

Linear Regression of Used Toyota Cars in UK Project

Presented by:

Ahmad Alharthi

Faisal Alsufyani

Yahya Alyoubi

Instructor:

Dr.Mejdal Alqahtani



Introduction

This data set was collected from kaggle.com website which contains information of price, transmission, mileage, fuel type, road tax, miles per gallon (mpg), and engine size of used Toyota cars in United Kingdom.

Problem Statement:

- Find which factor has the most effect on car price?
- Which factor has more impact on car price drop?
- Find the best regression model to predict price with high accuracy?

Web Scrapping



Web Scrapping

```
import csv
from bs4 import BeautifulSoup

# opening html offline & parsing with BeautifulSoup
htm= open("UK_usedCars.html", 'r')
soup = BeautifulSoup(htm.read(), 'html.parser')

def extract_rows(soup,class_='sc-gA0GNj sc-bXGNvv eYELsP hTKZnU'):
    """ Extracting data from html based on div class """
    rows= soup.find_all('div',class_)

    return [rows[i].getText('div').split('div') for i in range(len(rows))]

def export_data(rows):
    """ Export data to csv file with headers """
    with open('toyota.csv', 'w') as csvfile:
        file_obj = csv.writer(csvfile)
        file_obj.writerow(['model', 'year','price','transmission','mileage','fuelType','tax','mpg', 'engineSize'])
        file_obj.writerows(rows)

data = extract_rows(soup,class_='sc-gA0GNj sc-bXGNvv eYELsP hTKZnU')
export_data(data)
```

```
data[0:5]
```

```
[[' GT86', '2016', '16000', 'Manual', '24089', 'Petrol', '265', '36.2', '2.0'],
 [' GT86', '2017', '15995', 'Manual', '18615', 'Petrol', '145', '36.2', '2.0'],
 [' GT86', '2015', '13998', 'Manual', '27469', 'Petrol', '265', '36.2', '2.0'],
 [' GT86', '2017', '18998', 'Manual', '14736', 'Petrol', '150', '36.2', '2.0'],
 [' GT86', '2017', '17498', 'Manual', '36284', 'Petrol', '145', '36.2', '2.0']]
```

Duplicate values

```
# searching for douplicate values
```

```
dataset.groupby(dataset.columns.tolist(),as_index=False).size().sort_values()
```

model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price	
Auris	2007	Automatic	84000	Petrol	205	40.9	1.6	3395	1
Verso	2011	Manual	67085	Petrol	200	41.5	1.6	5999	1
			54000	Diesel	145	53.3	2.0	5995	1
			47570	Petrol	200	41.5	1.6	6795	1
			29761	Diesel	145	53.3	2.0	7995	1
									..
Aygo	2016	Manual	12935	Petrol	0	69.0	1.0	8450	2
	2019	Manual	2000	Petrol	145	57.7	1.0	10350	3
			1500	Petrol	145	56.5	1.0	9999	3
			25	Petrol	145	56.5	1.0	9995	3
RAV4	2015	Manual	45757	Diesel	125	57.6	2.0	13500	3

Length: 6699, dtype: int64

```
dataset = dataset.drop_duplicates()
```

```
dataset.shape
```

(6699, 9)

```
dataset.groupby(dataset.columns.tolist(),as_index=False).size().sort_values()
```

model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price	
Auris	2007	Automatic	84000	Petrol	205	40.9	1.6	3395	1
Verso	2011	Manual	67085	Petrol	200	41.5	1.6	5999	1
			54000	Diesel	145	53.3	2.0	5995	1
			47570	Petrol	200	41.5	1.6	6795	1
			29761	Diesel	145	53.3	2.0	7995	1
									..
Aygo	2018	Manual	14716	Petrol	145	56.5	1.0	8222	1
			14696	Petrol	145	68.9	1.0	7995	1
			14680	Petrol	145	69.0	1.0	6995	1
			12987	Petrol	145	56.5	1.0	8295	1
Yaris	2020	Manual	4895	Petrol	150	49.6	1.0	12495	1

Length: 6699, dtype: int64

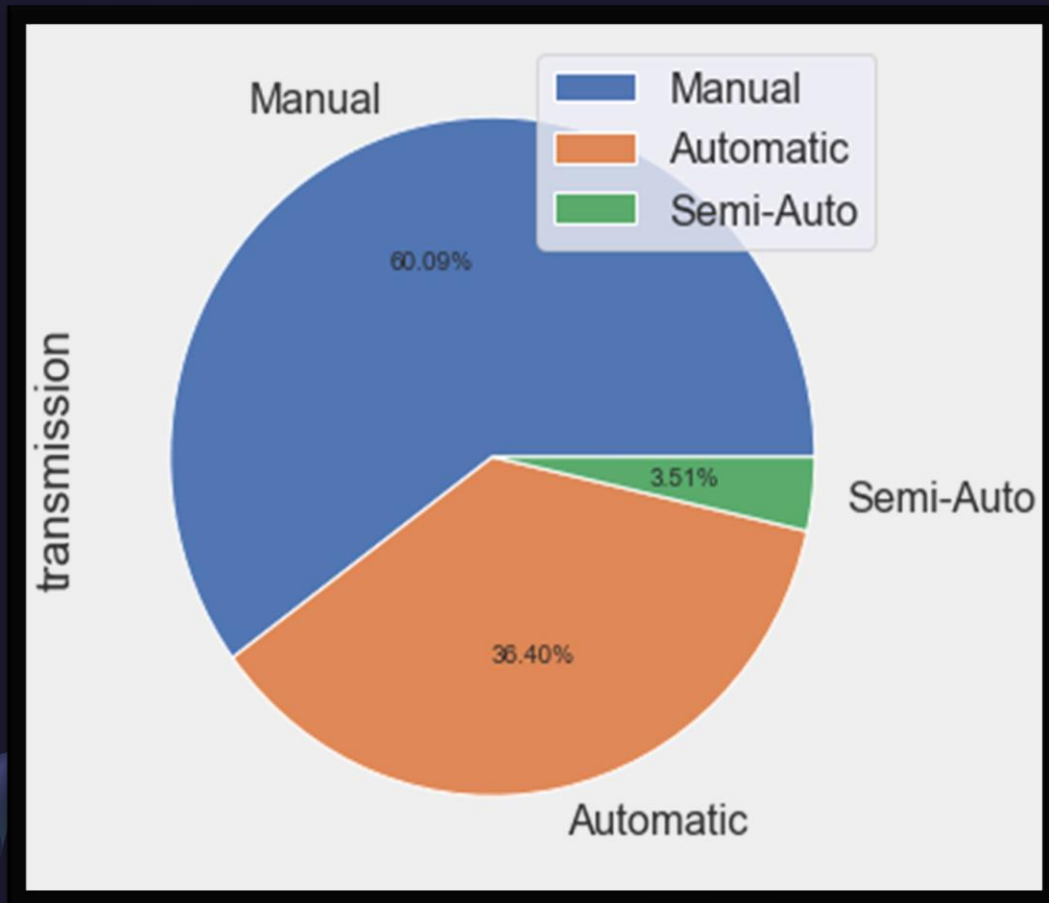
```
df=df[~((df.transmission == "Other"))]
```

```
df=df[~((df.fuelType == "Other"))]
```

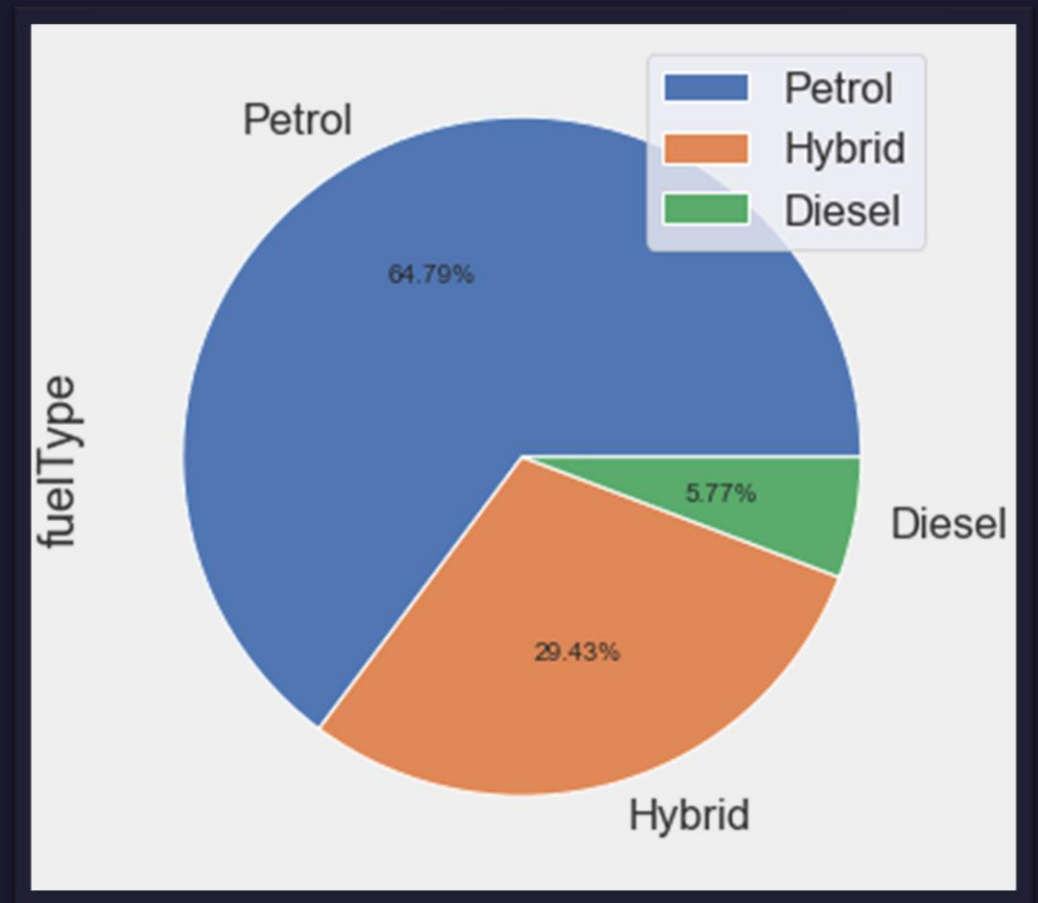
```
df[df.columns[[1,3,5,6,7,8,]]].corr()
```


visualization

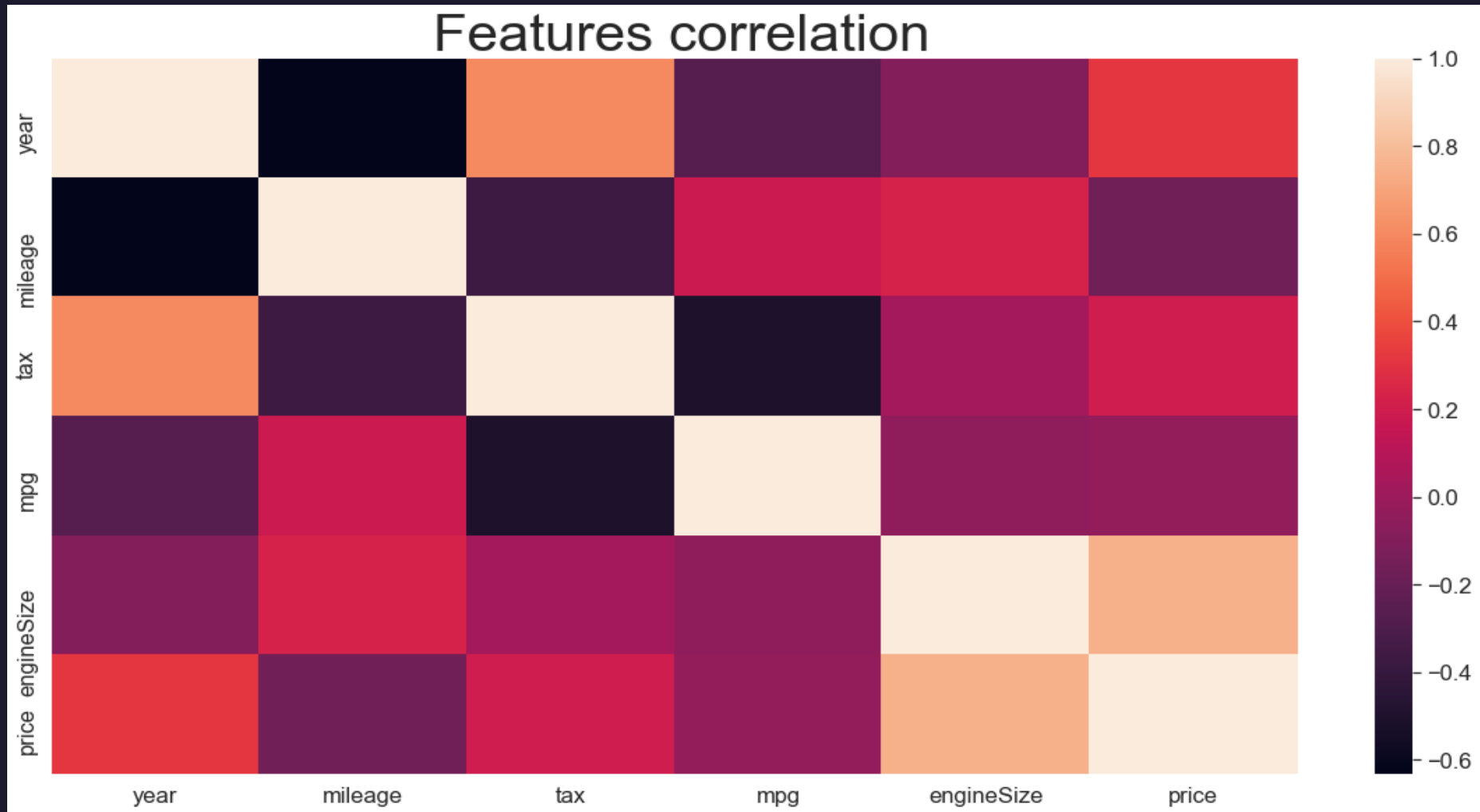
Frequency /counts of car transmission types



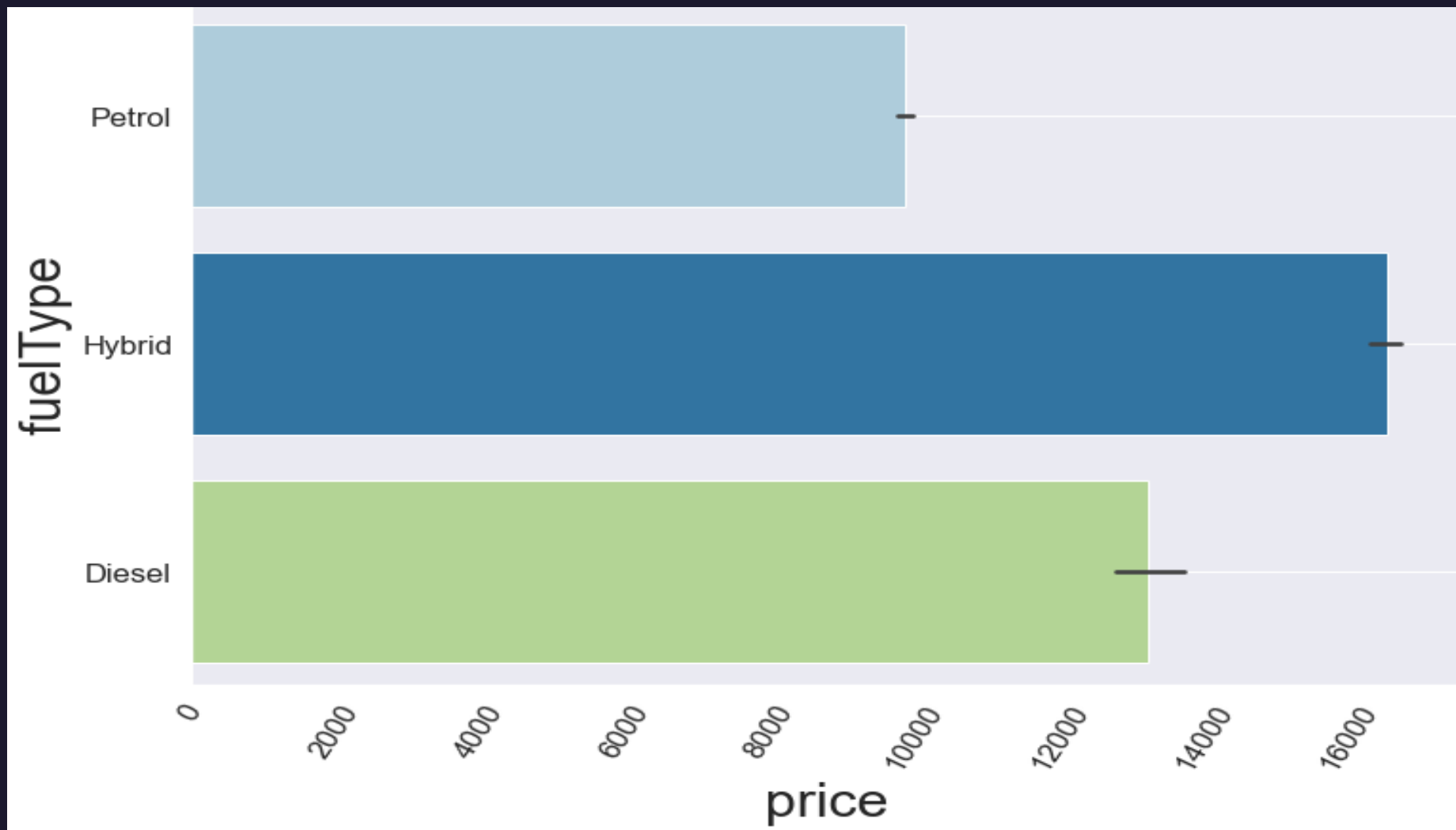
Frequency /counts of car based on fuel types



Heatmap



Price Based on Fuel Type



Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pylab import rcParams
rcParams['figure.figsize'] = 9, 5
```

```
dataset = pd.read_csv('toyota_cleaned_dataset_v2.csv')
dataset.head(2)
```

	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	GT86	2016	Manual	24089	Petrol	265	36.2	2.0	16000
1	GT86	2017	Manual	18615	Petrol	145	36.2	2.0	15995

```
# up to 6 values are 0
# to solve this issue substitute with mean
dataset['engineSize'][dataset.engineSize == 0] = dataset.engineSize.median()
```

```
dataset = pd.get_dummies(dataset, drop_first=True)
dataset.head(3)
```

	year	mileage	tax	mpg	engineSize	price	model_Avensis	model_Aygo	model_C-HR	model_Camry	...	model_Hilux	model_PROACE VERSO
0	2016	24089	265	36.2	2.0	16000	0	0	0	0	...	0	0
1	2017	18615	145	36.2	2.0	15995	0	0	0	0	...	0	0
2	2015	27469	265	36.2	2.0	13998	0	0	0	0	...	0	0

3 rows x 22 columns

```
X, y = dataset.drop('price', axis=1), dataset['price']
```

```
# splitting data to 20/80 for testing and training
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=10)
```

```
print(X_train.shape, X_test.shape)
```

```
(4672, 21) (1169, 21)
```

Linear Regression

```
# linear model
lm = LinearRegression()

# linear regression model cross validation
val_score = cross_val_score(lm, X_train, y_train, # estimator, features, target
                             cv=5, # number of folds
                             scoring='r2') # scoring metric
val_sc = round(val_score.mean(),4)

lm.fit(X_train, y_train)
y_pred_test = lm.predict(X_test)
r_2 = round(metrics.r2_score(y_test,y_pred_test),4)
mean_sq = round(mean_squared_error(y_test,y_pred_test),2 )
print(f'The average validation score is: {val_sc}\t AND R^2 = {r_2}')
print(f'Mean Squared Error: {mean_sq}')
```

The average validation score is: 0.9487 AND R^2 = 0.9452
Mean Squared Error: 1264586.81

```
# lasso model
lm_lasso = Lasso(alpha=0.7)

# lasso regression model cross validation
val_score = cross_val_score(lm_lasso, X_train, y_train, # estimator, features, target
                             cv=5, # number of folds
                             scoring='r2') # scoring metric
val_sc = round(val_score.mean(),4)

lm_lasso.fit(X_train, y_train)
y_pred_test = lm_lasso.predict(X_test)
r_2 = round(metrics.r2_score(y_test,y_pred_test),4)
mean_sq = round(mean_squared_error(y_test,y_pred_test),2 )
print(f'The average validation score is: {val_sc}\t AND R^2 = {r_2}')
print(f'Mean Squared Error: {mean_sq}')
```

The average validation score is: 0.9486 AND R^2 = 0.9454
Mean Squared Error: 1260993.2

Linear Regression

```
# Ridge model
lm_reg = Ridge(alpha=1)

# Ridge regression model cross validation
val_score = cross_val_score(lm_reg, X_train, y_train, # estimator, features, target
                             cv=5, # number of folds
                             scoring='r2') # scoring metric
val_sc = round(val_score.mean(),4)

lm_reg.fit(X_train, y_train)
y_pred_test = lm_reg.predict(X_test)
r_2 = round(metrics.r2_score(y_test,y_pred_test),4)
mean_sq = round(mean_squared_error(y_test,y_pred_test),2 )
print(f'The average validation score is: {val_sc}\t AND R^2 = {r_2}')
print(f'Mean Squared Error: {mean_sq}')
```

The average validation score is: 0.9483 AND R² = 0.9454
Mean Squared Error: 1260829.35

```
# Feature scaling for train, val, and test so that we can run our ridge model on each
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train.values)
X_test_scaled = scaler.transform(X_test.values)

# Ridge model
lm_reg = Ridge(alpha=1)

# Ridge regression model cross validation
val_score = cross_val_score(lm_reg, X_train_scaled, y_train, # estimator, features, target
                             cv=5, # number of folds
                             scoring='r2') # scoring metric
val_sc = round(val_score.mean(),4)

lm_reg.fit(X_train_scaled, y_train)
y_pred_test = lm_reg.predict(X_test_scaled)
r_2 = round(metrics.r2_score(y_test,y_pred_test),4)
mean_sq = round(mean_squared_error(y_test,y_pred_test),2 )

print(f'The average validation score is: {val_sc}\t AND R^2 = {r_2}')
print(f'Mean Squared Error: {mean_sq}')
```

The average validation score is: 0.9487 AND R² = 0.9452
Mean Squared Error: 1264357.72

Linear Regression

```
# polynomial model
# Feature transforms for train test so that we can run our poly model on each
poly = PolynomialFeatures(degree=2)

X_train_poly = poly.fit_transform(X_train.values)
X_test_poly = poly.transform(X_test.values)

lm_poly = LinearRegression()

# polynomial regression model cross validation
val_score = cross_val_score(lm_poly, X_train_scaled, y_train, # estimator, features, target
                             cv=5, # number of folds
                             scoring='r2') # scoring metric
val_sc = round(val_score.mean(),4)

lm_poly.fit(X_train_scaled, y_train)
y_pred_test = lm_poly.predict(X_test_scaled)
r_2 = round(metrics.r2_score(y_test,y_pred_test),4)
mean_sq = round(mean_squared_error(y_test,y_pred_test),2 )
print(f'The average validation score of Polynomial regression with degree 2 is: {val_sc}\t AND R^2 = {r_2}')
print(f'Mean Squared Error: {mean_sq}')
```

The average validation score of Polynomial regression with degree 2 is: 0.9487 AND R^2 = 0.9452
Mean Squared Error: 1264586.81

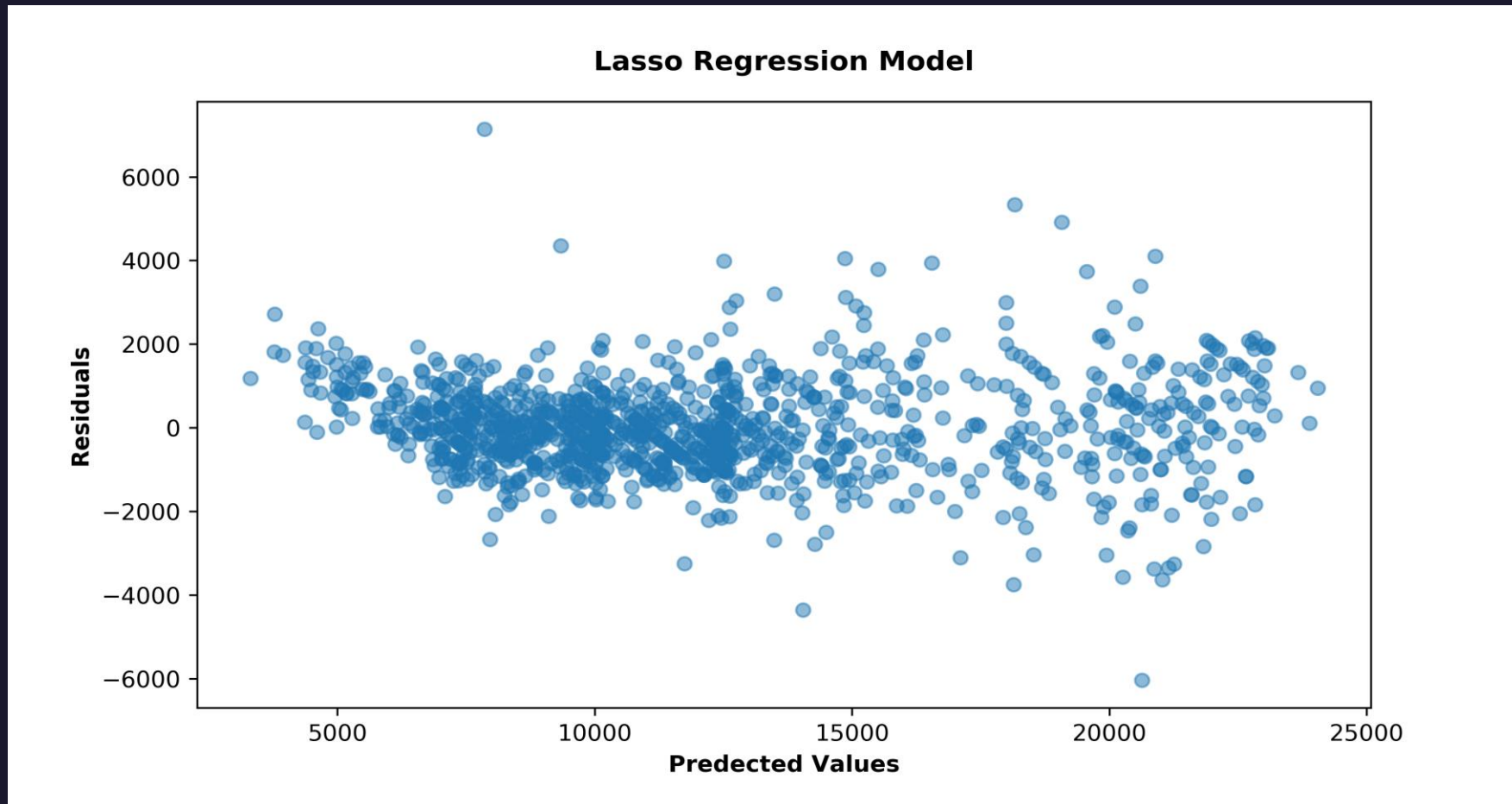
```
# mean sq error
# lasso Error: min => 1260993.2
print("Selected Model: Lasso Regression")
print('Model coefficients: %s'%(lm_lasso.coef_))
print('Model intercept: ',round(lm_lasso.intercept_),4)
```

Selected Model: Lasso Regression

Model coefficients: [7.99151974e+02 -5.20032320e-02 -3.48953929e+00 -2.69901292e+01
2.86263477e+03 3.18919738e+02 -2.51605732e+03 5.21558343e+03
1.58117349e+03 4.69229517e+03 4.81102119e+03 6.36237794e+03
8.50522130e+03 5.07727276e+03 3.51444407e+03 7.18838980e+02
-1.64006402e+03 -1.39089813e+03 -2.37771014e+02 2.40951966e+03
5.59965688e+02]

Model intercept: -1600765.0 4

Residuals



Conclusion

We test 4 models of regression & based on R^2 score we select Lasso Regression .

With 0.94 accuracy .

Minimum mean squared error .

Future work :

Applying deep learning algorithm to enhance the accuracy .



Thank you for
listening
Any question ?

