國　立　清　華　大　學
工業工程與工程管理學系

智慧化企業整合
Intelligent Integration of Enterprise

應用機器學習與深度學習於中古車價格預測之比較

第 7 組
110034538 許淨嵐
110034540 徐阡珊
110034553 李慕瑄

指導教授：　邱銘傳　博士

中 華 民 國 一 一 O 年 十 二 月

目錄

圖目錄

表目錄

壹、簡介

一、背景與動機

根據表 1顯示，我國每年中古車過戶數遠超過於新車領牌數，107年中古車過戶數是新車交易量約1.7倍，且近年中古車交易量皆維持在70至80萬輛，約莫新車市場的1.5至1.6倍。由於新車落地即折舊0.5至1成，且第一年又會再多折舊0.5至1成，因此愈來愈多民眾選擇購買較新之二手車輛或選購中古車。此外，新冠肺炎疫情也刺激通勤民眾之購車代步意願，提升了中古車市場行情，產業前景備受看好。

表 1、全台歷年新車和中古車銷售量 (參考資料：交通部)

年度	新車領牌數	中古車過戶數
100	261,491	678,985
101	365,837	801,366
102	378,462	814,893
103	423,831	848,517
104	420,770	828,487
105	439,581	751,963
106	444,624	759,002
107	435,114	741,488

二、研究目的

中古車買賣雖然可議價，但價錢通常由二手車商依據車況而定，若一般民眾缺乏車況背景知識，恐較無議價能力，導致交易價由車商單方面決定。為防止不公正交易並提升民眾議價能力，本研究期望透過中古車買賣歷史資料，發展一預測模型，在給定特定車況數據下，得出最符合該車輛之二手價錢。

三、5W1H 分析

What	中古車價格預測
When	賣家於價格刊登前、買家於議價前
Where	中古車商、中古車刊登平台
Why	達到資訊對稱、公平交易原則
Who	二手車買賣雙方
How	運用機器學習與深度學習方法並進行比較

貳、資料集介紹

本研究使用資料科學社群平台Kaggle上之公開資料集 “US Used cars dataset”，該資料來源為一美國汽車研究集購物網站CarGurus。原始資料為CSV檔，由表 2彙整顯示該檔案擁有66種資訊欄位，且 price 為本研究目標預測值。然而，圖 1顯示該66種資訊欄位可被分類為12種資訊類型。

表 2、US Used car dataset 欄位資料彙整表

US Used car dataset 欄位資料總表					
vin	dealer_zip	fuel_tank_volume	is_oemcpo	mileage	torque
back_legroom	description	fuel_type	latitude	model_name	transmission
bed	engine_cylinders	has_accidents	length	owner_count	transmission_display
bed_height	engine_displacement	height	listed_date	power	trimId
bed_length	engine_type	high_fuel_economy	listeing_color	price	trim_name
body_type	exterior_color	horsepower	listing_id	salvage	vehicle_damage_category
cabin	fleet	interior_color	longitude	savings_amount	wheel_system
city	frame_damaged	isCab	main_picture_url	seller_rating	wheel_system_display
city_fuel_economy	franchise_dealer	is_certified	major_options	sp_id	wheelbase
combine_fuel_economy	franchise_make	is_cpo	make_name	sp_name	width
daysonmarket	front_legroom	is_new	maximum_seating	theft_title	year

圖 1、欄位資料類別種類示意圖

此外，該資料集擁有超過300萬筆資料列，其中，透過程式碼檢視發現約有152萬筆資料包含缺失值，考量填補缺失值的方法探討需耗費較常研究時間成本，且該資料集之資料量足夠，本研究將直接剃除擁有缺失值之資料列。

參、研究方法

一、資料前處理

(一) 皮爾森積差相關分析 (Pearson correlation)

皮爾森相關分析用於探討兩變數之間的線性關係，其值介於-1到1之間，當兩變數之間的相關係數絕對值較大，則代表彼此相互共變的程度較大。一般而言，正值越大代表兩變數屬於強正相關，反之負值越大則代表屬於強負相關。其公式如下：

$$r(x,y) = \frac{COV(x,y)}{SxSy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

(二) 獨熱編碼 (One-Hot Encoding)

又稱為一位有效編碼，為解決離散型類別資料，運用獨熱編碼將類別特徵轉換為數值特徵，如圖 2所示，將分類變數作為二進位制向量的表示，以0或1進行表示。

ID	性別
1	男
2	女
3	男
4	男

獨熱編碼

ID	男	女
1	1	0
2	0	1
3	1	0
4	1	0

圖 2、獨熱編碼過程示意圖

(三) 主成分分析 (Principal Component Analysis)

PCA 是機器學習與統計學領域中被廣泛用於縮減數據維度以及去關聯性的線性降維方法，其做法為使用少數幾個變數的線性組合，以解釋原始數據大部分的變異。由於資料分析時常需要面對大量彼此可能相關的資料與變數，造成多元共線性的問題，為減少變數的共線性對分析模式造成的誤判與干擾，可藉由此方法將輸入資料轉換為數個線性獨立的變數。

降維是一種無監督學習，主要目的是「化繁為簡」，將原本高維的數據（比方說 N 維）重新以一個相較低維的形式表達（比方說 K 維，且 K<N）。目的在於 K 維的表徵（representation）具有代表性，能夠抓住原來 N 維數據的大部分特性，以至於在沒有損失什麼資訊的情況下，用更簡潔的方式呈現該組數據，進而對其本質有更深的理解。

二、機器學習演算法

(一) 線性迴歸 (Linear Regression)

線性迴歸是利用被稱為線性迴歸方程式的最小平方函數，建構一個或多個自變數和應變數關係的模型。這種方程式是一個或多個迴歸係數的模型參數的線性組合。在只有一個自變數的情況稱為簡單迴歸，大於一個自變數的情況則為多元迴歸（multivariable linear regression）。

線性迴歸是監督式學習的一種，會依照訓練資料找出一個最接近各點的線性方程式，並用這個函式去預測出新的資料(測試資料)應該在的位置，圖 3 顯示了線性迴歸的基礎圖示。

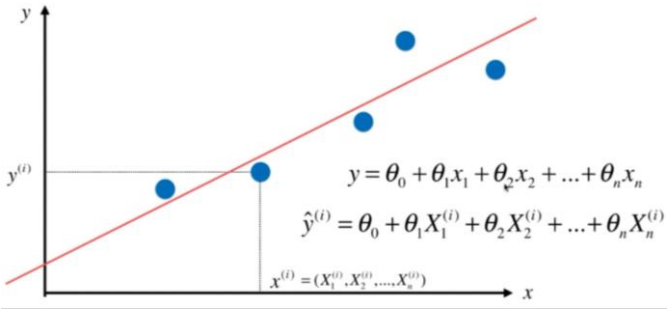


圖 3、線性迴歸基礎圖示

(二) 多層感知器迴歸 (Multi-layer Perceptron Regressor)

多層感知機（Multilayer Perceptron, MLP）也叫人工神經網路（Artificial Neural Network, ANN），除了輸入層及輸出層，它中間可以有多個隱藏層，最簡單的 MLP 只包含一個隱藏層，即三層的結構，因此深度神經網路可視為 MLP 的多隱藏層網路。

(三) 支援向量機 (Support Vector Machine, SVM)

SVM 是一種監督式演算法，利用統計風險最小化的原則來估計一個分類的超平面(Hyperplane)，即找到一組決策邊界(Decision boundary)的參數，讓兩個類別之間的間隔寬度最大化(所有在邊界上的點離得越遠越好)，並完美分割兩類資料。由於在邊界上的樣本稱為 Support vectors，因此該方法被稱為 Support vector machine。基礎的線性可分的集合如圖 4 所示；若非線性可分集合則可運用核函數(Kernel function) 來造出不可分的分割平面，如圖 5 所示。

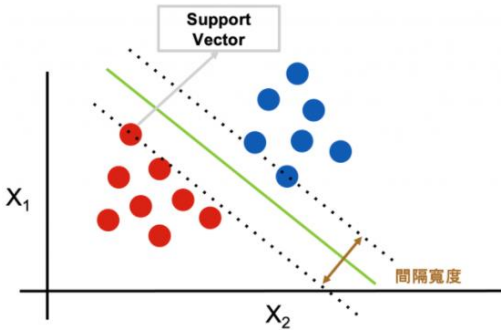


圖 4、線性可分集合的支持向量機

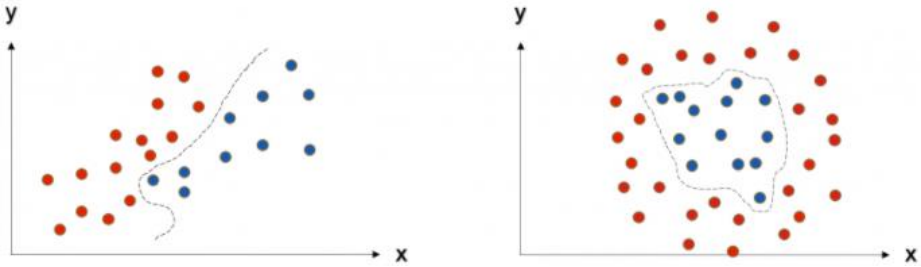


圖 5、非線性可分集合示意圖

(四) 隨機森林 (Random Forest)

隨機森林是一個包含多棵決策樹的模型，透過並行且獨立地分別訓練一系列的決策樹，降低決策樹過度擬合的風險，最後由各決策樹輸出的類別的眾數(若為離散型資料)或數值平均數(若為連續型資料)來決定最終輸出的結果。其基礎概念如圖 6 所示。

隨機森林最主要的運作原理為 Bagging，以 Bootstrap 的方式分別對樣本及特徵進行取後放回的隨機抽樣，來建立資料子集，並用這些不同的資料子集隨機建立森林裡一棵棵的決策樹。在兩個隨機因子之下，使得隨機森林較不容易產生過度配適的現象，並具有易解釋性、可處理類別特徵、易擴充套件到多分類問題、不需特徵縮放等性質。

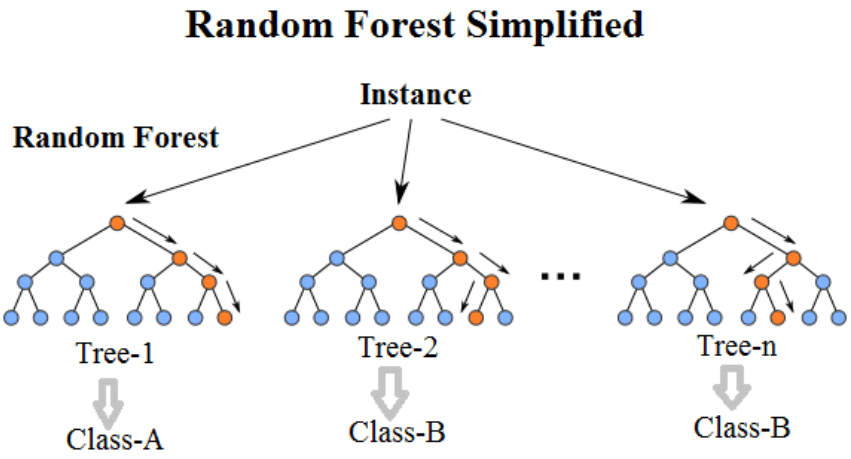


圖 6、隨機森林基礎概念圖示

(五) 羅吉斯迴歸 (Logistic Regression)

羅吉斯迴歸與線性迴歸類似，主要在探討相依變數與自變數之間的關係。線性迴歸中的相依變數(y)通常為連續型變數，但羅吉斯迴歸所探討的相依變數(y)主要為類別變數，且自變數 x 可以是類別變數或是連續變數。其簡單的圖示如圖 7 所示。

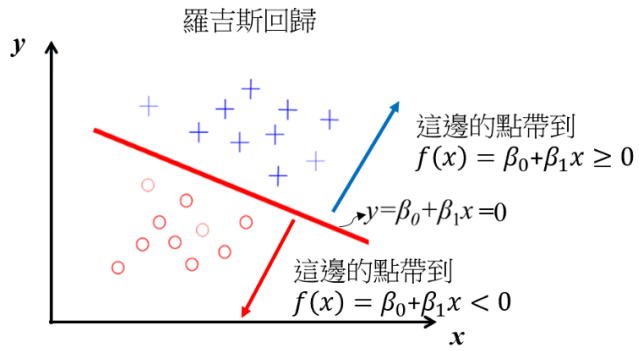


圖 7、簡易羅吉斯迴歸圖示

三、深度學習演算法

(一) 深度神經網路 (Deep Neural Network)

DNN 是一個包含多隱藏層的神經網路，可分為輸入層、隱藏層和輸出層。透過激活函數(Activation Function)做為下一層的輸入，再利用損失函數(Loss Function)計算其損失並更新權重值，最終得出其輸出值。其網路架構如圖 8 所示。

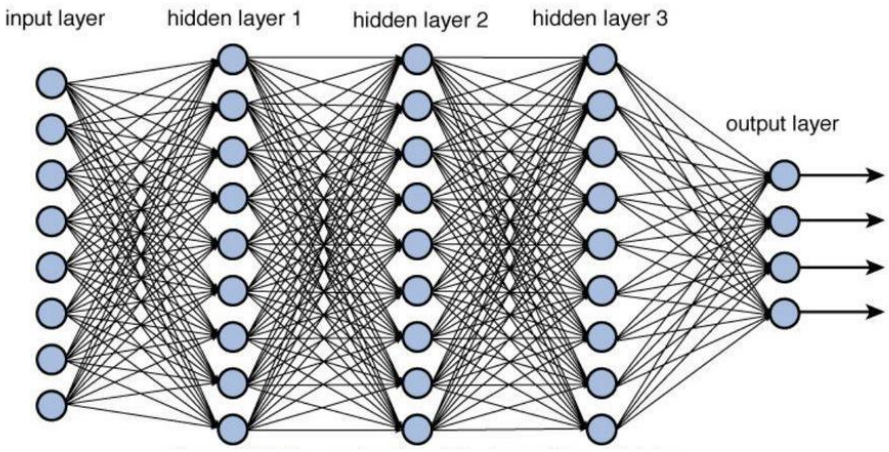


圖 8、深度神經網路架構示意圖

(二) 遞歸神經網絡 (RNN)

RNN 是專門設計用於處理時間序列數據的模型， 適用於文本，音節，視頻，天氣預報數據，股票交易以及其他時間序列數據。與傳統的神經網絡不同，它的核心概念是強調數據與數據之間的關係，就像網路具有暫存區一樣，它將根據過去的記憶確定當前的輸出。簡單來說，神經網絡只記住最近的事件，而較早的事件已被遺忘。

(三) 長短期記憶 (LSTM)

LSTM 是 RNN (Recurrent Neural Network) 的特殊型，目的在於解決長序列訓練過程中，梯度消失與梯度爆炸的問題。LSTM 主要分為三個階段，忘記階段、選擇記憶階段以及輸出階段。忘記階段主要是對上一個節點傳進來的輸入值執行選擇性遺忘，具體而言是透過計算得到之遺忘閥 (Forget Gate)，來控制上一個狀態中哪些需要保留以及遺忘。選擇記憶階段將這個階段的輸入值進行選擇性的「記憶」，重要之狀態予以著重記憶，反之不重要者則少記一些。輸出階段則決定哪些因子作為當前狀態之輸出。

(四) 門控循環單元 (GRU)

GRU 運用更新閥 (Update Gate) 取代 LSTM 中的遺忘閥與輸入閥，並把單元狀態和隱藏狀態進行合併，目的在於為了更好地捕捉時間序列中時間之間距離的依賴關係，透過可學習的來控制訊息的流通。GRU 在和弦音樂建模、語音信號建模和自然語言處理的某些任務上的表現與 LSTM 相似，且擁有較佳的準確率。GRU 單元結構示意圖如圖 9 所示。

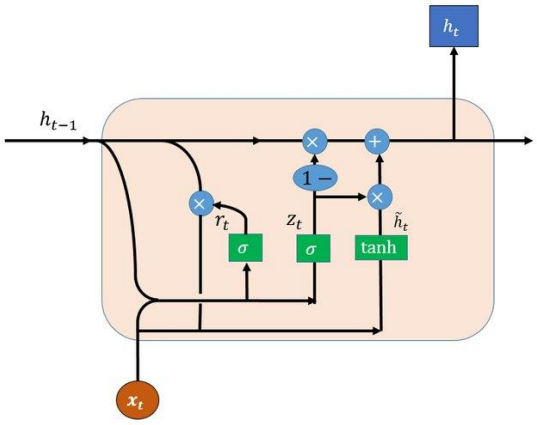


圖 9、GRU 的單元結構 (資料來源：https://reurl.cc/4aEnRR)

四、超參數優化 (Hyperparameter tuning)

針對深度學習模型，本研究採用手動方式調整，調整的超參數優化包含啟動函數 (Activation Function)，隱形層(Hidden Layer)，神經元數 (Number of neuron)，捨棄率 (Dropout rate)，學習率 (Learning rate) 等，詳細內容如表 3。

表 3、深度學習模型參數調整彙整表

Parameter	Value
Optimizer	Adam
Loss Function	Sparse Categorical Cross Entropy
Activation Function	Linear, ReLu, Sigmoid
Batch Size	{100,200,500}
Epochs	{10,20,50}
Number of Hidden Layers	{1,3,5}
Number of Neurons	{178,128,64,40}
Learning Rate	{0.01, 0.001, 0.0001, 0.000001}

肆、實驗流程與結果

一、資料前處理

(一) 資料清洗

1. 去除無用特徵

原始資料集中並非所有特徵適用於本次的研究，因此本研究去除了部分特徵，包含擁有大量缺失值、線性相依、特徵重複、無關聯及過於細節之特徵。其中，如線性相依之特徵是透過皮爾森積差相關分析得出，此階段將原始66個特徵下降至32個特徵。

2. 去除空值及文字於數值資料

原始資料集擁有3,000,040筆資料，考量資料量龐大，本研究直接去除含有空值特徵之資料，去除後筆數剩餘1,336,758筆。然而，有些許數值資料中包含文字單位，如 width 欄位內容可能為 176 in，由於後續分析希望為純數值資料，因此去除文字資訊，僅保留數值資訊。

3. 以州取代城市

原始資料集擁有城市特徵，由於美國有眾多小鎮，考量過多的變因恐降低模型訓練準確率，且因各州買賣稅率比不盡相同，故將城市與小鎮對比所在州區，以州作為主要特徵考量。此外，部分城市所在州區不明確，本研究將此資料直接剃除。

4. 獨熱編碼 (One-Hot Encoding)

此階段將所有類別型態之資料（如：顏色）以0或1呈現，並擴充特徵維度，轉換之特徵包含body_type、state、engine_cylinders、fuel_type、listing_color、wheel_system、make_name 以及transmission。

5. 移除極端值

考量極端值對模型訓練的影響，本研究剔除價錢超過此資料集三倍正負標準差之極端值。最終清洗完之資料集包含178個特徵，擁有1,089,802筆資料列。

(二) 價格區間貼標分類

首先透過python numpy函式庫中的np.percentile，計算價格上、下限，保留距離平均值三個標準差以內的資料 (99.73%)，刪除6,180筆極端值後，將剩餘1,089,802筆資料按比例切分為20個區間，每段價格區間約包含54,490筆資料，價格區間如表 4所示。

表 4、中古車價價格區間貼標分類說明

區間	價格下限	價格上限	區間	價格下限	價格上限
1	2,503	6,495	11	18,996	20,000
2	6,496	8,599	12	20,001	21,922
3	8,600	10,495	13	21,923	23,595
4	10,496	11,999	14	23,596	25,499
5	12,000	13,794	15	25,500	27,652
6	13,795	14,988	16	27,653	29,987
7	14,989	15,995	17	29,988	32,978
8	15,996	16,995	18	32,979	36,500
9	16,996	17,995	19	36,501	41,995
10	17,996	18,995	20	41,996	76,249

(三) 主成分分析 (Principal Component Analysis)

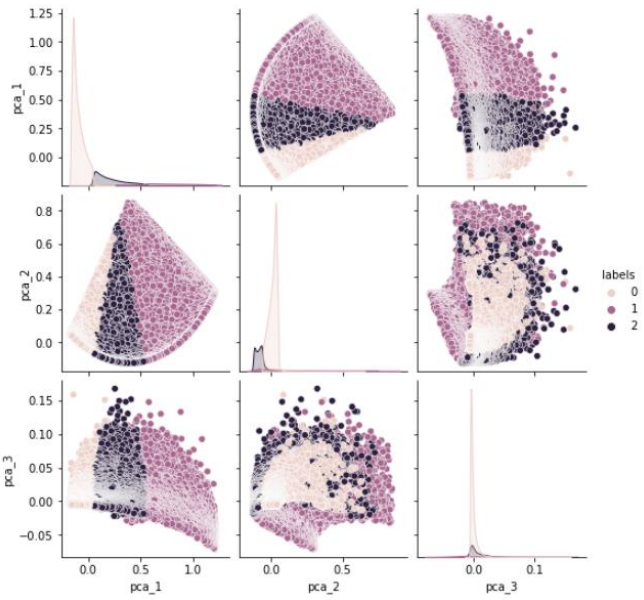
利用 sklearn.decomposition 中的 pca 對資料進行降維。

```
from sklearn.decomposition import PCA

pca = PCA(n_components=3)  # 把维度降至3维
# 进行PCA降維
X_pca = pca.fit_transform(X=ndf.drop("price",axis=1))
# 生成降维后的dataframe
X_pca_frame = pd.DataFrame(X_pca, columns=['pca_1', 'pca_2', 'pca_3'])
X_pca_frame.head()
```

	pca_1	pca_2	pca_3
0	0.490739	0.809031	-0.021137
1	1.171815	0.338632	-0.030002
2	1.126010	0.370599	-0.024534
3	1.083240	0.403391	0.012482
4	1.092461	0.397196	0.014629

接著透過 seaborn 中的 sns 畫出 pca 之間之關聯圖。



(四) 資料切割

將資料拆分成 80%的訓練集及 20%的測試集，並區分出特徵欄位與目標欄位。

```
#將資料分成訓練集及測試集
from sklearn.model_selection import train_test_split
train_data, test_data=train_test_split(data, random_state=777,train_size=0.8)

train_label=train_data[:,-1]
test_label=test_data[:,-1]

train_data=train_data[:,:-1]
test_data=test_data[:,:-1]
```

(五) 資料標準化

利用 sklearn.preprocessing 中的 scale 函數，將數據標準化。

```
from sklearn import preprocessing
train_data = preprocessing.scale(train_data)
test_data = preprocessing.scale(test_data)
```

二、機器學習模型建立

(一) 線性迴歸 (Linear Regression)

使用 sklearn 中 linear_model 套件的 LogisticRegression()函數進行線性迴歸。根據訓練後的模型，使用測試集進行預測，並印出預測分數。

```
In [47]: lr1 = LinearRegression()
         lr1.fit(x_train1,y_train1)
         lr1.score(x_test1,y_test1)
         print(lr1.intercept_)

         <bound method RegressorMixin.score of LinearRegression()>
         -1955161.7773043916

In [46]: lr1.predict(x_n)

Out[46]: array([18859.53939302, 42107.05528142])

In [48]: print(lr1.score(x_test1,y_test1))

         0.8410320133417097
```

(二) 多層感知器迴歸 (Multi-layer Perceptron Regressor)

使用 sklearn 中 neural_network 套件的 MLPRegressor()函數進行多層感知器迴歸。根據訓練後的模型，使用測試集進行預測，並印出預測分數。

```
In [49]: regr = MLPRegressor(random_state=1, max_iter=500).fit(X_train, y_train)
         regr.score(X_test, y_test)

Out[49]: 0.8911469955317866
```

(三) 支援向量機 (Support Vector Machine, SVM)

由於資料量龐大，使用 SVC()函數的訓練時間過長，因此使用隨機梯度下降分類(SGDClassifier)來訓練模型，其預設的 loss function 支持線性 SVM，因此不再多設參數。

```
from sklearn import preprocessing
train_data = preprocessing.scale(train_data)
test_data = preprocessing.scale(test_data)

from sklearn.linear_model import SGDClassifier
clf=SGDClassifier(random_state=50,n_jobs=-1)
clf.fit(train_data,train_label)
```

```
SGDClassifier(n_jobs=-1, random_state=50)
```

根據訓練後的模型，使用測試集進行預測，並印出對於不同類別的相關預測分數。

```
clf_pred=clf.predict(test_data)
print("train_score= ",clf.score(train_data,train_label))
print("test_score= ",clf.score(test_data, test_label))

from sklearn.metrics import classification_report
print(classification_report(test_label,clf_pred))
```

```
train_score= 0.18854928823030806
test_score= 0.18812998655722812
              precision    recall  f1-score   support

         0.0       0.78      0.77      0.78     10945
         1.0       0.27      0.22      0.25     10712
         2.0       0.14      0.13      0.14     10899
         3.0       0.13      0.13      0.13     10620
         4.0       0.13      0.17      0.15     11016
         5.0       0.08      0.06      0.07     10819
         6.0       0.10      0.13      0.11     11681
         7.0       0.11      0.11      0.11     10751
         8.0       0.08      0.07      0.07     10906
         9.0       0.08      0.13      0.10     10216
        10.0       0.07      0.08      0.07     11146
        11.0       0.09      0.08      0.09     10902
        12.0       0.09      0.08      0.08     10836
        13.0       0.10      0.13      0.11     10903
        14.0       0.12      0.09      0.11     10897
        15.0       0.09      0.09      0.09     11016
        16.0       0.14      0.19      0.16     10971
        17.0       0.18      0.10      0.13     10904
        18.0       0.29      0.28      0.28     10905
        19.0       0.74      0.73      0.74     10916

    accuracy                           0.19    217961
   macro avg       0.19      0.19      0.19    217961
weighted avg       0.19      0.19      0.19    217961
```

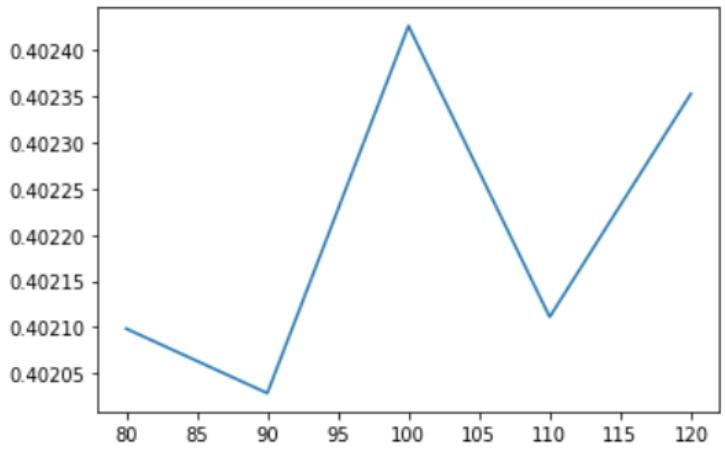
(四) 隨機森林 (Random Forest)

藉由交叉驗證，選擇最優的決策樹數量(n_estimators)，由於決策樹數量過多會導致運算量過大，此處僅驗證到120棵決策樹。

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
rfc = RandomForestClassifier()

superpa = []
print("i=")
for i in range(80,121,10):
    print(i)
    rfc = RandomForestClassifier(n_estimators=i,n_jobs=-1,min_samples_leaf = 10)
    rfc_s = cross_val_score(rfc,train_data1,train_label1,cv=10).mean()
    superpa.append(rfc_s)
```

```
plt.plot(range(80,121,10),superpa)
plt.show()
```



根據交叉驗證的結果，在100棵決策樹時有最好的分數，因此在此隨機森林模型中的決策樹數量之參數選擇100。

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=100,
                           random_state =50,n_jobs = -1,
                           min_samples_leaf = 10)
rfc.fit(train_data1,train_label1)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=10, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=50, verbose=0,
                       warm_start=False)
```

根據訓練後的隨機森林模型，使用測試集進行預測，並印出對於不同類別的相關預測分數。

```
rfc_pred=rfc.predict(test_data1)
print("train_score= ",rfc.score(train_data1,train_label1))
print("test_score= ",rfc.score(test_data1, test_label1))

from sklearn.metrics import classification_report
print(classification_report(test_label1,rfc_pred))
```

```
train_score=  0.4760959853918318
test_score=  0.4034070315331642
              precision    recall  f1-score   support

         0.0       0.76      0.81      0.78     10945
         1.0       0.53      0.53      0.53     10712
         2.0       0.46      0.48      0.47     10899
         3.0       0.39      0.42      0.40     10620
         4.0       0.37      0.38      0.37     11016
         5.0       0.31      0.28      0.29     10819
         6.0       0.28      0.34      0.31     11681
         7.0       0.27      0.23      0.25     10751
         8.0       0.27      0.26      0.26     10906
         9.0       0.25      0.17      0.20     10216
        10.0       0.27      0.29      0.28     11146
        11.0       0.32      0.29      0.30     10902
        12.0       0.30      0.31      0.30     10836
        13.0       0.31      0.34      0.32     10903
        14.0       0.34      0.32      0.33     10897
        15.0       0.37      0.39      0.38     11016
        16.0       0.40      0.40      0.40     10971
        17.0       0.45      0.43      0.44     10904
        18.0       0.55      0.59      0.57     10905
        19.0       0.79      0.82      0.81     10916

    accuracy                           0.40    217961
   macro avg       0.40      0.40      0.40    217961
weighted avg       0.40      0.40      0.40    217961
```

(五) 羅吉斯迴歸 (Logistic Regression)

使用 sklearn 中 linear_model 套件的 LogisticRegression()函數進行羅吉斯迴歸。

```
from sklearn import linear_model
model=linear_model.LogisticRegression()
model.fit(train_data3,train_label3)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

根據訓練後的羅吉斯迴歸模型，使用測試集進行預測，並印出對於不同類別的相關預測分數。

```
print("train_score= ",model.score(train_data3,train_label3))
print("test_score= ",model.score(test_data3,test_label3))

from sklearn.metrics import classification_report
print(classification_report(test_label3,model.predict(test_data3)))
```

```
train_score=  0.2685696130372396
test_score=  0.2657723170658971
              precision    recall  f1-score   support

         0.0       0.66      0.89      0.76     10945
         1.0       0.33      0.32      0.32     10712
         2.0       0.25      0.30      0.27     10899
         3.0       0.19      0.18      0.19     10620
         4.0       0.20      0.14      0.17     11016
         5.0       0.23      0.15      0.18     10819
         6.0       0.17      0.19      0.18     11681
         7.0       0.18      0.16      0.17     10751
         8.0       0.15      0.17      0.16     10906
         9.0       0.17      0.06      0.09     10216
        10.0       0.16      0.16      0.16     11146
        11.0       0.19      0.22      0.21     10902
        12.0       0.16      0.12      0.14     10836
        13.0       0.16      0.16      0.16     10903
        14.0       0.17      0.16      0.16     10897
        15.0       0.18      0.24      0.21     11016
        16.0       0.20      0.23      0.21     10971
        17.0       0.27      0.21      0.24     10904
        18.0       0.34      0.39      0.36     10905
        19.0       0.65      0.84      0.73     10916

    accuracy                           0.27    217961
   macro avg       0.25      0.27      0.25    217961
weighted avg       0.25      0.27      0.25    217961
```

進一步修改超參數，在分類方式選擇參數(multi_class)選擇multinomial，做多個類別對多個類別的分類；優化算法選擇參數(slover)選擇較適合大樣本的newton-cg，更新羅吉斯迴歸模型：

```
from sklearn import linear_model
model=linear_model.LogisticRegression(multi_class='multinomial', solver='newton-cg')
model.fit(train_data3,train_label3)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='newton-cg', tol=0.0001, verbose=0,
                   warm_start=False)
```

根據修改後的羅吉斯迴歸模型，使用測試集進行預測，並印出對於不同類別的相關預測分數。

```
print("train_score= ",model.score(train_data3,train_label3))
print("test_score= ",model.score(test_data3,test_label3))

from sklearn.metrics import classification_report
print(classification_report(test_label3,model.predict(test_data3)))
```

```
train_score=  0.35305864257358854
test_score=  0.3509710452787425
              precision    recall  f1-score   support

         0.0       0.78      0.78      0.78     10945
         1.0       0.54      0.54      0.54     10712
         2.0       0.45      0.49      0.47     10899
         3.0       0.37      0.39      0.38     10620
         4.0       0.33      0.33      0.33     11016
         5.0       0.28      0.23      0.25     10819
         6.0       0.23      0.33      0.27     11681
         7.0       0.23      0.18      0.20     10751
         8.0       0.21      0.22      0.22     10906
         9.0       0.22      0.10      0.13     10216
        10.0       0.20      0.25      0.23     11146
        11.0       0.24      0.21      0.22     10902
        12.0       0.22      0.22      0.22     10836
        13.0       0.24      0.25      0.24     10903
        14.0       0.25      0.25      0.25     10897
        15.0       0.30      0.31      0.30     11016
        16.0       0.31      0.32      0.32     10971
        17.0       0.36      0.32      0.34     10904
        18.0       0.45      0.51      0.47     10905
        19.0       0.75      0.77      0.76     10916

    accuracy                           0.35    217961
   macro avg       0.35      0.35      0.35    217961
weighted avg       0.35      0.35      0.35    217961
```

前面提到的隨機梯度下降分類(SGD Classifier)也可以支援羅吉斯迴歸的預測分類，將其loss function設為log即可進行羅吉斯迴歸。

```
from sklearn.linear_model import SGDClassifier
SGD_Lo=SGDClassifier(loss='log',random_state=50,n_jobs=-1,penalty='elasticnet')
SGD_Lo.fit(train_data3,train_label3)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=1000,
              n_iter_no_change=5, n_jobs=-1, penalty='elasticnet', power_t=0.5,
              random_state=50, shuffle=True, tol=0.001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

根據訓練後的模型，使用測試集進行預測，並印出對於不同類別的相關預測分數。

```
print("train_score= ",SGD_Lo.score(train_data3,train_label3))
print("test_score= ",SGD_Lo.score(test_data3,test_label3))

from sklearn.metrics import classification_report
print(classification_report(test_label3,SGD_Lo.predict(test_data3)))
```

```
train_score=  0.25857008330647446
test_score=  0.25598616266212765
              precision    recall  f1-score   support

         0.0       0.79      0.76      0.78     10945
         1.0       0.34      0.51      0.40     10712
         2.0       0.26      0.20      0.23     10899
         3.0       0.19      0.22      0.21     10620
         4.0       0.17      0.09      0.12     11016
         5.0       0.19      0.16      0.18     10819
         6.0       0.16      0.17      0.16     11681
         7.0       0.14      0.09      0.11     10751
         8.0       0.16      0.09      0.11     10906
         9.0       0.15      0.16      0.15     10216
        10.0       0.14      0.26      0.18     11146
        11.0       0.18      0.16      0.17     10902
        12.0       0.15      0.11      0.13     10836
        13.0       0.17      0.18      0.17     10903
        14.0       0.15      0.09      0.11     10897
        15.0       0.18      0.32      0.23     11016
        16.0       0.20      0.13      0.16     10971
        17.0       0.23      0.23      0.23     10904
        18.0       0.34      0.47      0.39     10905
        19.0       0.75      0.72      0.73     10916

    accuracy                           0.26    217961
   macro avg       0.25      0.26      0.25    217961
weighted avg       0.25      0.26      0.25    217961
```

可以發現與LogisticRegression()函數的結果相去不遠。

三、深度學習模型建立

(一) 深度神經網路 (DNN)

因模型最終結果以分類形式呈現，故在最後一層皆以 Softmax 作為激活函數，然而在隱藏層中的激活函數調整包含ReLu、Linear 以及 Sigmoid。其中，以 Linear 的表現力最佳。在 Batch Size 的調整中，以 Batch Size 為 500 時的模型訓練結果最好。DNN 模型之激活函數評估比較表如

表 5。

表 5、DNN 模型之超參數設定表

超參數	參數設定
激活函數	Linear
Batch Size	500
第一層神經元數	178
第二層神經元數	178
第三層神經元數	64
第四層神經元數	64
第五層神經元數	40
Epochs	50
學習率	0.000001
模型準確率	0.2421

(二) 遞歸神經網絡 (RNN)

在建立 RNN 模型的過程中，激活函數調整以 ReLu 的表現力最佳。在 Batch Size 的調整中，以 Batch Size 為 100 時的模型訓練結果最好。RNN 模型之激活函數評估比較表如表 6。

表 6、RNN 模型之超參數設定表

超參數	參數設定
激活函數	Linear
Batch Size	500
第一層神經元數	178
第二層神經元數	178
第三層神經元數	64
Epochs	10
學習率	0.001
模型準確率	0.0505

(三) 長短期記憶 (LSTM)

在建立 LSTM 模型的過程中，激活函數調整以 ReLu 的表現力最佳。在 Batch Size 的調整中，以 Batch Size 為 500 時的模型訓練結果最好。LSTM 模型之激活函數評估比較表如表 7。

表 7、LSTM 模型之超參數設定表

超參數	參數設定
激活函數	ReLu
Batch Size	100
第一層神經元數	178
第二層神經元數	64
第三層神經元數	40
Epochs	10
學習率	0.001
模型準確率	0.0711

(四) 門控循環單元 (GRU)

在建立 GRU 模型的過程中，激活函數調整以 ReLu 的表現力最佳。在 Batch Size 的調整中，以 Batch Size 為 200 時的模型訓練結果最好。GRU 模型之激活函數評估比較表如表 8。

表 8、GRU 模型之超參數設定表

超參數	參數設定
激活函數	ReLu
Batch Size	200
第一層神經元數	178
第二層神經元數	178
第三層神經元數	64
Epochs	10
學習率	0.0001
模型準確率	0.0546

四、實驗結果與分析

(一) 機器學習結果統整

如表 9所示，該資料集在機器學習演算法中以多層感知器迴歸之分數表現最佳，分數為0.89；次者表現佳之演算法為線性迴歸，分數為0.84；其餘演算法之分數表現力皆未超過0.5，其原因可能為資料集特性，本研究雖將欲預測值進行區間貼標，然而特徵與特徵之間存在線性關係，較不適用於分類之演算法。

表 9、機器學習訓練結果分數統整表

	線性迴歸	多層感知器迴歸	支持向量機	隨機森林	羅吉斯迴歸
分數	0.84	0.89	0.19	0.40	0.35

如表 10所示，該資料集在機器學習演算法中的泛化結果以資料二的泛化能力最佳，其原因可能在於前段價錢的區間較為緊密，而高單價之中古車的價錢則較為離散，故在猜測前段價錢較容易產生偏誤。

表 10、機器學習泛化結果統整表

DATA	實際結果	線性迴歸	多層感知器迴歸	支持向量機	隨機森林	羅吉斯迴歸
一	7	9	9	8	1	3
二	19	19	17	14	19	17

(二) 深度學習結果統整

如表 11所示，該資料集在深度學習演算法中普遍結果皆不理想，其原因可能為資料集特性不適用於深度學習演算法。此外，相較於其他演算法，DNN模型在分數的表現上較佳，RNN、LSTM以及GRU都屬於分析擁有時間序列之資料集，而本研究所分析之中古車資料集並未擁有此特性，因此訓練出之模型準確度不甚理想。

表 11、深度學習訓練結果分數統整表

	DNN	RNN	LSTM	GRU
分數	0.2421	0.0505	0.0711	0.0546

如前述以及表 12所示，該資料集在深度學習演算法中的泛化結果皆產生較大的偏誤，對應模型訓練準確度不佳之結果。

表 12、深度學習泛化結果統整表

DATA	實際結果	DNN	RNN	LSTM	GRU
一	7	8	1	1	3
二	19	18	1	1	3

伍、結論

一、整體貢獻

我們透過分析中古車相關特徵與價錢之間的關聯性，取得對於購買中古車相對重要的指標，以本研究結果發現，中古車之馬力、扭力、年份以及里程為最重要之四個指標。透過這些指標，我們期望對於消費者而言，可以提供更有利的資訊，以作為購買中古車的考量關鍵。再者，本研究也驗證在不同演算法下，針對同一資料集的表現力，除了與資料特性相關，演算法的適用程度也會有所不同。

二、研究限制

礙於時間與硬體設備限制，此次的研究無法針對太多超參數優化進行模型訓練及分析。在資料前處理的過程中，本研究去除了大部分有缺失值的資料，這在方面並未作其他填補缺失值的研究。此外，本研究所採用之資料集來源為美國地區，不適用於台灣地區的中古車市場行情。

三、應用方面

任何中古車買賣平台或實體商家皆可參考本研究結果，在價格訂定上可以有所調整。對於資料分析有興趣之研究者，本研究之資料集擁有足夠的資訊量，供研究者做更進一步之分析。

四、未來展望

綜上所述，未來可考慮延伸資料前處理的方法，針對資料集做更完善的處理，有助於提升訓練結果的表現力。另外，也能針對台灣市場的中古車行情進行研究，更符合研究者所處之地域性。

參考資料

[Day 11]核模型-支持向量機(SVM)

https://ithelp.ithome.com.tw/articles/10270447?sc=pt

ML 入門（十七）隨機森林(Random Forest) https://reurl.cc/352xkl

機器/統計學習: 羅吉斯迴歸(Logistic regression) https://reurl.cc/52A1vn