In [8]:

```
##### =============================================================================
# PREDICTING PRICE OF PRE-OWNED CARS
# =============================================================================

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
# Importing model_selection library for using cross_validation
from sklearn import model_selection
# Importing the library for PCA
from sklearn.decomposition import PCA
```

In [9]:

```
# =============================================================================
# Setting dimensions for plot
# =============================================================================

sns.set(rc={'figure.figsize':(11.7,8.27)})
```

In [80]:

```
# =============================================================================
# Reading CSV file
# =============================================================================
data = pd.read_csv('Toyota.csv',index_col=0)
```

In [81]:

```
##### =============================================================================
# Structure of the dataset
# =============================================================================
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price        1436 non-null int64
Age          1436 non-null float64
KM           1436 non-null float64
FuelType     1436 non-null object
HP           1436 non-null float64
MetColor     1436 non-null float64
Automatic    1436 non-null int64
CC           1436 non-null int64
Doors        1436 non-null int64
Weight       1436 non-null int64
dtypes: float64(4), int64(5), object(1)
memory usage: 123.4+ KB
None
```

In [83]:

```
######## CHECK FOR THE MISSING VALUES
data.isnull().sum()
```

Out[83]:

```
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

In [84]:

```
data=pd.get_dummies(data,drop_first=True)
```

In [85]:

```
# Storing the column names in variables
features = list(set(data.columns)-set(['Price']))
target   = list(['Price'])

print(features)
print(target)
```

```
['FuelType_Diesel', 'HP', 'Automatic', 'CC', 'Age', 'FuelType_Petrol', 'K
M', 'Weight', 'MetColor', 'Doors']
['Price']
```

In [123]:

```
# =============================================================================

x = data.loc[:, features]
y = data.loc[:,target]

# Sklearn - package to split data into train & test
from sklearn.model_selection import train_test_split

# Splitting test & train as 30% and 70%

train_x, test_x, train_y, test_y = train_test_split(x,y,
                                                    test_size=0.3,
                                                    random_state=40)
```

In [103]:

```
#####################MULTIPLE LINEAR REGRESSION#######################################
####
import statsmodels.api as sm
model = sm.OLS(train_y, train_x).fit() ## sm.OLS(output, input)
model.summary()
predictions = model.predict(test_x)
```

In [104]:

```
# finding the mean for test data value
base_pred = np.mean(test_y)
print(base_pred)

# Repeating same value till length of test data
base_pred = np.repeat(base_pred, len(test_y))
```

```
Price    10521.624
dtype: float64
```

In [105]:

```
# finding the RMSE
from sklearn.metrics import mean_squared_error
base_RMSE =(mean_squared_error(test_y,base_pred))**0.5
print(base_RMSE)
```

```
3518.8221654138106
```

In [106]:

```
# RMSE of the linear model
lr_rmse = (mean_squared_error(test_y,predictions))**0.5
print("RMSE corresponding to Linear Regression model between X and Y: ",lr_rmse)
```

```
1345.6596407279985
```

Below is the script for Principal Component Regression

1. Response variable is Price (train_y, test_y)
2. Independent variables are 'FuelType_Diesel', 'HP', 'Automatic', 'CC', 'Age', 'FuelType_Petrol', 'KM', 'Weight', 'MetColor', 'Doors' (train_x and test_x)
3. Choose number of PCs starting from 1 till number of features in the dataset (which is 10 in this case). i. Run PCA among independent variables of train_x and get the PCs in pc_train, pc_test for each fold in the cross validation. ii. pc_train, pc_test are linear combinations of train_x and test_x iii. Regress train_y and test_y on pc_train and pc_test respectively. iv. Predict value of test_y based on value of pc_test
4. Draw a plot between the number of PCs Vs. RMSE
5. Choose the number of PCs corresponding to lowest RMSE

In [124]:

```
'''
Function to linearly regress the training samples of X against Y
and to return the list of rmse for each variable in the output (y) of the testing sampl
es
Arguments: X_train (numpy.ndarray),y_train (numpy.ndarray),X_test (numpy.ndarray),y_tes
t (numpy.ndarray)
Returns: rmse (list of arrays)
'''
def linreg(X_train,X_test,y_train,y_test):
 ols = LinearRegression()
 ols.fit(X_train, y_train)
 y_pred = ols.predict(X_test)
 rmse=np.sqrt(((y_test-y_pred)**2).mean()) #square root (mean square error)
 return rmse
```

In [125]:

```
'''
Function to perform linear regression using leave one out cross validation between Y
and reduced set of X iteratively from 1 principal component to maximum number of princi
pal coomponents specified
Arguments: X (numpy.ndarray), y (numpy.ndarray), nfact (int)
Returns: RMSE_pcr_arr (numpy.ndarray of size nfact,number of variables of y)
'''
def pca_kfold_ols(X,y,nfact):
 RMSE_pcr_lst=list()
 for pc in range(1,nfact,1): # Iterating over number of principal components
     pca = PCA(n_components=pc) # Instantiating pca instance for each number of princip
al components in the iteration process
     X_red = pca.fit_transform(X)
     rmse_pcr_lst=list()
     k = model_selection.KFold(5) # Instantiating a kfold cv instance
     for tr_idx, tst_idx in k.split(X_red): # interating through multiple folds
         pc_train, pc_test = X_red[tr_idx], X_red[tst_idx] # input X for both train and
test
         y_train, y_test = y[tr_idx], y[tst_idx] # output Y for both train and test
         rmse_pcr_lst.append(linreg(pc_train,pc_test,y_train,y_test)) # Appending rmse
 for each fold into a list
     RMSE_pcr_lst.append(np.array(rmse_pcr_lst).mean()) # Averaging RMSE of all the fol
ds per
 return np.array(RMSE_pcr_lst) #
```

In [126]:

```
train_x.head()
```

Out[126]:

	FuelType_Diesel	HP	Automatic	CC	Age	FuelType_Petrol	KM	Weight
978	0	110.000	0	1600	65.000	1	45681.000	1050
1206	0	110.000	0	1600	73.000	1	87358.000	1050
1168	0	86.000	0	1300	78.000	1	96000.000	1015
1226	0	110.000	0	1600	55.672	1	84000.000	1075
1010	0	110.000	0	1600	60.000	1	36943.000	1070

In [127]:

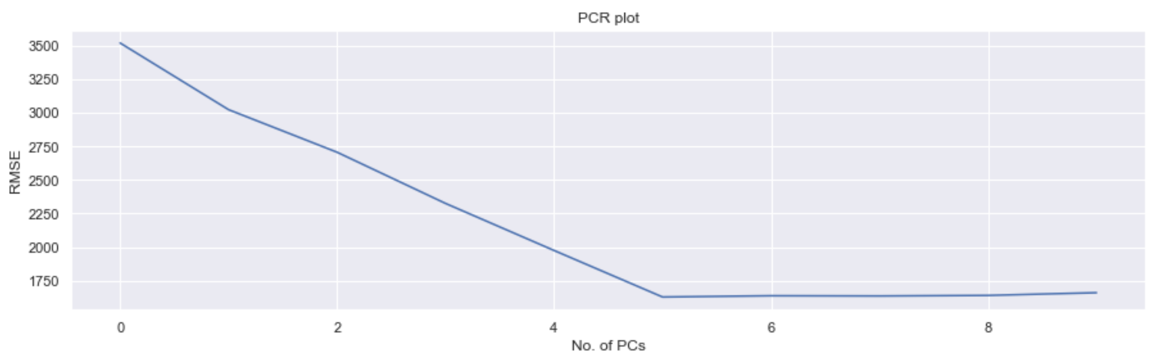
```
pcs=10
train_x=train_x.to_numpy()
train_y=train_y.to_numpy()
RMSE_pcr=np.append(base_RMSE,pca_kfold_ols(train_x,train_y,pcs))
RMSE_pcr.shape
```

Out[127]:

```
(10,)
```

In [128]:

```
fig = plt.figure(figsize=(15,4))
plt.plot(range(pcs),RMSE_pcr)
plt.xlabel('No. of PCs')
plt.ylabel('RMSE')
plt.title('PCR plot')
plt.grid(True)
```



First five PCs capture maximum variance in the given data.

In [129]:

```
print("The RMSE corresponding to linreg model between Z(5 PCs) and Y is: ", RMSE_pcr[5
])
```

```
The RMSE corresponding to linreg model between Z(5 PCs) and Y is:  1628.83
5612295244
```

In [130]:

```
## END OF SCRIPT
```