In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:
```
data=pd.read_excel('ToyotaCorolla.xlsx',sheet_name='Data')
```

In [14]:
```
data.head()
```

Out[14]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	...	Powered_Windows	Power_Steering	Radio	Mistlamps	Sp
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	1	...	1	1	0	0	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	1	...	0	1	0	0	
2	3	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	1	...	0	1	0	0	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	Diesel	90	0	...	0	1	0	0	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	0	...	1	1	0	1	

5 rows × 39 columns

In [15]:
```
data.shape
```

Out[15]:
```
(1436, 39)
```

In [16]:
```
data.dtypes
```

Out[16]:
```
Id                  int64
Model              object
Price               int64
Age_08_04           int64
Mfg_Month           int64
Mfg_Year            int64
KM                  int64
Fuel_Type          object
HP                  int64
Met_Color           int64
Color              object
Automatic           int64
CC                  int64
Doors               int64
Cylinders           int64
Gears               int64
Quarterly_Tax       int64
Weight              int64
Mfr_Guarantee       int64
BOVAG_Guarantee     int64
Guarantee_Period    int64
ABS                 int64
Airbag_1            int64
Airbag_2            int64
Airco               int64
Automatic_airco     int64
Boardcomputer       int64
CD_Player           int64
Central_Lock        int64
Powered_Windows     int64
Power_Steering      int64
Radio               int64
Mistlamps           int64
Sport_Model         int64
Backseat_Divider    int64
Metallic_Rim        int64
Radio_cassette      int64
Parking_Assistant   int64
Tow_Bar             int64
dtype: object
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 39 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Id                 1436 non-null   int64
 1   Model              1436 non-null   object
 2   Price              1436 non-null   int64
 3   Age_08_04          1436 non-null   int64
 4   Mfg_Month          1436 non-null   int64
 5   Mfg_Year           1436 non-null   int64
 6   KM                 1436 non-null   int64
 7   Fuel_Type          1436 non-null   object
 8   HP                 1436 non-null   int64
 9   Met_Color          1436 non-null   int64
 10  Color              1436 non-null   object
 11  Automatic          1436 non-null   int64
 12  CC                 1436 non-null   int64
 13  Doors              1436 non-null   int64
 14  Cylinders          1436 non-null   int64
 15  Gears              1436 non-null   int64
 16  Quarterly_Tax      1436 non-null   int64
 17  Weight             1436 non-null   int64
 18  Mfr_Guarantee      1436 non-null   int64
 19  BOVAG_Guarantee    1436 non-null   int64
 20  Guarantee_Period   1436 non-null   int64
 21  ABS                1436 non-null   int64
 22  Airbag_1           1436 non-null   int64
 23  Airbag_2           1436 non-null   int64
 24  Airco              1436 non-null   int64
 25  Automatic_airco    1436 non-null   int64
 26  Boardcomputer      1436 non-null   int64
 27  CD_Player          1436 non-null   int64
 28  Central_Lock       1436 non-null   int64
 29  Powered_Windows    1436 non-null   int64
 30  Power_Steering     1436 non-null   int64
 31  Radio              1436 non-null   int64
 32  Mistlamps          1436 non-null   int64
 33  Sport_Model        1436 non-null   int64
 34  Backseat_Divider   1436 non-null   int64
 35  Metallic_Rim       1436 non-null   int64
 36  Radio_cassette     1436 non-null   int64
 37  Parking_Assistant  1436 non-null   int64
 38  Tow_Bar            1436 non-null   int64
dtypes: int64(36), object(3)
memory usage: 437.7+ KB
```

In [ ]:
```
```

In [20]:
```
newdata=data.drop('Model',axis=1)
newdata.head()
```

Out[20]:

	Id	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	Color	...	Powered_Windows	Power_Steering	Radio	Mistlamps	Spor
0	1	13500	23	10	2002	46986	Diesel	90	1	Blue	...	1	1	0	0	
1	2	13750	23	10	2002	72937	Diesel	90	1	Silver	...	0	1	0	0	
2	3	13950	24	9	2002	41711	Diesel	90	1	Blue	...	0	1	0	0	
3	4	14950	26	7	2002	48000	Diesel	90	0	Black	...	0	1	0	0	
4	5	13750	30	3	2002	38500	Diesel	90	0	Black	...	1	1	0	1	

5 rows × 38 columns

In [21]:
```
#applying categorical encoding for columnsfuel_type and color
```

In [23]:
```
newdata_encoded=pd.get_dummies(newdata,columns=['Fuel_Type','Color'],drop_first=True,dtype=int)
```

In [24]:
```
newdata_encoded.shape
```

Out[24]:
```
(1436, 47)
```

In [26]:
```
newdata1=newdata_encoded.drop('Id',axis=1)
newdata1.head()
```

Out[26]:

	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	CC	Doors	...	Fuel_Type_Petrol	Color_Black	Color_Blue	Color_Green
0	13500	23	10	2002	46986	90	1	0	2000	3	...	0	0	1	0
1	13750	23	10	2002	72937	90	1	0	2000	3	...	0	0	0	0
2	13950	24	9	2002	41711	90	1	0	2000	3	...	0	0	1	0
3	14950	26	7	2002	48000	90	0	0	2000	3	...	0	1	0	0
4	13750	30	3	2002	38500	90	0	0	2000	3	...	0	1	0	0

5 rows × 46 columns

In [27]:
```
# price is dependent variuable, target varible and others are independent and features variable for that we use PCA ...how to
```

In [34]:
```
newdata1.shape
```

Out[34]: (1436, 46)

In [38]:
```
target_df=pd.DataFrame(newdata1.iloc[:,0])
#above commands separate target column into new data frame

feature_df=newdata1.drop('Price',axis=1)
#above commands remove first column and keeps all other columns
```

In [39]:
```
target_df.head()
```

Out[39]:

	Price
0	13500
1	13750
2	13950
3	14950
4	13750

In [40]:
```
feature_df.head()
```

Out[40]:

	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	CC	Doors	Cylinders	...	Fuel_Type_Petrol	Color_Black	Color_Blue	Color_Gre
0	23	10	2002	46986	90	1	0	2000	3	4	...	0	0	1	
1	23	10	2002	72937	90	1	0	2000	3	4	...	0	0	0	
2	24	9	2002	41711	90	1	0	2000	3	4	...	0	0	1	
3	26	7	2002	48000	90	0	0	2000	3	4	...	0	1	0	
4	30	3	2002	38500	90	0	0	2000	3	4	...	0	1	0	

5 rows × 45 columns

In [41]:
```
#applying scaling to ALL COLUMNS
#for presence of outliers use standard scaling other wise
#use min max scaling
from sklearn.preprocessing import StandardScaler
```

In [44]:
```
scaler=StandardScaler()
scaler.fit(feature_df)
```

Out[44]: StandardScaler()

In [48]:
```
scaler_df=scaler.transform(feature_df)
final_df=pd.DataFrame(scaler_df,columns=feature_df.columns)
final_df.head()
```

Out[48]:

	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	CC	Doors	Cylinders	...	Fuel_Type_Petrol	Color_Black	Color_
0	-1.771966	1.327576	1.541796	-0.574695	-0.768042	0.694219	-0.242893	0.997419	-1.085139	0.0	...	-2.710874	-0.391681	2.0
1	-1.771966	1.327576	1.541796	0.117454	-0.768042	0.694219	-0.242893	0.997419	-1.085139	0.0	...	-2.710874	-0.391681	-0.4
2	-1.718184	1.029329	1.541796	-0.715386	-0.768042	0.694219	-0.242893	0.997419	-1.085139	0.0	...	-2.710874	-0.391681	2.0
3	-1.610620	0.432833	1.541796	-0.547650	-0.768042	-1.440467	-0.242893	0.997419	-1.085139	0.0	...	-2.710874	2.553101	-0.4
4	-1.395491	-0.760158	1.541796	-0.801028	-0.768042	-1.440467	-0.242893	0.997419	-1.085139	0.0	...	-2.710874	2.553101	-0.4

5 rows × 45 columns

In [ ]:
```
#applying princial component analysis on this dataframe
```

In [50]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=0.95,random_state=5)
```

In [51]:
```
pca.fit(final_df)
```

Out[51]:
```
PCA(n_components=0.95, random_state=5)
```

In [52]:
```
feature_pca=pca.transform(final_df)
feature_pca.shape
```

Out[52]:
```
(1436, 32)
```

In [53]:
```
pca_df=pd.DataFrame(feature_pca)
pca_df.head()
```

Out[53]:

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25
0	2.797736	4.848568	0.343319	-0.727748	-0.141659	-0.388216	-1.339298	-1.458137	0.297432	1.220118	...	0.042180	-0.039088	-1.182932	-1.191405
1	3.222340	5.160836	0.882364	-1.161806	-0.323688	0.047251	-1.057623	-0.742248	-1.042610	0.963418	...	0.766375	1.445599	-1.445530	-0.389533
2	2.111314	5.011514	-0.582138	-2.319647	-0.390371	-0.702172	-1.436148	-0.391840	0.345422	1.848586	...	-0.973201	-0.387814	-0.454865	-1.539763
3	1.950404	5.083516	-0.839480	-1.781638	0.296364	-1.715682	-0.847632	0.705913	-0.836044	0.647460	...	-0.518379	-0.552121	-0.722059	-2.172290
4	3.794326	4.287844	0.137677	0.859629	0.696811	-1.076015	-0.457588	0.507594	-1.316836	0.371489	...	-0.104411	-1.180035	-0.311513	-2.369693

5 rows × 32 columns

In [54]:
```
pca_table=pd.DataFrame(pca.components_.T,index=final_df.columns)
```

In [55]:
```
pca_table
```

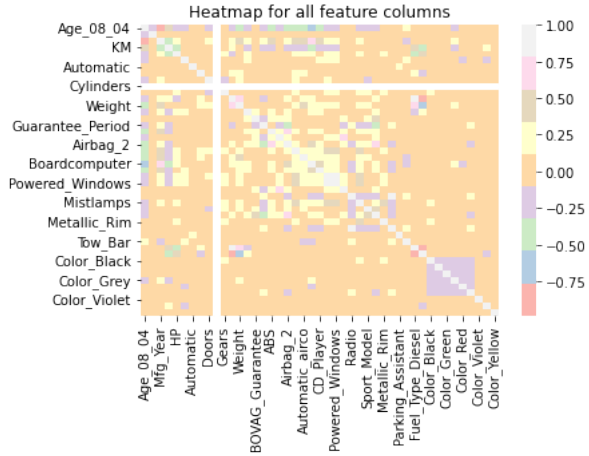
Out[55]:

	0	1	2	3	4	5	6	7	8	9	...	
Age_08_04	-3.404190e-01	-2.115058e-02	-1.582002e-01	2.550518e-01	1.823454e-02	1.220895e-01	8.599873e-03	0.067060	6.103597e-02	-2.356409e-02	...	2.815
Mfg_Month	1.588516e-02	2.527798e-02	-1.078668e-04	-1.154071e-02	-4.503892e-02	-1.160742e-01	-6.318007e-02	-0.167967	-3.241562e-02	2.502723e-01	...	-6.453
Mfg_Year	3.395870e-01	1.669218e-02	1.591723e-01	-2.544939e-01	-1.017368e-02	-1.017673e-01	2.810035e-03	-0.036992	-5.552286e-02	-2.169669e-02	...	-1.662
KM	-1.721741e-01	2.504879e-01	-9.851922e-02	2.685095e-01	-1.384994e-02	3.839868e-02	-3.367457e-02	-0.058246	-6.220366e-02	-4.230837e-03	...	1.083
HP	1.154640e-01	-2.900713e-01	1.077846e-01	7.490710e-02	9.361193e-03	-1.864322e-02	1.808414e-01	0.166084	1.998755e-01	-1.375759e-02	...	1.284
Met_Color	7.980948e-02	-3.676033e-02	7.093389e-02	2.689544e-02	-1.818924e-01	2.781066e-01	-4.229378e-01	0.160382	-3.040390e-02	1.522320e-01	...	-1.137
Automatic	-6.056964e-03	-3.333030e-02	8.204808e-03	-5.132102e-02	1.128064e-02	1.554104e-01	9.197235e-02	0.133317	4.312673e-01	1.206380e-01	...	-3.396

In [56]:
```
feature_mean=feature_df.mean()
feature_std=feature_df.std()
Z=(feature_df-feature_mean)/feature_std
```

In [57]:
```
C=Z.cov()
```

In [58]:
```
sns.heatmap(C,cmap='Pastel1')
plt.title('Heatmap for all feature columns')
plt.show()
```



In [59]:
```
print('The selected variables are: \n',pca.explained_variance_ratio_)
```

```
The selected variables are:
 [0.12722256 0.0866339  0.06910537 0.05533474 0.04644602 0.03651697
 0.03262553 0.030516   0.0295579  0.02901791 0.02718674 0.02633937
 0.025785   0.0248257  0.02374919 0.02322952 0.02272153 0.02206638
 0.02165934 0.01980327 0.01940082 0.01877815 0.01697536 0.01618503
 0.01579723 0.01527302 0.01389044 0.01349731 0.0120324  0.01130805
 0.01073508 0.01003409]
```

In [60]:
```
variable_list=pca.explained_variance_ratio_
print(variable_list)
```

```
[0.12722256 0.0866339  0.06910537 0.05533474 0.04644602 0.03651697
 0.03262553 0.030516   0.0295579  0.02901791 0.02718674 0.02633937
 0.025785   0.0248257  0.02374919 0.02322952 0.02272153 0.02206638
 0.02165934 0.01980327 0.01940082 0.01877815 0.01697536 0.01618503
 0.01579723 0.01527302 0.01389044 0.01349731 0.0120324  0.01130805
 0.01073508 0.01003409]
```

In [61]:
```
total_variation=variable_list.sum()
```

In [62]:
```
print('The total variation=',total_variation)
```

```
The total variation= 0.9542499514462021
```

In [ ]:
```
  #conclusion
   #original 45 features were reducted to 32 features
   #this presents data reduction technique
   #these 32 features represents around 95% of data variation
   #PCA reduced 45 features to 32 features
   #these 32 features explains about 95% of data
```