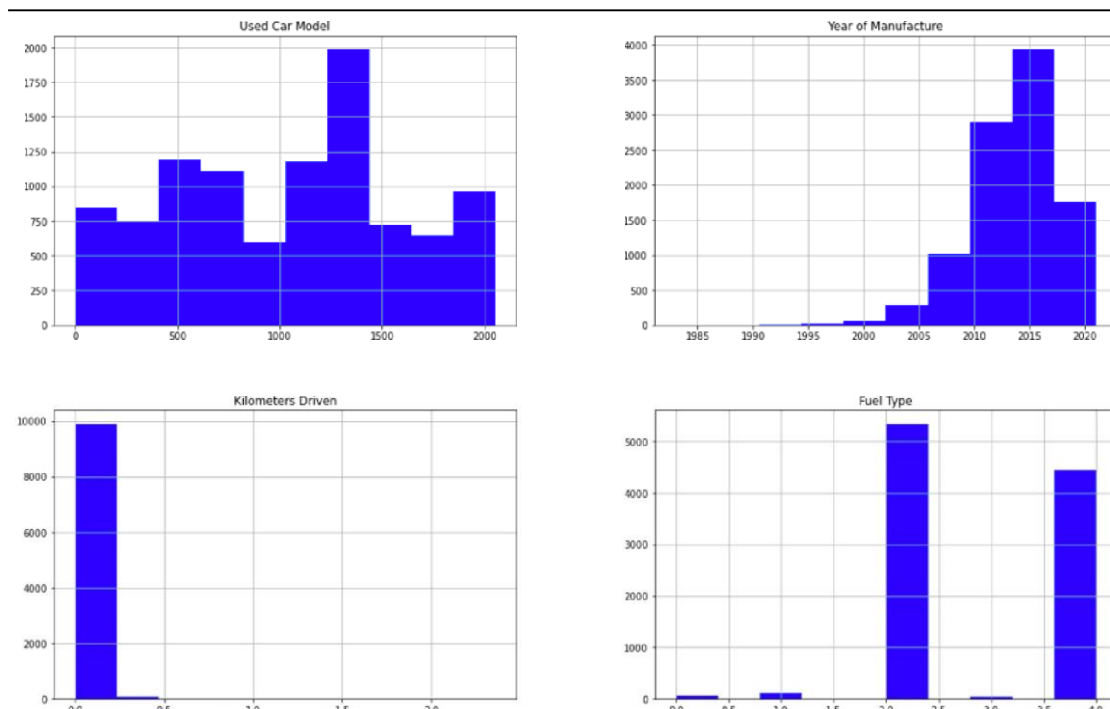
Nanua I t ransm1s s1on type used car 1' ue \ de ta 1\s
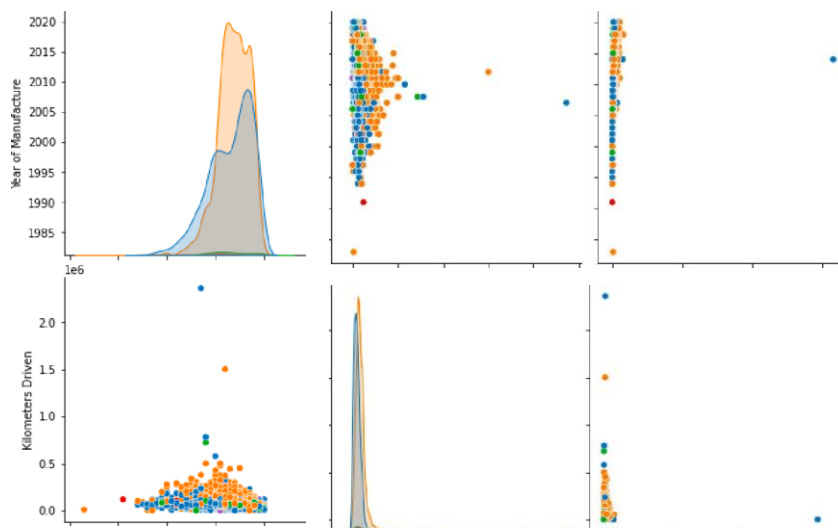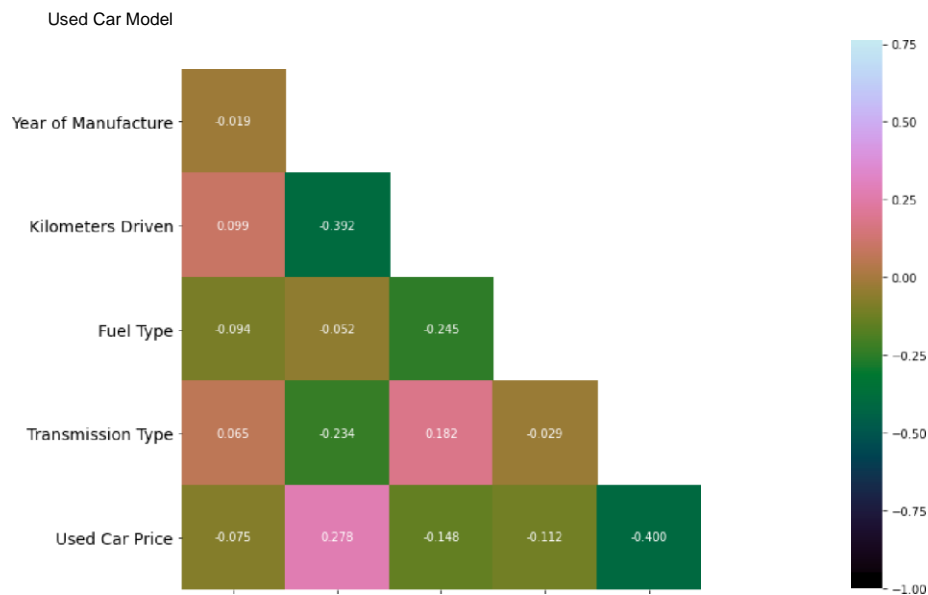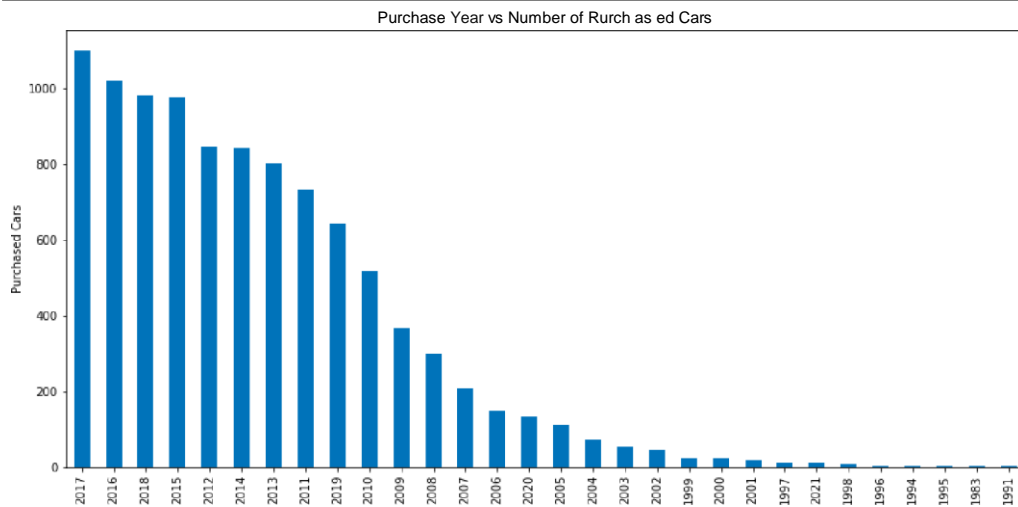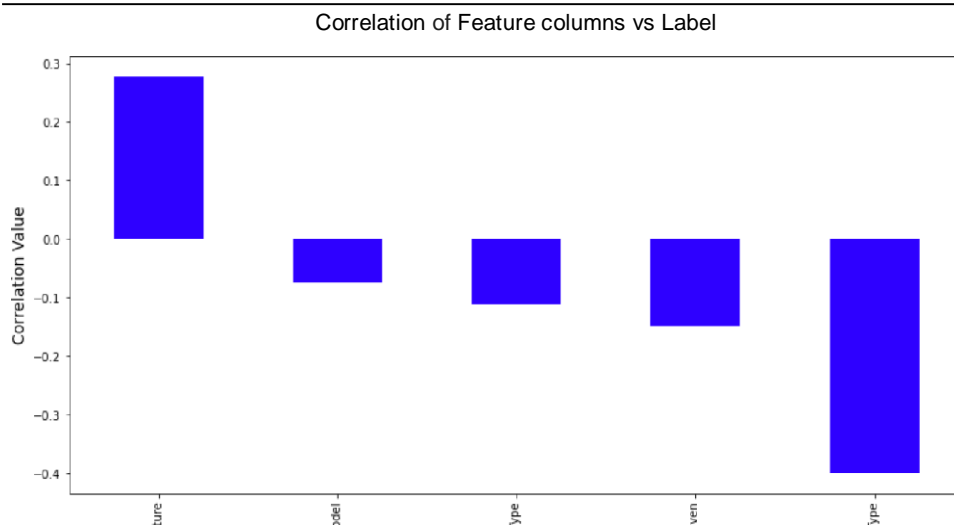
```
In [33]: upper_triangle = np.triu(df.corr())
         plt.figure(figsize=(15,10))
         sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.3f',
                     annot_kws={'size':10}, cmap="cubehelix", mask=upper_triangle)
         plt.xticks(fontsize=14)
         plt.yticks(fontsize=14)
         plt.show()
```

```
In [291] : pit.Figure(figsize=119,7])
           purchased_car_oer_year = d1'l'Year of Manulacture'].va\ue_counts()
           purchased_car_oer_year. aLot(kind='bar'!
           p\t. xlabet("\n"urelase Year")
           plt. ylabet("Purchased Cars")
           p\t. title\"Purchase Year vs Number of Purshased cars")
           p\t. s nou()
```


Purchase Year vs Number of Rurch as ed Cars

```
In [34] : df_corr = df.carr()
          p\t. 1'igure(f1gsize=(14, 7))
          df_corr['Used 0ar Prlce'].sort_vatues(ascend1ng=Fatse).droo 'Used Car Price').plot.bar()
          p\t. title\"correfation of Feature co        vs Labelsn", lonts1ze-l6J
          p\t. xlabet("\nFeatures L is l", 1'ont s ize=14)
          plt. ylabet("corre         vatue", fantsize-14}
          p\t. s nou()
```


Correlation of Feature columns vs Label

```
In f35 : plt . f1gure ( f 1gs1ze= ( 14, 7 ) )
         out1 df = df. colunns . va1ues
         fo r 1 1n range ( 0, ten ( out l df) ) :
             p1t. s ubp1ot ( 2, 3, 1+1)
             ax = sns . boxenp1ot (df l out l dl [1] ] , co to r= ' red ' )
             p1t. t1ght layout ( )
```

```
fig, ax = pit . s ubp \o t s ( ncot s=3, n rous=2, fig size= T4,7) )
index  =  0
ax = ax . l tatten ( )
fa r co l , va due 1n df . items \ ):
    s n s . boxenp l ol ( y=co l, data=df , ax=ax [ index] , co Lo r=" pu r a \e" )
    Index += 1
plt.  l ight_tayoul \ pad=0. 4,  u_pad=8. 4,  h_pad=l . B )
p \t.  s nou ( )
```



```
In [39 : plt. figure ( figs1ze= ( 14, 7) ) :
        fa r l in ra nge \ 0, \e n \ out \_df / ):
            pit.subplot(2,  3,  i+l)
            ax - sn s . dis tp\a t \ d f tout l_dt [iJ , cola r• ' blue' )
            pit.tight_layout(l
```
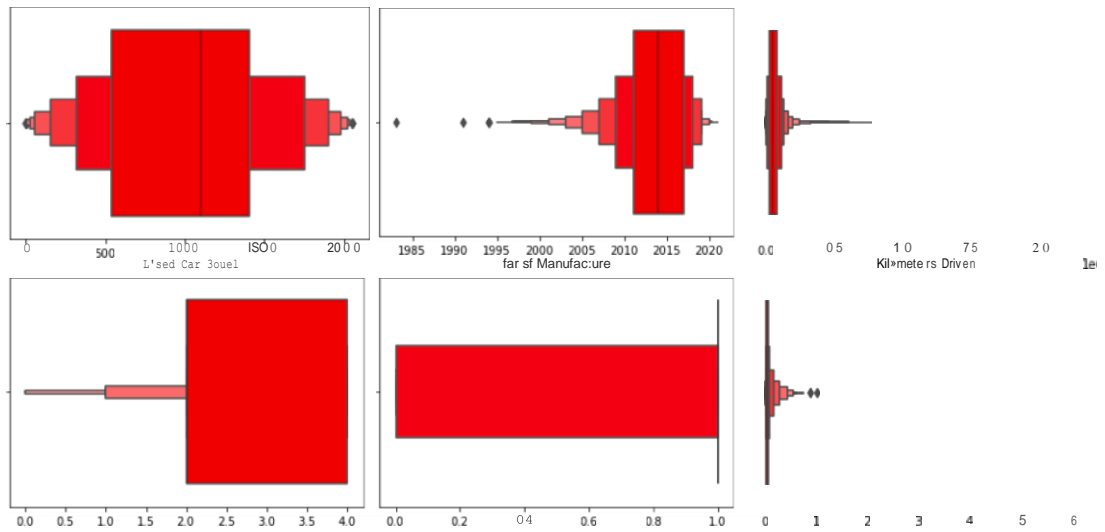
```
In [41]:  fig, ax = plt.subplots(ncols=3, nrows=2, figsize=(14,7))
index = 0
ax = ax.flatten()
for col, value in df_log.items():
    sns.distplot(value, ax=ax[index], hist=False, color="g", kde_kws={"shade": True})
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.4, h_pad=1.0)
plt.show()
```
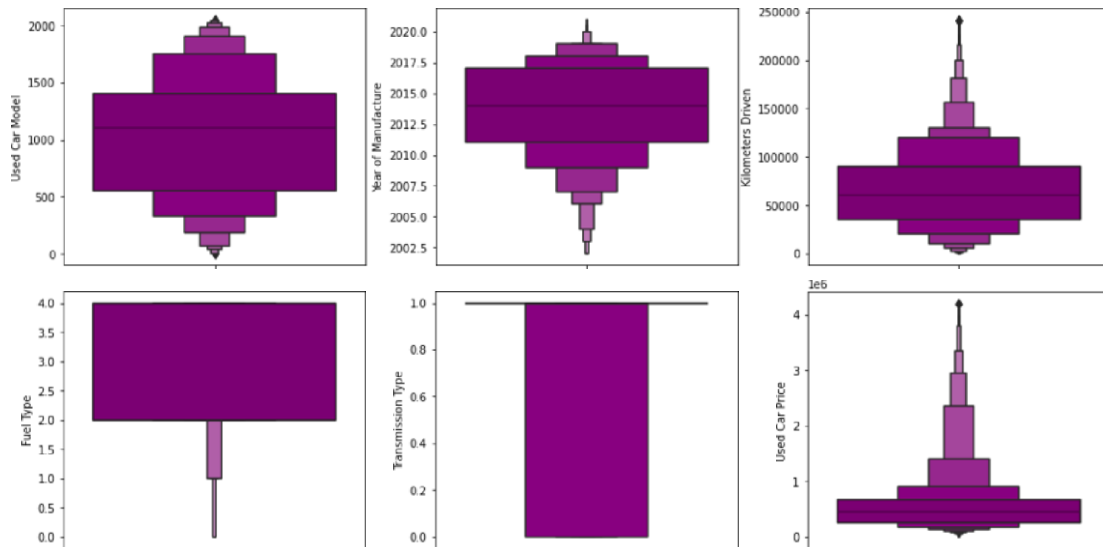

Fuel Type   0.064
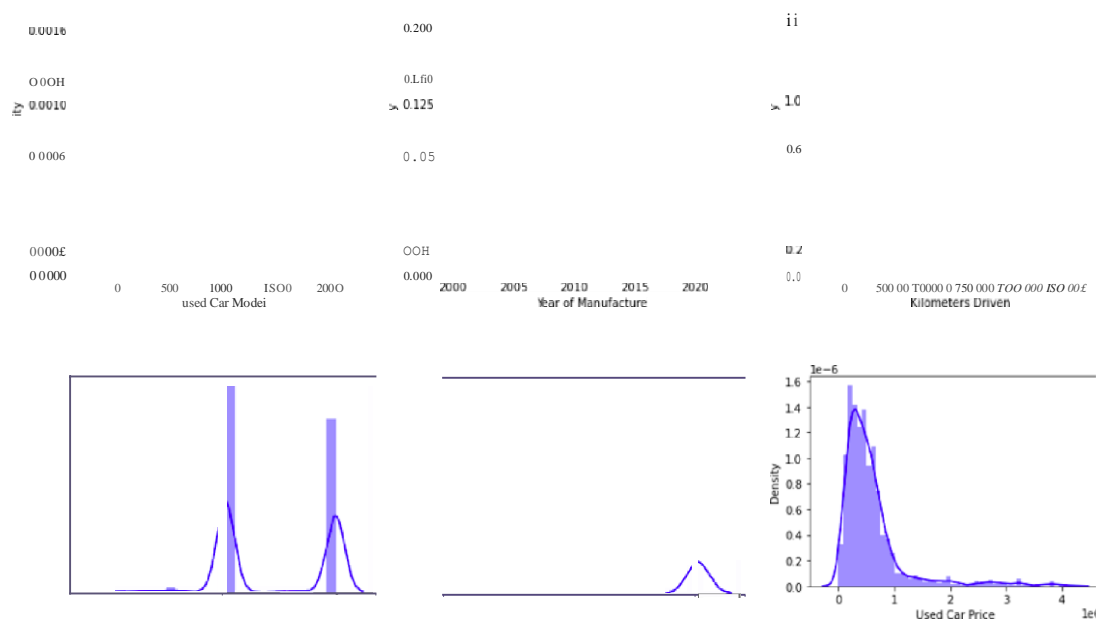
• **Interpretation of the Results**

We can see from the visuals that the features are impacting the price of used cars. There were categorical columns which I encoded using the ordinal encoder method instead of the one hot encoding to avoid the generation of large number of columns. Also, I our target label stored continuous numeric data and therefore label encoder was out of the picture to be used.

CONCLUSION
Key Findings and Conclusions of the Study

After the completion of this project, we got an insight on how to collect data, pre-processing the data, analysing the data and building a model. First, we collected the used cars data from different websites like OLX, Car Dekho, Cars 24, OLA etc and it was done by using Web Scraping. The framework used for web scraping was Beautiful Soup and Selenium, which has an advantage of automating our process of collecting data. We collected almost 10000 of data which contained the selling price and other related features of used cars. Then the scrapped data was combined in a single data frame and saved in a csv file so that we can open it and analyse the data. We did data cleaning, data pre-processing steps like finding and handling null values, removing words from numbers, converting object to int type, data visualization, handling outliers and skewness etc. After separating our train and test data, we started running different machine learning regression algorithms to find out the best performing model. We found that Extra Tree Regressor Algorithm was performing well according to their r2_score and cross validation scores. Then we performed Hyperparameter Tuning technique using Grid Search CV for getting the best parameters and improving the score. In that Extra Tree Regressor Algorithm did not perform quite well as previously on the defaults but we finalised that model for further predictions as it was still better than the rest. We saved the final model in pkl format using the joblib library after getting a dataframe of predicted and actual used car price details.

• Learning Outcomes of the Study in respect of Data Science

Visualization part helped me to understand the data as it provides graphical representation of huge data. It assisted me to understand the feature importance, outliers/skewness detection and tocompare the independent-dependent features. Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned and scaled. I have

generated multiple regression machine learning models to get the best model wherein I found Extra Trees Regressor Model being the best based on the metrics I have used.

• Limitations of this work and Scope for Future Work

The limitations we faced during this project were:

The website was poorly designed because the scrapping took a lot of time and there were many issues in accessing to next page. Also need further practise in terms of various web scraping techniques. More negative correlated data were present than the positive correlated one's. Presence of outliers and skewness were detected and while dealing with them we had to lose a bit of valuable data. No information for handling these fast-paced websites were provided so that was consuming more time in web scraping part.