Old car price Predcitions

2022/11~2022/12 Create by Ming

Old car price Predcitions

2022/11~2022/12

【根據資料製作模型，預測二手車價值】

- 利用對二手車市場的認識，進行資料的分類處理
  - 品牌價值
  - 排氣量
- 資料除錯，分析可能有資料誤植的可能性
- 數據視覺化呈現，觀察各項指標與價錢的關係
  - 特別做年份高的二手車分析，希望排除收藏車的疑慮
- 使用迴歸與資料探勘進行推估，推估準確度超過80%
- 比對預測與實際資料，分析模型現階段的問題，並提出可改良的方向

About Dataset

The steps listed below must be included in your notebooks:

1.Understand the problem statement.

2.Import required libraries and Data.

3.Check the Data

4.Pre-processing and data cleansing.

5.Utilize the provided dataset to conduct exploratory data analysis.

6.Feature Selection

7.Data splitting

8.Create an ML model, then test it using various metrics.

了解資料結構

判斷目標及其他特徵

資料共5512筆，有10個欄位

```
Data shape: (5512, 10)
```

Unnamed: 0	car_name	car_prices_in_rupee	kms_driven	fuel_type	transmission	ownership	manufacture	engine	Seats
0	Jeep Compass 2.0 Longitude Option BSIV	10.03 Lakh	86,226 kms	Diesel	Manual	1st Owner	2017	1956 cc	5 Seats
1	Renault Duster RXZ Turbo CVT	12.83 Lakh	13,248 kms	Petrol	Automatic	1st Owner	2021	1330 cc	5 Seats
2	Toyota Camry 2.5 G	16.40 Lakh	60,343 kms	Petrol	Automatic	1st Owner	2016	2494 cc	5 Seats
3	Honda Jazz VX CVT	7.77 Lakh	26,696 kms	Petrol	Automatic	1st Owner	2018	1199 cc	5 Seats
4	Volkswagen Polo 1.2 MPI Highline	5.15 Lakh	69,414 kms	Petrol	Manual	1st Owner	2016	1199 cc	5 Seats

學習目標 – 價格

Objec型態需要做轉換

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5512 entries, 0 to 5511
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Unnamed: 0           5512 non-null   int64 
 1   car_name             5512 non-null   object
 2   car_prices_in_rupee  5512 non-null   object
 3   kms_driven           5512 non-null   object
 4   fuel_type            5512 non-null   object
 5   transmission         5512 non-null   object
 6   ownership            5512 non-null   object
 7   manufacture          5512 non-null   int64 
 8   engine               5512 non-null   object
 9   Seats                5512 non-null   object
dtypes: int64(2), object(8)
memory usage: 430.8+ KB
```

資料沒有重複

```
df[df.duplicated()].count()
```

```
Unnamed: 0             0
car_name               0
car_prices_in_rupee    0
kms_driven             0
fuel_type              0
transmission           0
ownership              0
manufacture            0
engine                 0
Seats                  0
dtype: int64
```

汽車品牌對車價也有重要影響

因此將汽車品牌從原始欄位獨立出來，方便後續操作

In [6]:

```
newcols = df["car_name"].str.split(" ",n=1)
newcols.index = df.index
df.insert(1,"Brand","")
for i in range(5512):
    df["Brand"][i] = newcols[i][0]
    df["car_name"][i] = newcols[i][1]
df.head()
```

Out[6]:

	Unnamed: 0	Brand	car_name	car_prices_in_rupee	kms_driven	fuel_type	transmission	ownership	manufacture
0	0	Jeep	Compass 2.0 Longitude Option BSIV	10.03 Lakh	86,226 kms	Diesel	Manual	1st Owner	2017
1	1	Renault	Duster RXZ Turbo CVT	12.83 Lakh	13,248 kms	Petrol	Automatic	1st Owner	2021
2	2	Toyota	Camry 2.5 G	16.40 Lakh	60,343 kms	Petrol	Automatic	1st Owner	2016
3	3	Honda	Jazz VX CVT	7.77 Lakh	26,696 kms	Petrol	Automatic	1st Owner	2018
4	4	Volkswagen	Polo 1.2 MPI Highline	5.15 Lakh	69,414 kms	Petrol	Manual	1st Owner	2016

移除欄位中的單位，保留數字，讓資料更整潔，也方便轉換型態

In [9]:

```
df["kms_driven"] = df["kms_driven"].str.strip("kms")
df["kms_driven"] = df["kms_driven"].str.replace(",","")

df['Seats']=df['Seats'].str.replace(' Seats','')

df['engine']=df['engine'].str.replace(' cc','')

df["ownership"] = df["ownership"].str.strip("Owner")
for i,item in  enumerate(df["ownership"]):
    df["ownership"][i] = df["ownership"][i][0]
```

將製造年份，轉換成為使用年份，更適用於判斷二手車的價值

```
df.insert(11,"Age","")
df["Age"] = 2023 - df["manufacture"]
df.head()
```

	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	manufacture	Age	engine	engine_class	Seats
	37.30	10K	86226	Diesel	Manual	1	2017	6	1956	3	5
	47.72	10K	13248	Petrol	Automatic	1	2021	2	1330	2	5
	61.00	10K	60343	Petrol	Automatic	1	2016	7	2494	4	5
	28.90	10K	26696	Petrol	Automatic	1	2018	5	1199	1	5
	19.15	10K	69414	Petrol	Manual	1	2016	7	1199	1	5

```
for j,item in enumerate(df["car_prices_in_rupee"]):
    if "Lakh" in item:
       df["car_prices_in_rupee"][j] = df["car_prices_in_rupee"][j].strip("Lakh")
       #print(j,",",df["car_prices_in_rupee"][j],type(df["car_prices_in_rupee"][j]))
       df["car_prices_in_rupee"][j]=pd.to_numeric(df["car_prices_in_rupee"][j])
       df["car_prices_in_rupee"][j]= round((df["car_prices_in_rupee"][j]*37193),2)
    elif "Crore" in item:
        df["car_prices_in_rupee"][j] = df["car_prices_in_rupee"][j].strip("Crore")
        #print(df["car_prices_in_rupee"][j])
        df["car_prices_in_rupee"][j]=pd.to_numeric(df["car_prices_in_rupee"][j])
        df["car_prices_in_rupee"][j]= round((df["car_prices_in_rupee"][j]*3712136),2)
    else:
        df["car_prices_in_rupee"][j] = df["car_prices_in_rupee"][j].replace(",","")
        df["car_prices_in_rupee"][j]= pd.to_numeric(df["car_prices_in_rupee"][j])
        df["car_prices_in_rupee"][j]= round((df["car_prices_in_rupee"][j]*0.37),2)

df.rename(columns={'car_prices_in_rupee': 'car_prices_in_NT',
                   }, inplace=True)
```

```
df["car_prices_in_NT"] = (df["car_prices_in_NT"]/10000).astype('float')
df["car_prices_in_NT"] = df["car_prices_in_NT"].round(2)
df.rename(columns={'car_prices_in_NT': 'car_prices',
                   }, inplace=True)
df.insert(4,"multiply NT","10K")
df.head()
```

原始資料使用的幣值是盧比，為了更好去理解相對價值

因此我利用匯率，將幣值轉換成台幣，並統一設定是以

10K當作基礎。

	Unnamed: 0	Brand	car_name	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	manufacture
0	0	Jeep	Compass 2.0 Longitude Option BSIV	37.30	10K	86,226 kms	Diesel	Manual	1st Owner	2017
1	1	Renault	Duster RXZ Turbo CVT	47.72	10K	13,248 kms	Petrol	Automatic	1st Owner	2021
2	2	Toyota	Camry 2.5 G	61.00	10K	60,343 kms	Petrol	Automatic	1st Owner	2016
3	3	Honda	Jazz VX CVT	28.90	10K	26,696 kms	Petrol	Automatic	1st Owner	2018
4	4	Volkswagen	Polo 1.2 MPI Highline	19.15	10K	69,414 kms	Petrol	Manual	1st Owner	2016

發現有排氣量為0的資料，特別找出不為0的最大最小值

```
#Find out the range of "engine"
print("min:",df[df["engine"]!=0]["engine"].min(),"max:",df["engine"].max())
```

```
min: 624 max: 5950
```

```
df.insert(11,"engine_class","")
for i,size in enumerate (df["engine"]):
    if size == 0:
        df["engine_class"][i] = 0
    elif size <=1200:
        df["engine_class"][i] = 1
    elif 1201<= size <=1800:
        df["engine_class"][i] = 2
    elif 1801<= size <=2400:
        df["engine_class"][i] = 3
    elif 2401<= size <=3000:
        df["engine_class"][i] = 4
    elif 3001<= size <=4200:
        df["engine_class"][i] = 5
    elif 4201<= size <=5400:
        df["engine_class"][i] = 6
    elif 5401<= size <=6600:
        df["engine_class"][i] = 7
df.sample(5)
```

汽機車稅金級距表

小客車自用 / 營業使用牌照稅稅額表

排氣量 (c.c)	稅額 (座位9人以下者)	
	自用 (全年)	營業 (全年)
500以下	1,620	900
501~600	2,160	1,260
601~1200	4,320	2,160
1201~1800	7,120	3,060
1801~2400	11,230	6,480
2401~3000	15,210	9,900
3001~4200	28,220	16,380
4201~5400	46,170	24,300
5401~6600	69,690	33,660
6001~7800	117,000	44,460
7801以上	151,200	56,700

利用臺灣小客車 牌照稅的稅金級距

將排氣量歸納成不同等級

r_name	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	manufacture	engine	engine_class	Seats
lo 1.5 ıl ımfortline	14.62	10K	71033	Diesel	Manual	2	2014	1248	2	5
nue S ıs	29.31	10K	16591	Petrol	Manual	1	2021	796	1	5
and i10 2 Kappa ortz ıtion AT	23.25	10K	28387	Petrol	Automatic	1	2019	1197	1	5
ıctor arp CVT	66.09	10K	11081	Petrol	Automatic	1	2021	1995	3	5
n Estilo 1 LX BSIII	3.29	10K	29081	Petrol	Manual	3	2006	2179	3	5

```
car_brand = {'LV1':["Toyota","Nissan","Mitsubishi","Ford","Hyundai","Isuzu","Honda","Datsu
n","Fiat","Kia","MG","Mahindra","Maruti","Premier","Tata","Renault","Force"],
             'LV2':["Jeep","Lexus","Mini","Skoda","Volkswagen","Volvo"],
             'LV3':["Audi","BMW","Chevrolet","Jaguar","Land","Mercedes-Benz","Porsche"],
             'LV4':["Bentley","Maserati"]}

df.insert(2,"Brand_lv","")
for i,brand in enumerate (df["Brand"]):
    if brand in car_brand['LV1']:
        df["Brand_lv"][i] = 0
    elif brand in car_brand['LV2']:
        df["Brand_lv"][i] = 1
    elif brand in car_brand['LV3']:
        df["Brand_lv"][i] = 2
    elif brand in car_brand['LV4']:
        df["Brand_lv"][i] = 3
    else:
        print(brand)

df["Brand_lv"].value_counts()
```

```
0    4337
2     793
1     376
3       6
Name: Brand_lv, dtype: int64
```

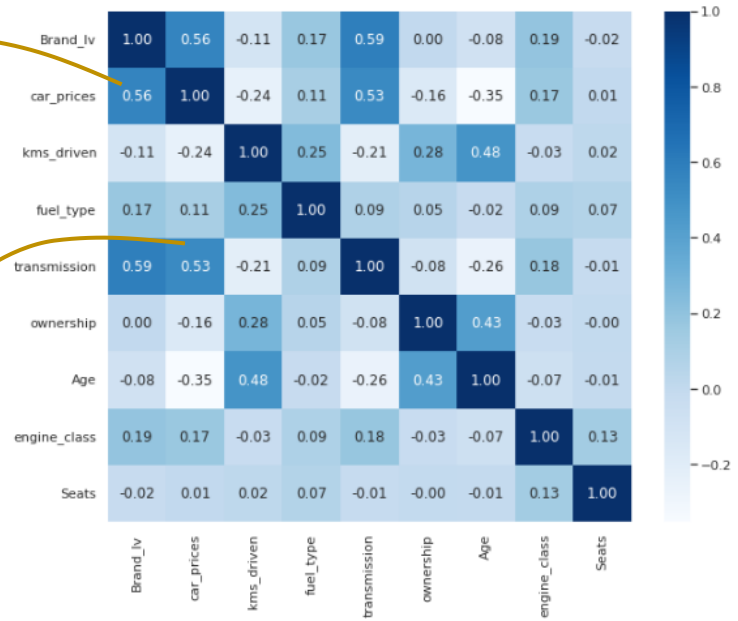
利用汽車個品牌的品牌價值，將汽車品牌分為4個級別

	Unnamed: 0	Brand	Brand_lv	car_name	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	m
0	0	Jeep	1	Compass 2.0 Longitude Option BSIV	37.30	10K	86226	Diesel	Manual	1	2(
1	1	Renault	0	Duster RXZ Turbo CVT	47.72	10K	13248	Petrol	Automatic	1	2(
2	2	Toyota	0	Camry 2.5 G	61.00	10K	60343	Petrol	Automatic	1	2(
3	3	Honda	0	Jazz VX CVT	28.90	10K	26696	Petrol	Automatic	1	2(
4	4	Volkswagen	1	Polo 1.2 MPI Highline	19.15	10K	69414	Petrol	Manual	1	2(

```
df_hm = df2.copy()
df_hm["fuel_type"] = df_hm["fuel_type"].map({"Petrol":0,"Diesel":1,"Cng":2,"Lpg":3,"Electric":
4})
df_hm["transmission"] = df_hm["transmission"].map({"Manual":0,"Automatic":1})

# Verify the heatmap of correlations
plt.figure(figsize = (10, 8))
sns.heatmap(df_hm.corr(), cmap = 'Blues', annot = True, fmt = '.2f');
```

價格與品牌有較高關聯

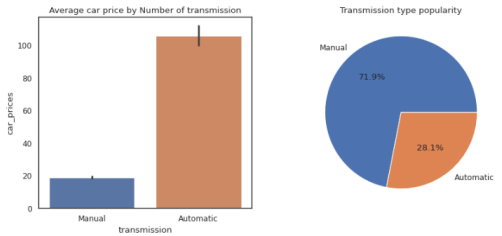
價格與傳動有較高關聯



傳動方式的比例關係

```
#Transmission

plt.figure(figsize=(12, 5), dpi=80)
plt.subplot(1, 2, 1)
ax = sns.barplot(x="transmission", y='car_prices', data=df2)
ax.set_title('Average car price by Number of transmission')

trans = df2["transmission"].copy()
count = trans.value_counts()
#print(count)
plt.subplot(1, 2, 2)
plt.pie(x=count,autopct='%.1f%%',labels = ["Manual","Automatic"])
plt.title('Transmission type popularity')
```

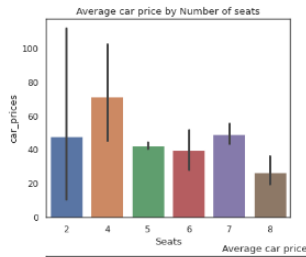
```
Text(0.5, 1.0, 'Transmission type popularity')
```

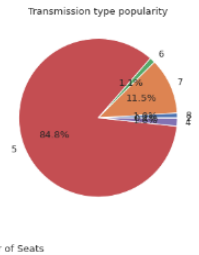


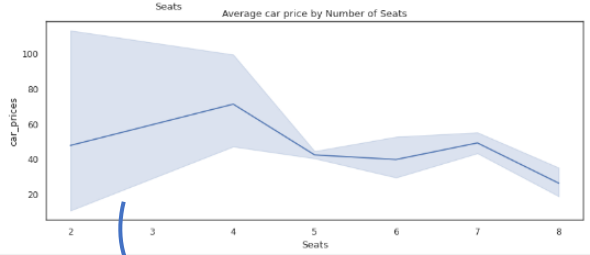
座位數的比例關係

```
#Seats
plt.figure(figsize=(12, 10), dpi=80)

plt.subplot(2, 1, 2)
ax1= sns.lineplot(x="Seats", y='car_prices', data=df2)
ax1.set_title('Average car price by Number of Seats')

plt.subplot(2, 2, 1)
ax2 = sns.barplot(x="Seats", y='car_prices', data=df2)
ax2.set_title('Average car price by Number of seats')

seat = df2["Seats"].copy()
count = seat.value_counts().sort_index(ascending=False)
#print(count)
plt.subplot(2, 2, 2)
plt.pie(x=count,autopct='%.1f%%',labels = ["8","7","6","5","4","2"])
plt.title('Transmission type popularity')
```

```
Text(0.5, 1.0, 'Transmission type popularity')
```





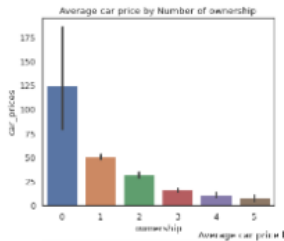


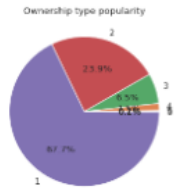
座位數對價格影響

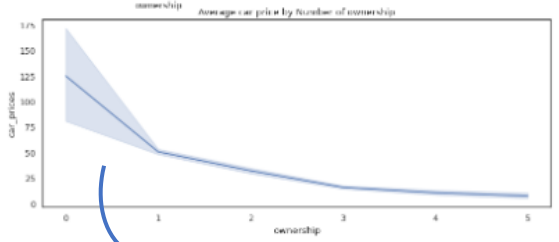
使用者次數比例關係

```
#ownership
plt.figure(figsize=(12, 10), dpi=80)

plt.subplot(2, 1, 2)
pv2=sns.lineplot(x="ownership", y='car_prices', data=df2)
pv2.set_title('Average car price by Number of ownership')

plt.subplot(2, 2, 1)
ax = sns.barplot(x="ownership", y='car_prices', data=df2)
ax.set_title('Average car price by Number of ownership')

ownership = df2["ownership"].copy()
count = ownership.value_counts().sort_index(ascending=False)
print(count)
plt.subplot(2, 2, 2)
plt.pie(x=count,autopct='%.1f%%',labels = ["5","4","3","2","1","0"])
plt.title('Ownership type popularity')
```

```
5     12
4     84
3    359
2   1314
1   3724
0      7
Name: ownership, dtype: int64

Text(0.5, 1.0, 'Ownership type popularity')
```





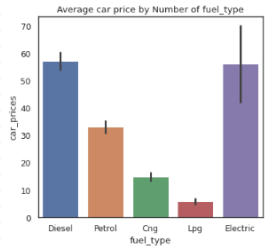


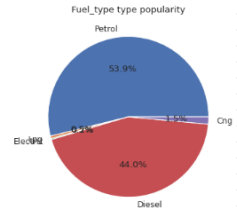
使用者次數對價格影響

燃料方式比例關係

```
#fuel_type
plt.figure(figsize=(12, 5), dpi=80)
plt.subplot(1, 2, 1)
ax = sns.barplot(x="fuel_type", y='car_prices', data=df2)
ax.set_title('Average car price by Number of fuel_type')

fuel = df2["fuel_type"].copy()
count = fuel.value_counts().sort_index(ascending=False)
print(count)
plt.subplot(1, 2, 2)
plt.pie(x=count,autopct='%.1f%%',labels = ["Petrol","Lpg","Electric","Diesel","Cng"])
plt.title('Fuel_type type popularity')
```

```
Petrol      2962
Lpg           28
Electric      12
Diesel      2418
Cng           80
Name: fuel_type, dtype: int64

Text(0.5, 1.0, 'Fuel_type type popularity')
```

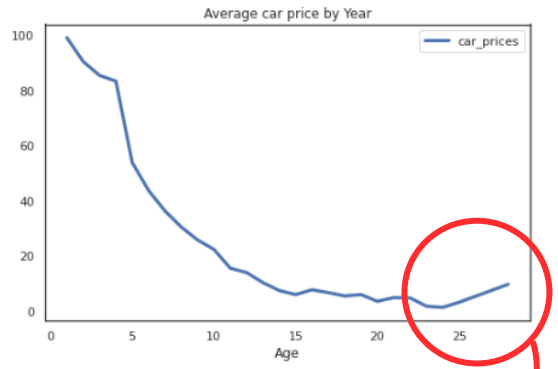




年份對價格的關係

```
pv1=pd.pivot_table(df2, index=['Age'],values = ['car_prices'],aggfunc = 'mean')
pv1.plot(kind='line',linewidth=3,figsize=(8,5),title='Average car price by Year')
```

```
<AxesSubplot:title={'center':'Average car price by Year'}, xlabel='Age'>
```



車齡越高理論上價格會越低

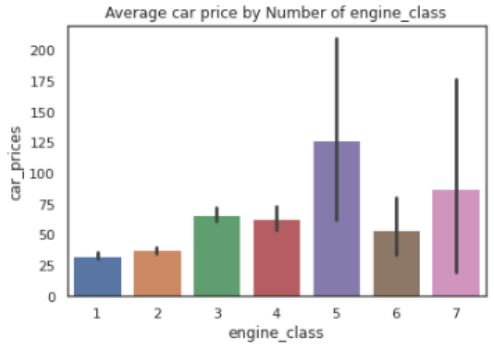
但在高年份的車資，金額卻有往上的趨勢

表示這部分可能需要深入了解

排氣量對價格的關係

```
ax = sns.barplot(x="engine_class", y='car_prices', data=df2)
ax.set_title('Average car price by Number of engine_class')
```

```
Text(0.5, 1.0, 'Average car price by Number of engine_class')
```



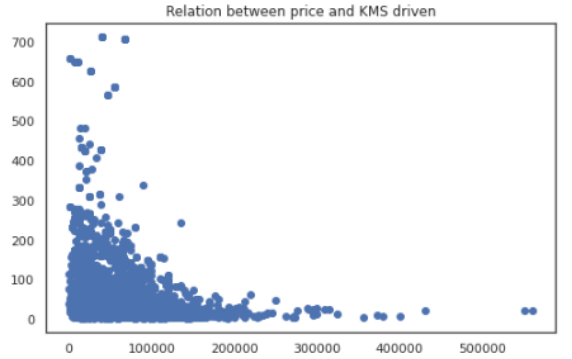
在3000~4200cc 這個級距

可能有較高的價格

里程對價格的關係

```
plt.figure(figsize=(8,5))
plt.title('Relation between price and KMS driven')
plt.scatter(df2.kms_driven,df2.car_prices,color="b")
```

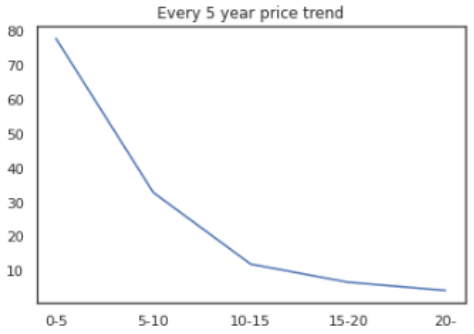
```
<matplotlib.collections.PathCollection at 0x7f7c083f5d10>
```



年份對價格的衰退關係(探討保值的問題)

```
P1 = df[df['Age'] <= 5]['car_prices'].mean()
P2 = df[(df['Age']>5) & (df['Age']<=10)]['car_prices'].mean()
P3 = df[(df['Age']>10) & (df['Age']<=15)]['car_prices'].mean()
P4 = df[(df['Age']>15) & (df['Age']<=20)]['car_prices'].mean()
P5 = df[df['Age']>20]['car_prices'].mean()
Plist = [P1,P2,P3,P4,P5]
Rlist = ["0-5","5-10","10-15","15-20","20-"]
pv2=sns.lineplot(x=Rlist, y=Plist)
pv2.set_title('Every 5 year price trend')
```

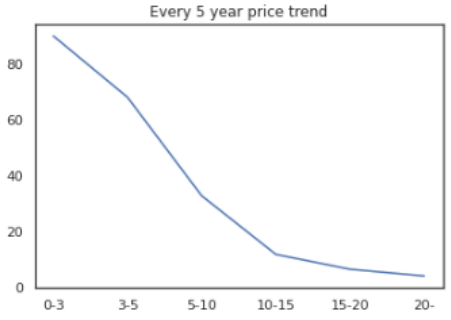
```
Text(0.5, 1.0, 'Every 5 year price trend')
```



可以看出在前五年，價格衰退幅度最大

```
P1 = df[df['Age'] <= 3]['car_prices'].mean()
P2 = df[(df['Age']>3) & (df['Age']<=5)]['car_prices'].mean()
P3 = df[(df['Age']>5) & (df['Age']<=10)]['car_prices'].mean()
P4 = df[(df['Age']>10) & (df['Age']<=15)]['car_prices'].mean()
P5 = df[(df['Age']>15) & (df['Age']<=20)]['car_prices'].mean()
P6 = df[df['Age']>20]['car_prices'].mean()
Plist = [P1,P2,P3,P4,P5,P6]
Rlist = ["0-3","3-5","5-10","10-15","15-20","20-"]
pv2=sns.lineplot(x=Rlist, y=Plist)
pv2.set_title('Every 5 year price trend')
```

```
Text(0.5, 1.0, 'Every 5 year price trend')
```



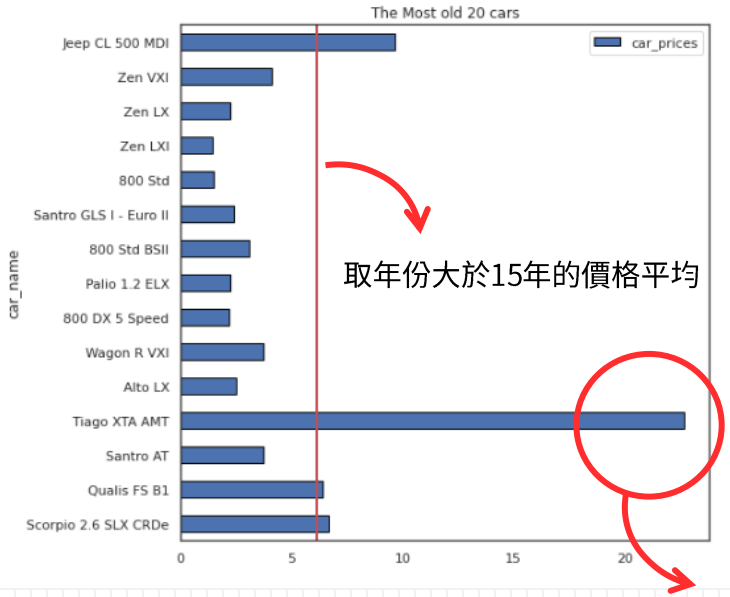
拆分出0-3年、3-5年，

可以看出在車齡為0-3區間，價格衰減幅度相對小

列表年份最高的資料

```
df3 = df2.copy()
df3.sort_values(by=['Age'],inplace = True)
old5=df3.iloc[-20:,:]

top5=pd.pivot_table(old5,index='car_name',values='car_prices',sort=False)
top5.plot(kind='barh',figsize=(8,8),edgecolor = 'black',title='The Most old 20 cars')

price = df2.copy().sort_values('car_prices')
plt.axvline(x = df2[df2['Age']>=15]['car_prices'].mean(),linewidth=2, color='r')
```

```
<matplotlib.lines.Line2D at 0x7f7c0907a510>
```



根據網路資料顯示，這一台車的發表年份應該是2017年

特別算出資料中2017年車的平均車價 25.49 與異常資料的車價22.69 相近

因此，特別修正此筆資料，更正他的車齡。

```
print(df[(df["manufacture"]==2017) & (df["Brand_lv"] == 0)]["car_prices"].mean())
df[df["car_name"]=="Tiago XTA AMT"]
```

```
25.49777559055118
```

	Unnamed: 0	Brand	Brand_lv	car_name	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	man
4633	4633	Tata	0	Tiago XTA AMT	22.69	10K	2640	Petrol	Automatic	1	200

```
df2[df2["car_name"] == "Tiago XTA AMT"]["Age"] = df2[df2["car_name"] == "Tiago XTA AMT"]["Age"]-15
df2[df2["car_name"] == "Tiago XTA AMT"] = df2[df2["car_name"] == "Tiago XTA AMT"].replace(21, 6)
df2[df2["car_name"] == "Tiago XTA AMT"]
```

	Brand_lv	car_name	car_prices	multiply NT	kms_driven	fuel_type	transmission	ownership	Age	engine_class	Sea
4633	0	Tiago XTA AMT	22.69	10K	2640	Petrol	Automatic	1	6	2	5

呼應年份對車價的曲線，確實有異常值

拆分資料集

```
# Split data to train and test:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y ,test_size=0.25, random_state=
1)
```

匯入模組

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score,GridSearchCV


Regression=[ ]
Regression.append(RandomForestRegressor())
Regression.append(DecisionTreeRegressor())
Regression.append(LinearRegression())

Regression
```

交叉驗證

選擇模型

```
cv_results=[]

LR_model = LinearRegression(n_jobs=-1)
cv_results.append(cross_val_score(LR_model,X_train,y_train,cv=10,n_jobs=-1))


randomForestModel = RandomForestRegressor(n_estimators=100, criterion = 'mse')
cv_results.append(cross_val_score(randomForestModel,X_train,y_train,cv=10,n_jobs=-1))


decisionTreeModel = DecisionTreeRegressor(max_depth=4, splitter='best', random_state=42)
cv_results.append(cross_val_score(decisionTreeModel,X_train,y_train,cv=10,n_jobs=-1))


cv_results
```

```
[array([0.5394516 , 0.41487987, 0.39776766, 0.50158566, 0.42269799,
        0.39561387, 0.44830671, 0.46311598, 0.41100744, 0.4880446 ]),
 array([0.79767139, 0.85354635, 0.83595641, 0.87258154, 0.8327886 ,
        0.8828843 , 0.87732571, 0.83738766, 0.81691255, 0.78655707]),
 array([0.67327155, 0.71271987, 0.67647237, 0.67946329, 0.57162285,
        0.57606692, 0.66082801, 0.58281966, 0.54949619, 0.82526434])]
```

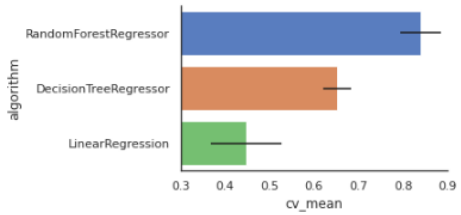
```
cv_means=[]
cv_std=[]
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cvResDf=pd.DataFrame({'cv_mean': cv_means,
                      'cv_std' : cv_std,
                      'algorithm':['LinearRegression','RandomForestRegressor','DecisionTre
eRegressor']})
cvResDf
```

	cv_mean	cv_std	algorithm
0	0.448247	0.046303	LinearRegression
1	0.839361	0.031192	RandomForestRegressor
2	0.650803	0.079433	DecisionTreeRegressor

將結果以表格、圖表展示

```
#Model Selection
cvResFacet=sns. FacetGrid(cvResDf . sort_values(by='cv_mean', ascending=False),sharex=False,
                          sharey=False, aspect=2)
cvResFacet.map(sns.barplot, 'cv_mean', 'algorithm' ,**{'xerr' :cv_std},
               palette='muted')
cvResFacet.set(xlim=(0.3,0.9))
cvResFacet.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f7c01f80b90>
```



使用模型進行預測並計算成績

```
rFModel = RandomForestRegressor(n_estimators=100, criterion = 'mse')
# 使用訓練資料訓練模型
rFModel.fit(X_train,y_train)
```

```
RandomForestRegressor(criterion='mse')
```

```
score = rFModel.score(X_train,y_train)
score
```

```
0.9777049298772856
```

```
from sklearn.metrics import mean_squared_error #均方误差
from sklearn.metrics import mean_absolute_error #平方绝对误差
from sklearn.metrics import r2_score#R square

y_pred = rFModel.predict(X_test)
rscore=r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print("rscore:",rscore,"mse:",mse,"mae:",mae)
```

```
rscore: 0.720344292867565 mse: 1379.6932845396402 mae: 13.389703901616159
```

OLS Regression Results

Dep. Variable:	car_prices	R-squared (uncentered):	0.585
Model:	OLS	Adj. R-squared (uncentered):	0.584
Method:	Least Squares	F-statistic:	724.2
Date:	Fri, 06 Jan 2023	Prob (F-statistic):	0.00
Time:	07:09:23	Log-Likelihood:	-22422.
No. Observations:	4125	AIC:	4.486e+04
Df Residuals:	4117	BIC:	4.491e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Brand_lv	40.7745	1.520	26.826	0.000	37.795	43.754
kms_driven	-5.056e-05	2.45e-05	-2.063	0.039	-9.86e-05	-2.51e-06
fuel_type	4.5848	1.590	2.884	0.004	1.468	7.702
transmission	38.1282	2.475	15.407	0.000	33.276	42.980
ownership	-2.1298	1.359	-1.567	0.117	-4.794	0.534
Age	-3.9646	0.273	-14.545	0.000	-4.499	-3.430
engine_class	3.0647	0.897	3.415	0.001	1.306	4.824
Seats	8.6461	0.537	16.095	0.000	7.593	9.699

Omnibus:	4528.663	Durbin-Watson:	2.037
Prob(Omnibus):	0.000	Jarque-Bera (JB):	422546.461
Skew:	5.544	Prob(JB):	0.00
Kurtosis:	51.327	Cond. No.	2.34e+05

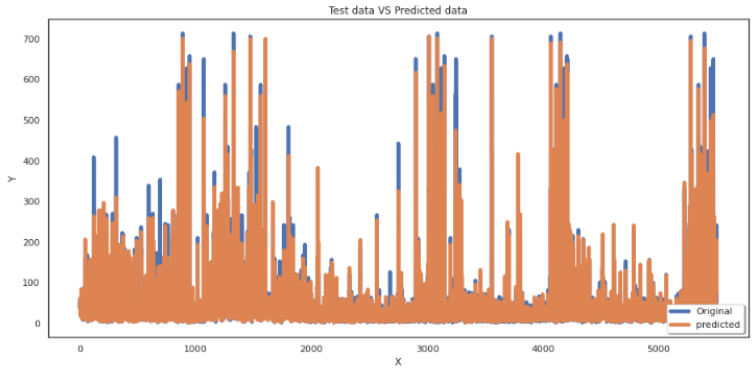
Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 2.34e+05. This might indicate that there are strong multicollinearity or other numerical problems.

判別預測結果和實際值

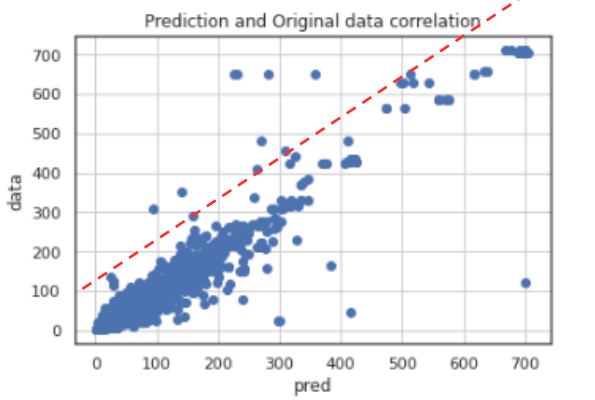
1. 大部分預測結果有貼近實際值
2. 在價格介於200-600間的時候
   預測偏離率較大

```
df2['Price_prediction']=rFModel.predict(df_X)
```

```
prd=df2['Price_prediction']
x_ax = range(len(df_y))
plt.figure(figsize=(15,7))
plt.plot(x_ax, df_y,linewidth = '4.5', label="Original")
plt.plot(x_ax, prd, linewidth = '4.5', label="predicted")
plt.title("Test data VS Predicted data")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='lower right',fancybox=True, shadow=True)
plt.show()
```




```
plt.scatter(df2['Price_prediction'],df2['car_prices'],color="b",)
plt.title('Prediction and Original data correlation')
plt.xlabel("pred")
plt.ylabel("data")
plt.grid()
```



按照原始預測，品牌對於價格的影響幅度較大，由結果可以判斷，在對品牌價值的分類上，可能在中高階級的品牌，需要做更仔細的區分。

使用beautifulsoup 爬取 YAHOO汽車 中古車版的資料

1. 目標爬取頁面中
   相同於紅框區塊的資訊

```python
import requests
from bs4 import BeautifulSoup
import requests
import string

column = ['車型','出廠時間','行駛里程','燃料排氣','顏色','變速系統','傳動系統','乘坐人數','所在地','售價']
df_tw = pd.DataFrame(data = detail, columns = column )

def getcardetails(title,link,c):
    url = link
    res = requests.get(url)
    soup = BeautifulSoup(res.text, "html.parser")
    content = soup.find_all("div",class_= 'text-bold',limit = 8)
    price = soup.find_all("span",class_= 'abc-article__price__num text-2xl')
    count = 0
    for i,n in enumerate (content[0:9]):
        #print(n)
        s = (n.text.strip().replace(" ",""))
        df_tw.loc[c,column[i+1]] = s
        df_tw.loc[c,'售價'] = price[0].text.strip().replace(" ","")
        if i == 0 :
            df_tw.loc[c,column[i]] = title
        else:
            continue

        #print(s)
    return df_tw
```

```python
url ='https://autos.yahoo.com.tw/used-cars/'

res = requests.get(url)
soup = BeautifulSoup(res.text, "html.parser")
limit = 15
count = 0
content = soup.find_all("div",class_= 'related-content',limit = limit)
#print(content[0].find_all('a')[1].get('title'))


for n in range(limit):
    for r in range(5):
        link = content[n].find_all('a')[r].get('href')
        title = content[n].find_all('a')[r].get('title')
        getcardetails(title,link,count)
        count +=1

df_tw
```

2. 從紅框內的資訊
   取得車子的車型以及該車的連結

3. 由於版面上的資訊過於精簡
   特別寫一函式 getcardetails()
   進到各連結中取的汽車完整資訊

	車型	出廠時間	行駛里程	燃料排氣	顏色	變速系統	傳動系統	乘坐人數	所在地	售價
0	2012 M-Benz 賓士 C-Class Sedan	2012年03月	166,718公里	汽油/1.8L	黑色	手自排	後輪驅動	5人	高雄市	68.8萬
1	2017 M-Benz 賓士 C-Class Estate	2017年11月	53,000公里	汽油/3.0L	白色	手自排	四輪驅動	5人	新北市	248萬
2	2008 M-Benz 賓士 C-Class Sedan	2008年05月	100,000公里	汽油/6.0L	黑色	手自排	後輪驅動	5人	彰化縣	53.8萬
3	2016 M-Benz 賓士 C-Class Coupe	2016年05月	56,500公里	汽油/2.0L	白色	手自排	後輪驅動	4人	台南市	158萬
4	2014 M-Benz 賓士 C-Class Sedan	2014年09月	87,165公里	汽油/3.0L	白色	手自排	後輪驅動	5人	新北市	139萬
...	...	...	...	...	...	...	...	...	...	...
70	2022 Suzuki 鈴木 Jimny	2022年12月	28公里	汽油/1.5L	其他	自排	四輪驅動	4人	高雄市	99.8萬
71	2022 Suzuki 鈴木 Jimny	2022年12月	27公里	汽油/1.5L	灰色	自排	四輪驅動	4人	台中市	98萬
72	2022 Suzuki 鈴木 Jimny	2022年12月	10公里	汽油/1.5L	綠色	自排	四輪驅動	4人	台中市	98.8萬
73	2022 Suzuki 鈴木 Jimny	2022年12月	43公里	汽油/1.5L	綠色	自排	四輪驅動	4人	彰化縣	95.9萬
74	2022 Suzuki 鈴木 Jimny	2022年12月	11公里	汽油/1.5L	綠色	自排	四輪驅動	4人	新北市	98.8萬

4. 將爬取的資料存成一個Dataframe
   並存成csv檔，以便後續使用

確認爬蟲資料與模型資料的特徵值

```
[77]  1  display(df_tw.head())
      2  display(X_test.head())
```

	車型	出廠時間	行駛里程	燃料排氣	顏色	變速系統	傳動系統	乘坐人數	所在地	售價
0	2012 M-Benz 賓士 C-Class Sedan	2012年03月	166,718公里	汽油/1.8L	黑色	手自排	後輪驅動	5人	高雄市	68.8萬
1	2017 M-Benz 賓士 C-Class Estate	2017年11月	53,000公里	汽油/3.0L	白色	手自排	四輪驅動	5人	新北市	248萬
2	2008 M-Benz 賓士 C-Class Sedan	2008年05月	100,000公里	汽油/6.0L	黑色	手自排	後輪驅動	5人	彰化縣	53.8萬
3	2016 M-Benz 賓士 C-Class Coupe	2016年05月	56,500公里	汽油/2.0L	白色	手自排	後輪驅動	4人	台南市	158萬
4	2014 M-Benz 賓士 C-Class Sedan	2014年09月	87,165公里	汽油/3.0L	白色	手自排	後輪驅動	5人	新北市	139萬

	Brand_lv	kms_driven	fuel_type	transmission	ownership	Age	engine_class	Seats
4420	0	140000	1	0	2	11	1	5
2266	0	142366	1	0	1	11	1	5
3694	0	70000	0	0	2	18	3	5
4716	0	27924	1	0	1	3	3	5
5441	0	79718	1	0	1	14	2	5

比對後發現

大部分特徵值可以互相匹配

唯獨 “ownership” 無法在yahoo汽車中取得

而 “傳動系統” 、” 所在地” 則是不存在於原始資料集

接下來要做新的資料處理，

並修正訓練資料，重新做模型訓練

新資料的資料預處理

Before

	車型	出廠時間	行駛里程	燃料排氣	顏色	變速系統	傳動系統	乘坐人數	所在地	售價
0	2012 M-Benz 賓士 C-Class Sedan	2012年03月	166,718公里	汽油/1.8L	黑色	手自排	後輪驅動	5人	高雄市	68.8萬
1	2017 M-Benz 賓士 C-Class Estate	2017年11月	53,000公里	汽油/3.0L	白色	手自排	四輪驅動	5人	新北市	248萬
2	2008 M-Benz 賓士 C-Class Sedan	2008年05月	100,000公里	汽油/6.0L	黑色	手自排	後輪驅動	5人	彰化縣	53.8萬
3	2016 M-Benz 賓士 C-Class Coupe	2016年05月	56,500公里	汽油/2.0L	白色	手自排	後輪驅動	4人	台南市	158萬
4	2014 M-Benz 賓士 C-Class Sedan	2014年09月	87,165公里	汽油/3.0L	白色	手自排	後輪驅動	5人	新北市	139萬

After

	Brand	Brand_lv	car_name	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	Mercedes-Benz	2	C-Class Sedan	166718	0	1	11	2	5
1	Mercedes-Benz	2	C-Class Estate	53000	0	1	6	4	5
2	Mercedes-Benz	2	C-Class Sedan	100000	0	1	15	7	5
3	Mercedes-Benz	2	C-Class Coupe	56500	0	1	7	3	4
4	Mercedes-Benz	2	C-Class Sedan	87165	0	1	9	4	5

將原始的資料集，利用split, strip, map, astype指令轉換內容以及資料型態，最後drop不需要的欄位完成最終的台灣二手汽車資料預測集

Final

	Brand_lv	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	2	166718	0	1	11	2	5
1	2	53000	0	1	6	4	5
2	2	100000	0	1	15	7	5
3	2	56500	0	1	7	3	4
4	2	87165	0	1	9	4	5
...	...	...	...	...	...	...	...
57	0	28	0	1	1	2	4
58	0	27	0	1	1	2	4
59	0	10	0	1	1	2	4
60	0	43	0	1	1	2	4
61	0	11	0	1	1	2	4

62 rows × 7 columns

建立新的模型

```
df_X = df_train.copy().drop(columns = ["car_prices","ownership"],axis = 1)
df_y = df_train["car_prices"].copy()

display(df_X.head())
display(df_twFinal.head())
```

	Brand_lv	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	1	86226	1	0	6	3	5
1	0	13248	0	1	2	2	5
2	0	60343	0	1	7	4	5
3	0	26696	0	1	5	1	5
4	1	69414	0	0	7	1	5

	Brand_lv	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	2	166718	0	1	11	2	5
1	2	53000	0	1	6	4	5
2	2	100000	0	1	15	7	5
3	2	56500	0	1	7	3	4
4	2	87165	0	1	9	4	5

移除訓練資料集中不需要的欄位

並確認與台灣二手汽車資料預測集 的特徵值相符

```
[106]
# Split data to train and test:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y ,test_size=0.25, random_state=1)
```

```
rFModel = RandomForestRegressor(n_estimators=100, criterion = 'friedman_mse')
# 使用訓練資料訓練模型
rFModel.fit(X_train,y_train)
```

```
RandomForestRegressor(criterion='friedman_mse')
```

```
[108]
score = rFModel.score(X_train,y_train)
score
```

```
0.9748132436011652
```

拆分資料集，

並使用最初模型篩選，效果最好的演算法RandomForestRegressor

重新訓練模型，並得到決定係數為0.97成績

建立新的模型

```
df_X = df_train.copy().drop(columns = ["car_prices","ownership"],axis = 1)
df_y = df_train["car_prices"].copy()

display(df_X.head())
display(df_twFinal.head())
```

	Brand_lv	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	1	86226	1	0	6	3	5
1	0	13248	0	1	2	2	5
2	0	60343	0	1	7	4	5
3	0	26696	0	1	5	1	5
4	1	69414	0	0	7	1	5

	Brand_lv	kms_driven	fuel_type	transmission	Age	engine_class	Seats
0	2	166718	0	1	11	2	5
1	2	53000	0	1	6	4	5
2	2	100000	0	1	15	7	5
3	2	56500	0	1	7	3	4
4	2	87165	0	1	9	4	5

移除訓練資料集中不需要的欄位

並確認與台灣二手汽車資料預測集 的特徵值相符

```
[106]
# Split data to train and test:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y ,test_size=0.25, random_state=1)
```

```
rFModel = RandomForestRegressor(n_estimators=100, criterion = 'friedman_mse')
# 使用訓練資料訓練模型
rFModel.fit(X_train,y_train)
```

```
RandomForestRegressor(criterion='friedman_mse')
```

```
[108]
score = rFModel.score(X_train,y_train)
score
```

```
0.9748132436011652
```

拆分資料集，

並使用最初模型篩選，效果最好的演算法RandomForestRegressor

重新訓練模型，並得到決定係數為0.97成績

```
[110]
import joblib
```

```
[111]
joblib.dump(rFModel,'rFModel.pkl')
```

```
['rFModel.pkl']
```

```
clf2 = joblib.load('rFModel.pkl')
```

儲存模型，可於未來繼續使用

使用模型預測台灣二手汽車資料

```
result = clf2.predict(df_twFinal)
result
```

```
loss = pd.DataFrame(data = [],columns=['predict_price','market_price'])
loss["predict_price"] = result
loss["market_price"] = tw_price
#print(loss[loss["market_price"].isnull()])
loss["loss"] = loss["predict_price"] - loss["market_price"]
loss
```

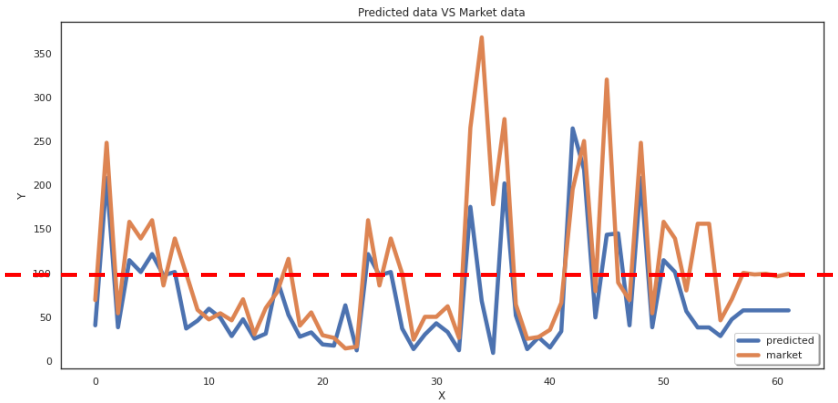
	predict_price	market_price	loss
0	40.0970	68.8	-28.7030
1	207.8660	248.0	-40.1340
2	37.9223	53.8	-15.8777
3	114.2926	158.0	-43.7074
4	100.8037	139.0	-38.1963
...	...	...	...
57	57.1297	99.8	-42.6703
58	57.1297	98.0	-40.8703
59	57.1297	98.8	-41.6703
60	57.1297	95.9	-38.7703
61	57.1297	98.8	-41.6703

62 rows × 3 columns

利用預測結果與市場上的真實資料
製作成一個Dataframe
另外求得兩者的誤差值(預測值-實際值)

預測與實際資料的折線圖(一)

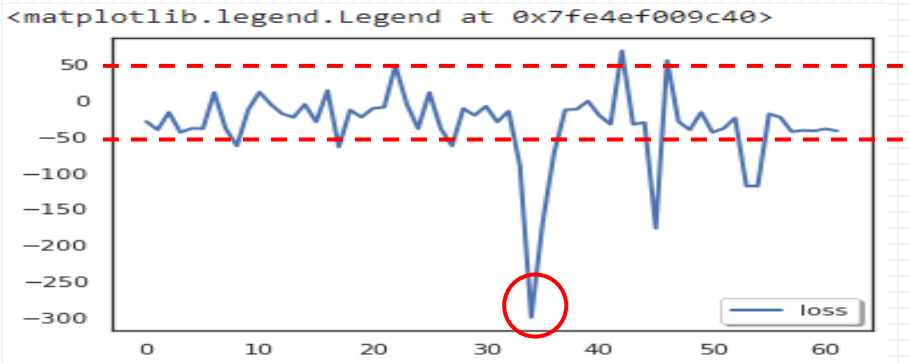
```
x_ax = range(len(result))
plt.figure(figsize=(15,7))
plt.plot(x_ax, result,linewidth = '4.5', label="predicted")
plt.plot(x_ax, tw_price, linewidth = '4.5', label="market")
plt.title("Predicted data VS Market data")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='lower right',fancybox=True, shadow=True)
plt.show()
```



由圖表(一)兩條線的走勢來看
模型在判斷的準確度還算不差
不僅走勢相合，部分線段還重和

相對來說
低價位的二手車預測結果比高價位的優異

誤差值的折線圖(二)



由圖表(二)來看
大部分的預測結果都在+-50之內
針對誤差值大的資料，則可作為後續改良的依據

分析預測誤差大的項目

```
[162]  1  loss[loss["loss"] < -250]
```

	predict_price	market_price	loss
34	67.7877	368.0	-300.2123

```
[165]  1  df_tw.iloc[34,:]
```

```
車型          2017 Porsche 保時捷 Panamera
出廠時間                        2017年05月
行駛里程                       86,000公里
燃料排氣                        汽油/3.0L
顏色                              灰色
變速系統                          自手排
傳動系統                        後輪驅動
乘坐人數                          4人
所在地                            新北市
售價                             368萬
Name: 41, dtype: object
```

```
1  df_twNew.iloc[34,:]
```

```
Brand           Porsche
Brand_lv              2
car_name       Panamera
kms_driven        86000
fuel_type             0
transmission          0
Age                   6
engine_class          4
Seats                 4
Name: 34, dtype: object
```

1. 取得誤差大的該筆資料

```
1  loss[33:37]
```

	predict_price	market_price	loss
33	175.0454	265.0	-89.9546
34	67.7877	368.0	-300.2123
35	8.6838	178.0	-169.3162
36	201.7659	275.0	-73.2341

2.

找到條件相近的資料

初步先找出同為同一家車廠的資料

發現34、35的誤差值明顯大很多

33	Porsche	2	Macan	28936	0	0	4	3	5
34	Porsche	2	Panamera	86000	0	0	6	4	4
35	Porsche	2	Macan	70000	0	0	9	4	5
36	Porsche	2	Macan	30000	0	0	4	3	5

3. 首先，從紅色框來看，雖為同一廠牌出產的汽車，但在模型訓練中，並未對車款進行特徵處理

進而影響到跑車、轎車、休旅車…等等車型對價格之影響，所以應該要嘗試將車型納入模型訓練中。

再者，從橘框及藍框來看，行駛里程以及車齡，對於車價的預估有顯著的影響，以33、35、36資料來看同為 Porsche Macan 車型，當里程數及車齡較大時，價格明顯被低估。

總結

1. 現在的模型，針對品牌價值及車價較平價之車款預測效果較好

可能因為品牌價值較平價之車款，相對來說不同車型價格差異沒有這麼大，因此預測的誤差也大為降低，另外，平價車款可能相對保值性接近，較能依照目前模型的特徵值(里程，車齡)作為主要價格預測，而在處理高檔汽車時，則無法有相應價值的預測結果

2. 需要增加一些特徵來優化模型的預測能力

從預測的結果判斷，最顯著的是車款樣式，特別是在高級車中，價差會特別顯著，因此應該要考量的車款的條件，另外，結果中也有發現，相似條件下的資料中，有的行駛里程數較高，但卻有較好的價格表現，因此可以發現，在影響車價的特徵中，還有一些條件是沒考慮到的，像是車色、安全配備等，也許都是可以作為對後續模型優化的調整方向。