應用機器學習與深度學習於中古車價格預測之比較

Group7
110034538 許淨嵐
110034540 徐阡珊
110034553 李慕瑄

Contents

01 簡介

02 實驗流程

03 結論

01

簡介

- 背景與動機
- 研究目的
- 5W1H 分析
- 資料集介紹

背景與動機

- 中古車過戶數遠超過於新車領牌數，約為新車市場的1.5至1.6倍
- 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
- 新冠肺炎疫情刺激通勤民眾之購車代步意願

▼全台歷年新車和中古車銷售量

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

研究目的

- 期望透過中古車買賣歷史資料，比較不同演算法，發展預測模型，在給定特定車況的數據下，得出最符合該車輛二手價錢，進而提升民眾議價能力

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

資料集介紹

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

02

實驗流程

- 資料前處理
- 機器學習模型建立
- 深度學習模型建立
- 實驗結果與分析

資料前處理 (1/4)

1. Drop Useless Features

66 Features ⟶ 32 Features

Large missing values

-bed
-bed_height
-bed_length
-cabin
-fleet
-is_certified
-vehicle_damage_category

Collinearity

-combine_fuel_economy

Duplicate

-engine_type(same as enginec_cylinders)
-exterior_color(same as listing color)
-power(same as horsepower)
-transmission_display
-trimId
-wheel_system_display

Not related

-daysonmarket
-dealer_zip
-description
-franchise_dealer
-franchise_make
-is_cpo
-is_oemcpo
-latitude
-listed_date
-listing_id
-longitude
-main_picture_url
-savings_amount
-seller_rating
-sp_id
-sp_name
-theft_title

Too detailed

-interior_color
-trim_name
-major_options

資料前處理 (2/4)

2. Drop NAs

Dimension from 3,000,040 to 1,336,758

- Drop “- -” in remain features

3. Drop Text from Numerical data

- back_legroom: drop “in”
- wheelbase: drop “in”
- width: drop “in”
- body_type: drop “/Crossover” from “SUV / Crossover”
- maximum_seating: drop “seats”

資料前處理 (3/4)

4. Replace “city” with “state”

Consider the tax at each state

- Drop unknown city

5. One Hot Encoding

- body_type
- state
- engine_cylinders
- fuel_type
- listing_color
- wheel_system
- make_name
- transmission

資料前處理 (4/4)

6. Remove extreme values

Clean max, min data (±3 std)

- From 1,095,982 to 1,089,802 Collections
- Divided into 20 price levels

機器學習模型建立

- 線性迴歸 (Linear Regression)
- 多層感知器迴歸 (Multi-layer Perceptron Regressor)
- 支援向量機 (Support Vector Machine, SVM)
- 隨機森林 (Random Forest)
- 羅吉斯迴歸 (Logistic Regression)

線性迴歸

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
```

多層感知器迴歸

```
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
```

- 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，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。

支援向量機

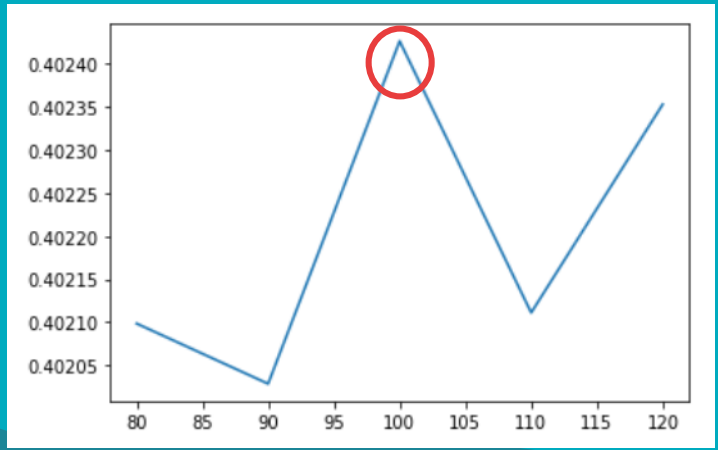
```
from sklearn import preprocessing
train_data = preprocessing.scale(train_data)
test_data = preprocessing.scale(test_data)

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)
```

```
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
```

隨機森林 (1/2)

```
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)
```



隨機森林 (2/2)

```
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)
```

```
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
```

羅吉斯迴歸

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

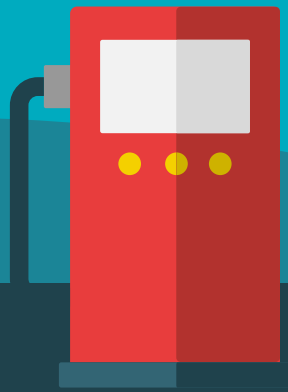
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='newton-cg', tol=0.0001, verbose=0,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

```
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
```

深度學習模型建立

- 深度神經網路 (DNN)
- 遞歸神經網絡 (RNN)
- 長短期記憶 (LSTM)
- 門控循環單元 (GRU)



DNN & RNN

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

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

LSTM & GRU

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

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

超參數優化

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}

實驗結果與分析

機器學習	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	實際結果
泛化 Data 1	9	9	8	1	3	7
泛化 Data 2	19	17	14	19	17	19

深度學習	DNN	RNN	LSTM	GRU	
分數	0.24	0.05	0.07	0.05	實際結果
泛化 Data 1	8	1	1	3	7
泛化 Data 2	18	1	1	3	19

03

結論

- 整體貢獻
- 研究限制
- 應用方面
- 未來展望

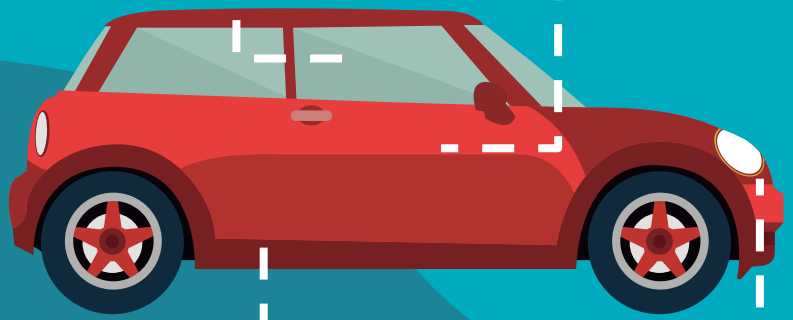
結論

整體貢獻

- 關鍵指標
- 演算法適用性

應用方面

- 資料內容豐富



研究限制

- 缺失值資料
- 文化差異

未來展望

- 分類→迴歸
- 超參數優化
- 自行建構演算法


THANK YOU
Group 7, 2021 IIE project 2