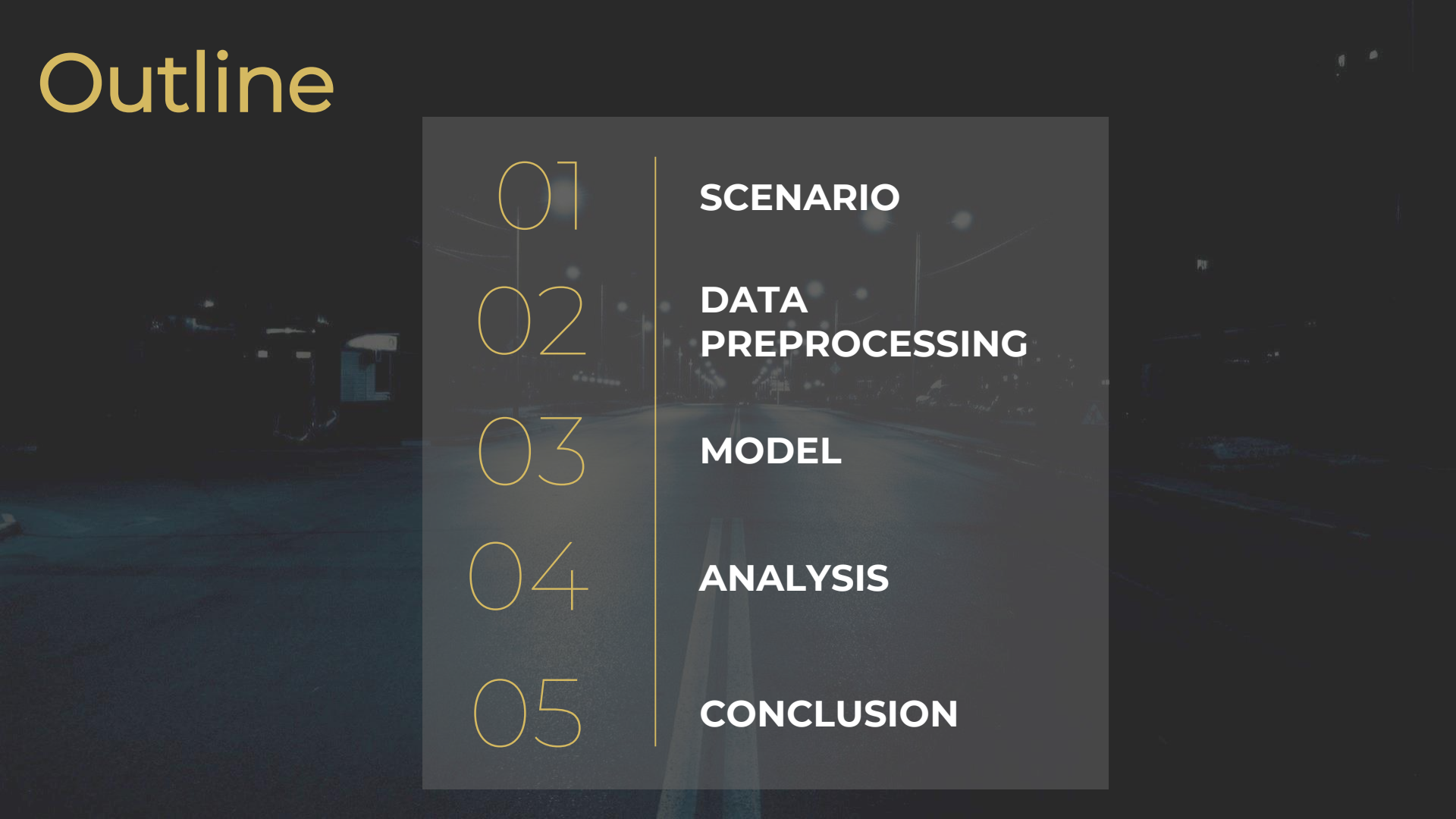
Project2　Group 8
Kaggle二手汽車售價預測

李勁緯、廖文辰、黃鉦程 、熊夆峻

2021.12.17 Intelligent Integration of Enterprise

Outline



01 SCENARIO

PROBLEM DEFINITION

背景介紹

- 因疫情影響，多數人財務緊縮。
- 新車價格逐年攀升。
- 印度人購買二手車之族群年齡有70%的比例為40歲以下。
- 2020年印度二手車市場的銷售量高達440萬輛。

5W1H

WHO

買二手車之買賣家

WHAT

為了能夠精準預估二手車的價格

WHEN

二手車交易時

WHY

以雙方皆可接受的價錢進行交易

WHERE

印度各城市為例

HOW

資料分析
深度學習

02
DATA-PREPROCESSING

Package used

Matplotlib

使用其Seaborn進階圖表繪製功能

Numpy

高階維度陣列與矩陣運算

Pandas

數據分析的data frame架構

Scikit-learn

提供許多機器學習的基本演算法

Dataset

Kaggle提供的資料集

Train.csv

6019筆資料

14項二手車特性

Variables

Train.csv的變數

特性	說明
Unnamed	編號
Name	汽車型號名稱
Location	汽車所在地點
Year	汽車出廠年份
Kilometers_Driven	已行駛里程數(公里)
Fuel_Type	燃油/燃料種類
Transimission	打檔方式
Owner_Type	汽車經過了幾手交易
Mileage	平均油耗
Engine	引擎規格
Power	馬力
Seats	汽車內的座位數
New_price	新車價格
Price	該汽車的價格

資料前處理

-刪除資料

```
(6019, 14)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         6019 non-null   int64
 1   Name               6019 non-null   object
 2   Location           6019 non-null   object
 3   Year               6019 non-null   int64
 4   Kilometers_Driven  6019 non-null   int64
 5   Fuel_Type          6019 non-null   object
 6   Transmission       6019 non-null   object
 7   Owner_Type         6019 non-null   object
 8   Mileage            6017 non-null   object
 9   Engine             5983 non-null   object
 10  Power              5983 non-null   object
 11  Seats              5977 non-null   float64
 12  New_Price          824 non-null    object
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 658.5+ KB
None
```

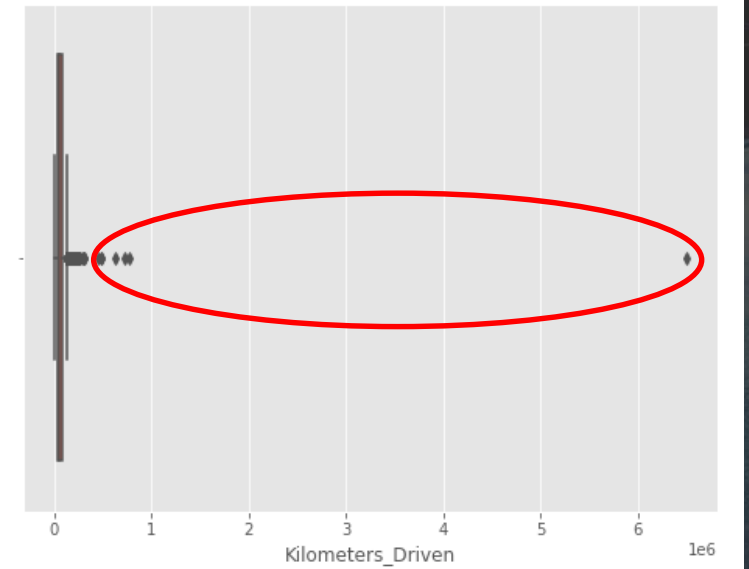
New_Price欄位中僅有824筆資料。將New_Price整欄刪除

```
train.drop('New_Price',  axis=1  ,  inplace  =  True)
```

資料前處理

-刪除資料

```
plt.figure(figsize=(8,6))
sns.boxplot(train['Kilometers_Driven'])
plt.show()
```



將Kilometers_Driven欄位中大於40萬公里之資料列刪除。

```
print(train.shape)
temp = train[train['Kilometers_Driven']>400000].index
train.drop(temp, axis = 0, inplace =True)
print(train.shape)
```

```
(6019, 12)
(6012, 12)
```

資料前處理

-填補空值

空值欄位	空值數量
Mileage	2
Engine	36
Power	36
Seats	42

對照車種及燃油種類，將「Mileage」「Engine」、「Power」、「Seats」四欄中的空值以手動的方式輸入

資料前處理

-分割數值和單位

```
engine_val = pd.DataFrame(columns=['Engine_value'])
engine_measure = pd.DataFrame(columns=['Engine_measure'])
train = pd.concat([train, engine_val, engine_measure], axis=1)

for i in train.index:
    value = train['Engine'][i].split()[0]
    measure = train['Engine'][i].split()[1]
    train['Engine_value'][i] = value
    train['Engine_measure'][i] = measure

train.drop('Engine', axis=1, inplace=True)
```

1

2

3

```
train['Engine_measure'].value_counts()

CC    6012
Name: Engine_measure, dtype: int64

train.drop('Engine_measure', axis=1, inplace=True)
```

4

以Engine為例

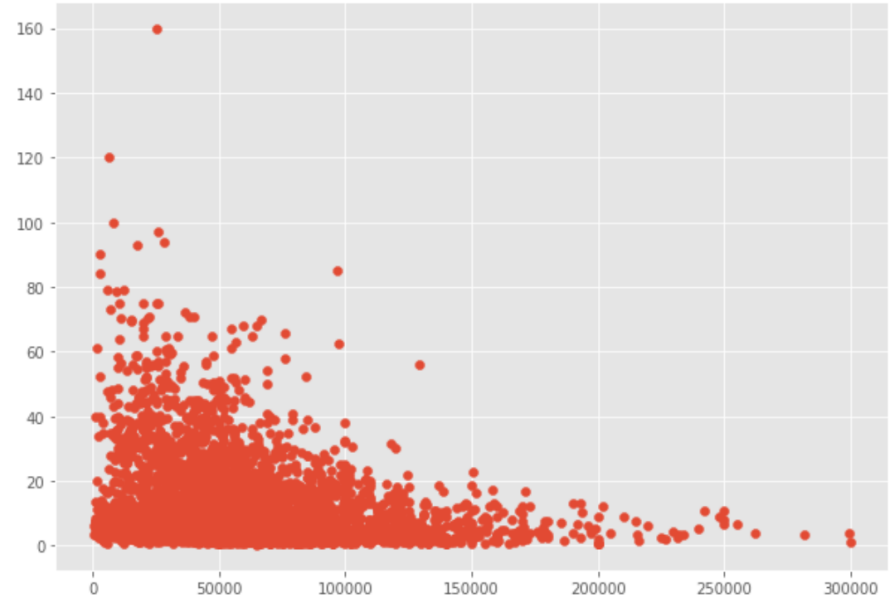
1. 新增兩欄
2. 加入至資料集中
3. 將原Engine（欄）刪除
4. 將「Engine_measure」移除

特徵提取

-公里數

```
plt.figure(figsize=(10,  7))
plt.scatter(train['Kilometers_Driven'],  train['Price'])
```

```
<matplotlib.collections.PathCollection at 0x7efe2b224390>
```



行駛公里數低於8萬公里的車價格較佳

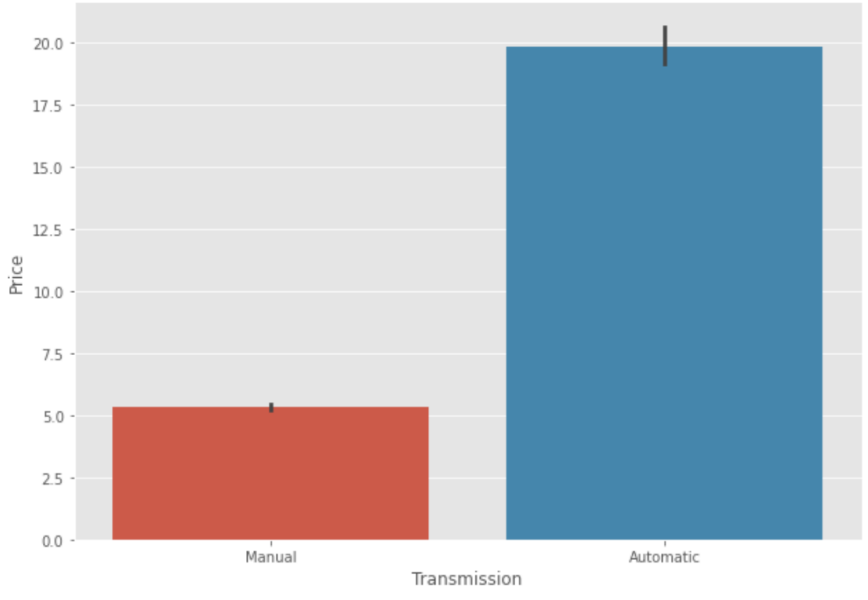
新增一欄以行駛公里數8萬為界線的資料

```
less_80k  =  pd.DataFrame(columns=['is_kilometers_driven_less_80k'])
train  =  pd.concat([train,  less_80k])


for  i  in  train.index:
        value  =  train['Kilometers_Driven'][i]
        if  value  <  80000:
                train['is_kilometers_driven_less_80k'][i]  =  1
        else:
                train['is_kilometers_driven_less_80k'][i]  =  0
```

特徵提取
-打檔方式

```
plt.figure(figsize=(10, 7))
sns.barplot('Transmission', 'Price', data=train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe228fe650>
```



自排的價格高，將此資料再新增一欄：是否為自排。

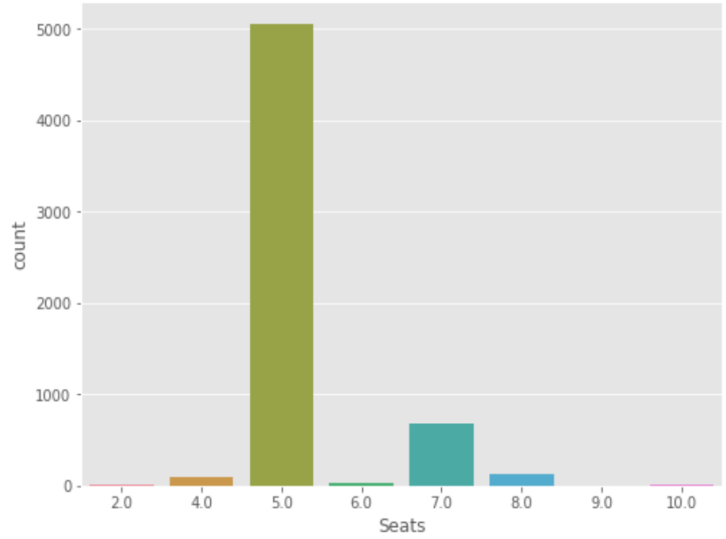
```
is_automatic = pd.DataFrame(columns=['is_automatic'])
train = pd.concat([train, is_automatic], axis=1)


for i in train.index:
        value = train['Transmission'][i]
        if value == 'Automatic':
                train['is_automatic'][i] = 1
        else:
                train['is_automatic'][i] = 0
```

特徵提取

-座位數

```
plt.figure(figsize=(8, 6))
sns.countplot('Seats', data=train)
```

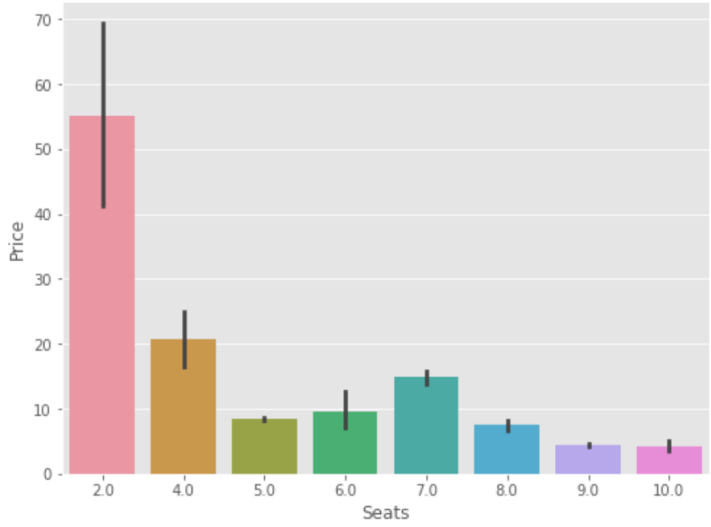
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe22624350>
```



兩人座的車子數量為佔比少。

```
plt.figure(figsize=(8, 6))
sns.barplot('Seats', 'Price', data=train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe2b60a810>
```



兩人座車子的價格高於其他種類。

新增一欄：是否為兩人座。

特徵提取

```
for  i  in  [  'is_kilometers_driven_less_80k',
               'is_diesel',  'is_automatic',  'is_2_seats',
               'Mileage_value',  'Engine_value','Power_value']:
      print(i)
      train[i]  =  train[i].astype('float32')
      test[i]  =  test[i].astype('float32')
```

將新增的欄位以及原本的「Mileage_value」、「Engine_value」及「Power_value」轉為浮點數

```
print(train.shape)
cat  =  train.select_dtypes(include='object')

for  i  in  cat.columns:
       le  =  LabelEncoder()
       train[i]  =  le.fit_transform(train[i])
```

使用 LabelEncoder 將特徵為「Object」的資料正規化，再將資料擬和

資料劃分與標準化

```
X_train, X_test, y_train, y_test = train_test_split(train, train_target, test_size=0.15, random_state=42)
```

Train

5109筆資料

15項二手車特性

Test

902筆資料

15項二手車特性

```
train_target = train['Price']
train.drop('Price', axis=1, inplace=True)
means = train.mean()
stds = train.std()
train = (train - means) / stds
```

將所有欄位資料標準化

處理後之資料欄位

將資料前處理完，經過重複資料刪除以及特徵提取新增後，總共有15個欄位。

特性	說明
Location	汽車所在地點
Year	汽車出廠年份
Kilometers_Driven	已行駛里程數(公里)
Fuel_Type	燃油/燃料種類
Transimission	打檔方式
Owner_Type	汽車經過了幾手交易
Seats	汽車內的座位數
Mileage_value	平均油耗

特性	說明
Mileage_measure	平均油耗單位
Engine_value	引擎數據
Power_value	傳動數據
is_kilometers_driven_less_80k	已行駛里程數低於8萬
is_diesel	是否為柴油
is_automatic	是否為自排
is_2_seats	是否為兩人座

03
MODEL STRUCTURE

模型介紹

XG Boost

XGboost 是採用特徵隨機採樣的技巧，和隨機森林一樣在生成每一棵樹的時候隨機抽取特徵，因此在每棵樹的生成中並不會每一次都拿全部的特徵參與決策。

優點：通過優化目標函數導出了增強樹，基本上可以用來解決幾乎所有可以寫出漸變的目標函數

缺點：樹木是按順序建造的，因此培訓通常需要更長時間

04
ANALYSIS

模型參數介紹

- booster：gbtree 樹模型/ dart(將深度神經網絡的 dropout 技術添加到 boosted 樹)/XGblinear。
- Eta: 學習率，範圍通常在0.01-0.2之間。
- gamma: 懲罰項係數，指定節點分裂所需的最小損失函數下降值。這個參數的值越大，算法越保守。這個參數的值和損失函數有關。
- Max_depth：為樹的最大深度，範圍通常在3-10之間。
- lambda: 權重的L2正則化項。這個參數是用來控制XGBoost的正則化部分的。
- alpha: 權重的L1正則化項。可以應用在很高維度的情況下，使得算法的速度更快。

尋找最佳參數
-XGboost Grid Search

透過網格搜索，將每一種參數組合實際下去Run，並且有五組驗證集，最後總共會有3240組。並將最好的參數顯示出來。

Booster	Gbtree, dart	2
Eta	0.01, 0.05, 0.1	3
Gamma	0, 0.5, 1	3
Max_depth	3, 5, 7, 9	4
Lambda	0.1, 1, 10	3
Alpha	0.1, 1, 10	3
CV		5
Total Run		3240

```
xgbb = xgb.XGBRegressor()

param_grid = {
        "booster": ['gbtree', 'dart'],
        "eta": [0.01, 0.05, 0.1],
        "gamma": [0, 0.5 ,1],
        "max_depth": [3, 5, 7, 9 ],
        "lambda": [0.1, 1, 10],
        "alpha": [0.1, 1, 10]
}

xgb_model = GridSearchCV(estimator=xgbb, cv=5, param_grid=param_grid,
xgb_model.fit(train, train_target)

print(xgb_model.best_score_)
print(xgb_model.best_estimator_.get_params())
```

最佳參數組合
-XGboost Grid Search Result

以下為最佳參數組合，MSE值為1.21，Accuracy達到0.87536。

最佳參數組合	
Booster	Gbtree
Eta	0.01
Gamma	0.5
Max_depth	7
Lambda	0.1
alpha	0.1
Test MSE	1.21
Accuracy	0.87536

05 CONCLUSION & PROSPECT

結論與未來展望

結論

- XGBoost準確度高，但尋找參數時間較長。
- 座位數、燃油種類、里程數為影響二手車售價的重要因素。

未來展望

- 加入更多二手車特性資料。
- 使用更多機器學習方法運用於模型中。
- 避免造假，以免影響預測準確度。