Data Collection and Preprocessing Phase

Date	15 July 2024
Team ID	739956
Project Title	Revolutionizing Automotive Resale: AI-Driven Prediction of Used Toyota Corolla Car Prices
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	**Description**

Data Overview	<u>Dimension:</u>
	1436 rows ,39 columns
	[6] `df.shape`
	`(1436, 39)`
	`df.describe()`
	8 rows × 36 columns

Univariate Analysis	-
Bivariate Analysis	-
Multivariate Analysis	-
Outliers and Anomalies	

	Id	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	CC	...	Powered_Windows	Power_Steering	Radio
count	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.00000	...	1436.000000	1436.000000	1436.000000
mean	721.555014	10730.824513	55.947075	5.546747	1999.625348	68533.259749	101.502089	0.674791	0.055710	1576.85585	...	0.561978	0.977716	0.146240
std	416.476890	3626.964585	18.599988	3.354085	1.540722	37506.448872	14.981080	0.468616	0.229441	424.38677	...	0.496317	0.147657	0.353469
min	1.000000	4350.000000	1.000000	1.000000	1998.000000	1.000000	69.000000	0.000000	0.000000	1300.00000	...	0.000000	0.000000	0.000000
25%	361.750000	8450.000000	44.000000	3.000000	1998.000000	43000.000000	90.000000	0.000000	0.000000	1400.00000	...	0.000000	1.000000	0.000000
50%	721.500000	9900.000000	61.000000	5.000000	1999.000000	63389.500000	110.000000	1.000000	0.000000	1600.00000	...	1.000000	1.000000	0.000000
75%	1081.250000	11950.000000	70.000000	8.000000	2001.000000	87020.750000	110.000000	1.000000	0.000000	1600.00000	...	1.000000	1.000000	0.000000
max	1442.000000	32500.000000	80.000000	12.000000	2004.000000	243000.000000	192.000000	1.000000	1.000000	16000.00000	...	1.000000	1.000000	1.000000

```
z_score_threshold = 3

outliers_df= pd.DataFrame (columns=['Column', 'Number of Outliers'])
for column in df.select_dtypes(include=['int64', 'int32']).columns:
    z_scores = np.abs((df[column] - df[column].mean()) /df[column].std())
    outliers = z_scores > z_score_threshold
    outliers_count= outliers.sum()

    outliers_df = pd.concat([outliers_df, pd.DataFrame({'Column': [column], 'Number of Outliers': [outliers_count]})], ignore_index=True)

print(outliers_df)
```

```
[10] df['Fuel_Type'] = df['Fuel_Type'].astype('category').cat.codes
     df['Fuel_Type']
```

```
0       1
1       1
2       1
3       1
4       1
       ..
1431    2
1432    2
1433    2
1434    2
1435    2
Name: Fuel_Type, Length: 1436, dtype: int8
```

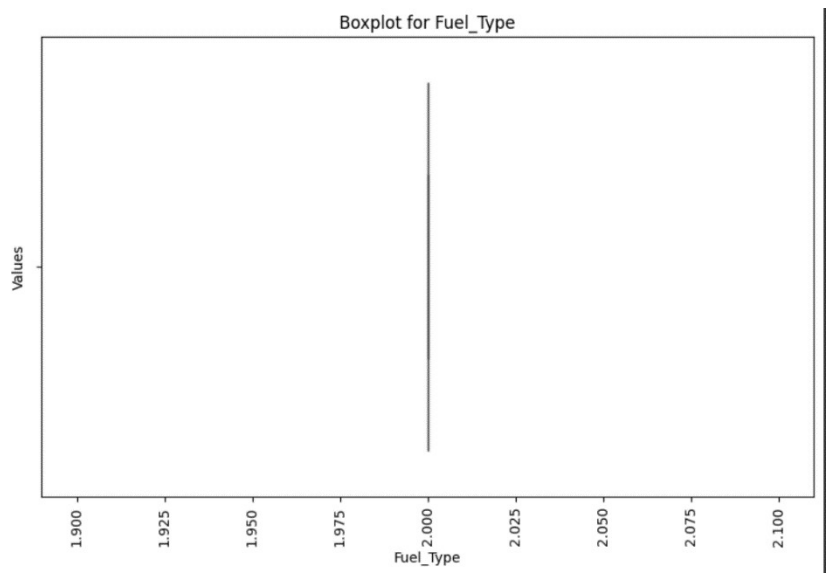
Handling Outliers For Fuel_type column

```
quant1 = df.Fuel_Type.quantile([0.75,0.25])
Q3_Fuel=quant1.loc[0.75]
Q1_Fuel=quant1.loc[0.25]
IQR_Fuel = Q3_Fuel - Q1_Fuel
maxwhisker_Fuel = Q3_Fuel + 1.5 * IQR_Fuel
print("maxwhisker_Fuel:", maxwhisker_Fuel)
minwhisker_Fuel = Q1_Fuel - 1.5 * IQR_Fuel
print("minwhisker_Fuel:", minwhisker_Fuel)
```

```
maxwhisker_Fuel: 2.0
minwhisker_Fuel: 2.0
```

```
[12] df['Fuel_Type'] = np.where(df.Fuel_Type<minwhisker_Fuel,minwhisker_Fuel,df.Fuel_Type)
     df['Fuel_Type']
```

```
0       2.0
1       2.0
2       2.0
3       2.0
4       2.0
       ...
1431    2.0
1432    2.0
1433    2.0
1434    2.0
1435    2.0
Name: Fuel_Type, Length: 1436, dtype: float64
```



Handling Outliers For Weight Column

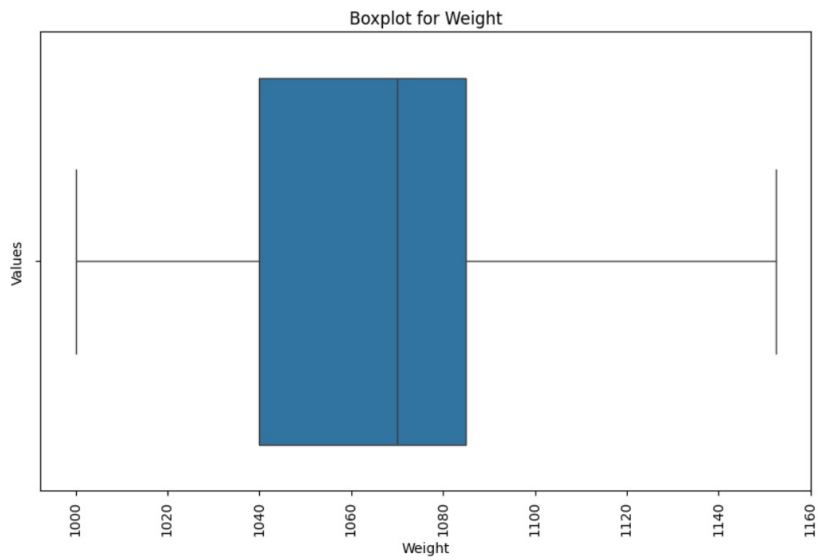
```
[14] quant2 = df.Weight.quantile([0.75,0.25])
     Q3_weight=quant2.loc[0.75]
     Q1_weight=quant2.loc[0.25]
     IQR_weight = Q3_weight - Q1_weight
     maxwhisker_weight = Q3_weight + 1.5 * IQR_weight
     print("maxwhisker_weight:", maxwhisker_weight)
     minwhisker_weight = Q1_weight - 1.5 * IQR_weight
     print("minwhisker_weight:", minwhisker_weight)
```

```
maxwhisker_weight: 1152.5
minwhisker_weight: 972.5
```

```
df['Weight'] = np.where(df.Weight>maxwhisker_weight,maxwhisker_weight,df.Weight)
df['Weight']
```

```
0       1152.5
1       1152.5
2       1152.5
3       1152.5
4       1152.5
         ...
1431    1025.0
1432    1015.0
1433    1015.0
1434    1015.0
1435    1114.0
Name: Weight, Length: 1436, dtype: float64
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Weight', data=df)
plt.title('Boxplot for Weight')
plt.xlabel('Weight')
plt.ylabel('Values')
plt.xticks(rotation=90) # Rotate x-axis labels for better visibility
plt.show()
```



Handling Outliers For CC Column

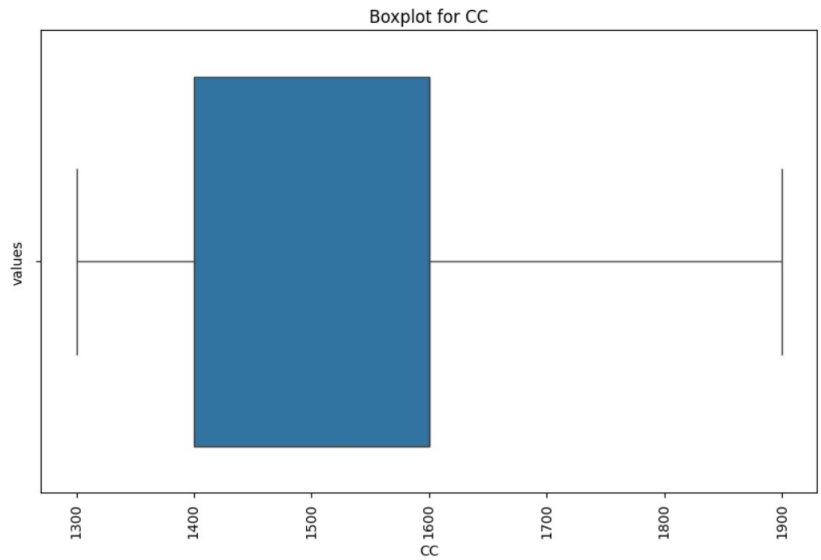
```
[14] quant3 = df.CC.quantile([0.75,0.25])
     Q3_CC=quant3.loc[0.75]
     Q1_CC=quant3.loc[0.25]
     IQR_CC = Q3_CC - Q1_CC
     maxwhisker_CC = Q3_CC + 1.5 * IQR_CC
     print("maxwhisker_CC:",maxwhisker_CC)
     minwhisker_CC = Q1_CC - 1.5 * IQR_CC
     print("minwhisker_CC:",minwhisker_CC)
```

```
maxwhisker_CC: 1900.0
minwhisker_CC: 1100.0
```

```
[15] df['CC'] = np.where(df.CC>maxwhisker_CC,maxwhisker_CC,df.CC)
     df['CC']
```

```
0       1900.0
1       1900.0
2       1900.0
3       1900.0
4       1900.0
        ...
1431    1300.0
1432    1300.0
1433    1300.0
1434    1300.0
1435    1600.0
Name: CC, Length: 1436, dtype: float64
```

```
[16] plt.figure(figsize=(10,6))
     sns.boxplot(x='CC', data=df)
     plt.title('Boxplot for CC')
     plt.xlabel('CC')
     plt.ylabel('values')
     plt.xticks(rotation=90) # Rotate x-axis labels for better visibility
     plt.show()
```



Handling Outliers For KM Column

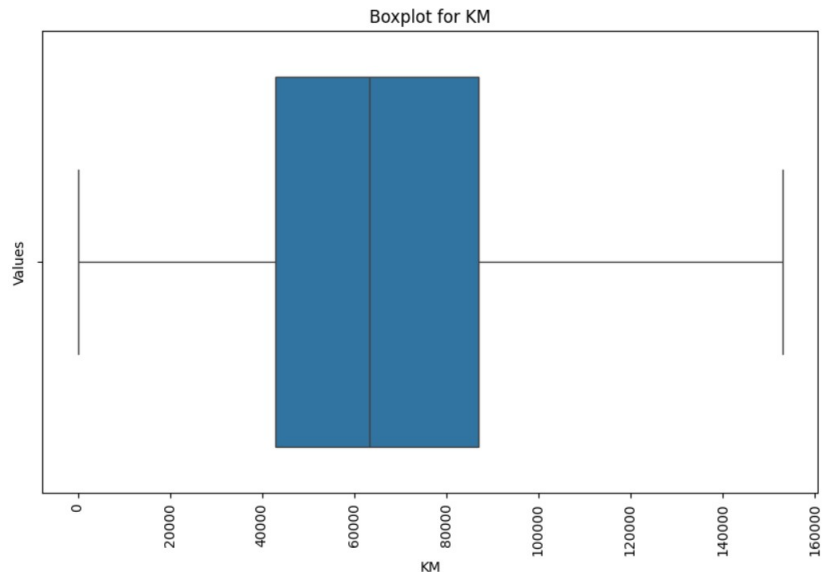
```
[17] quant4 = df.KM.quantile([0.75,0.25])
     Q3_KM=quant4.loc[0.75]
     Q1_KM=quant4.loc[0.25]
     IQR_KM = Q3_KM - Q1_KM
     maxwhisker_KM = Q3_KM + 1.5 * IQR_KM
     print("maxwhisker_KM:", maxwhisker_KM)
     minwhisker_KM = Q1_KM - 1.5 * IQR_KM
     print("minwhisker_KM:",minwhisker_KM)
```

```
maxwhisker_KM: 153051.875
minwhisker_KM: -23031.125
```

```
[18] df['KM'] = np.where(df.KM>maxwhisker_KM,maxwhisker_KM,df.KM)
     df['KM']
```

```
0       46986.0
1       72937.0
2       41711.0
3       48000.0
4       38500.0
        ...
1431    20544.0
1432    19000.0
1433    17016.0
1434    16916.0
1435        1.0
Name: KM, Length: 1436, dtype: float64
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='KM', data=df)
plt.title('Boxplot for KM')
plt.xlabel('KM')
plt.ylabel('Values')
plt.xticks(rotation=90) # Rotate x-axis labels for better visibility
plt.show()
```



Handling Outliers For price Column

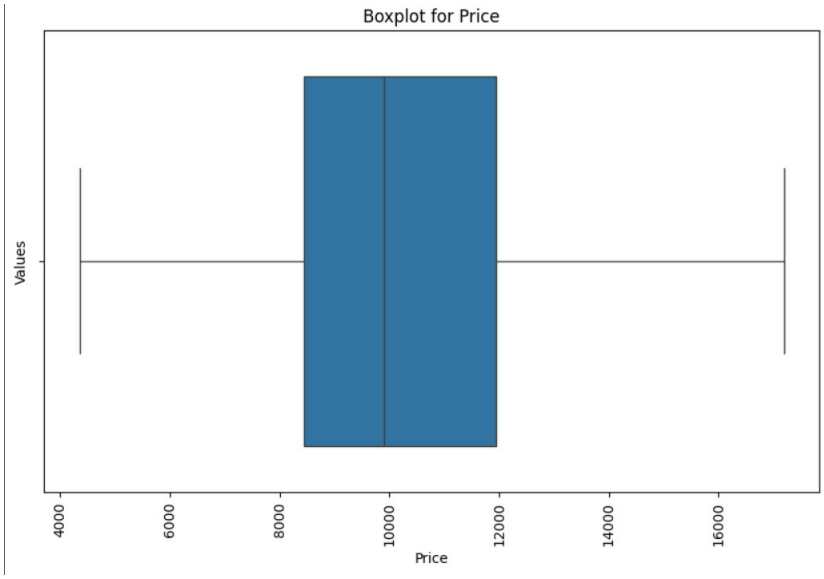
```
[20] quant5 = df.Price.quantile([0.75,0.25])
     Q3_Price=quant5.loc[0.75]
     Q1_Price=quant5.loc[0.25]
     IQR_Price = Q3_Price - Q1_Price
     maxwhisker_Price = Q3_Price + 1.5 * IQR_Price
     print("maxwhisker_Price:", maxwhisker_Price)
     minwhisker_Price = Q1_Price - 3.5* IQR_Price
     print("minwhisker_Price:", minwhisker_Price)
```

```
maxwhisker_Price: 17200.0
minwhisker_Price: -3800.0
```

```
df['Price'] = np.where(df.Price>maxwhisker_Price,maxwhisker_Price,df.Price)
df['Price']
```

```
0       13500.0
1       13750.0
2       13950.0
3       14950.0
4       13750.0
         ...
1431     7500.0
1432    10845.0
1433     8500.0
1434     7250.0
1435     6950.0
Name: Price, Length: 1436, dtype: float64
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Price', data=df)
plt.title('Boxplot for Price')
plt.xlabel('Price')
plt.ylabel('Values')
plt.xticks(rotation=90) # Rotate x-axis labels for better visibility
plt.show()
```



Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv("/content/ToyotaCorolla.csv")
df
```

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	...	Powered_Windows	Power_Steering	Radio	Mistlamps	Sport_Model	Backseat_Divider	Me
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	1	...	1	1	0	0	0	1	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	1	...	0	1	0	0	0	1	
2	3	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	1	...	0	1	0	0	0	1	
3	4	TOYOTA Corolla 2.0 D4D HATCHB	14950	26	7	2002	48000	Diesel	90	0	...		1	0	0	0	1	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	0	...	1	1	0	1	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	12	1998	20544	Petrol	86	1	...	1	1	0	1	1	1	
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	9	1998	19000	Petrol	86	0	...	0	1	0	0	1	1	
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	10	1998	17016	Petrol	86	0	...	0	1	0	0	0	1	

<table>
<tr><td></td><td>

1432	1438	Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	9	1998	19000	Petrol	86	0	...	0	1	0	0	1	1
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	10	1998	17016	Petrol	86	0	...	0	1	0	0	0	1
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	11	1998	16916	Petrol	86	1	...	0	0	0	0	0	1
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	5	1998	1	Petrol	110	0	...	0	1	0	0	0	0

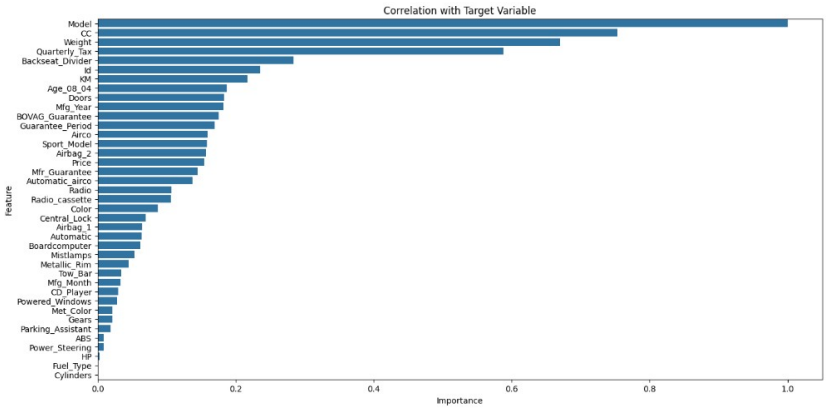
1436 rows × 39 columns

</td></tr>
<tr><td>Handling Missing Data</td><td>-</td></tr>
<tr><td>Data Transformation</td><td>-</td></tr>
<tr><td>Feature Engineering</td><td>

```python
#Assuming your DataFrame is named 'df' and you want to predict 'Model' as an example
correlations= df.corr() ['Model'].abs().sort_values(ascending=False)

feature_importance_df = pd.DataFrame({'Feature': correlations.index, 'Importance': correlations.values})

plt.figure(figsize=(16, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title("Correlation with Target Variable")
plt.show()
```



</td></tr>
<tr><td>Save Processed Data</td><td>-</td></tr>
</table>