智慧化企業整合

Individual Research Project
預測共享單車需求

109034543 黃荻雅
指導教授:邱銘傳 教授

摘要
我們都知道要精準用戶需求天經地義
共享單車本就是很依賴電子系統的產業，可想而知它的資料相對來說很完整，而且正確度也高

一、 介紹

1. 背景

與在台灣的欣欣向榮不同，共享單車在國外一向是毀譽參半。隨即取用的便利性造福許多人，然而這也是這一產業的最大缺點。停放帶來了管理成本和對城市公共資源的佔用，而共享單車多數仍保留著停車樁或固定停放站點的租用模式，重點關注車輛的維護和管理，整體市場不溫不火。更不用說有些極端案例，共享單車變成共享垃圾。

2. 目的

對於共享單車而言，決定區域中單車投放與回收的數量、時間和頻率事關重大，可以說是與單車的維護管理同等重要。本研究的共享單車資料取自華盛頓特區，藉由溫度、濕度、日期……等預測共享單車使用數據，希望可以協助管理者分配人力與車輛。

3. 5W1H 分析

What:預測華盛頓特區的共享單車需求。
Why:希望藉由數據分析，幫助管理者預測需求，達到更高的獲利與客戶滿意度。
Who:由本專案成員執行，數據收集由共享單車 IT 部門負責，預計可使公司和客戶雙方受益
Where:共享單車站點、總公司。
When:顧客需求與預估量有差異時。
How:透過歷史資料建立模型，並進行預測。

二、 方法

1. Random Forest(隨機森林演算法)

Random Forest 像是進階版的決策數，基本原理是結合多顆使用 GINI 的決策樹，並加入隨機分配的訓練資料，以大幅增進最終的運算結果。也就是由許多不同的決策樹所組成的一個機器學習模型，其想法就是結合多個「弱模型」來建構一個更強的模型。這種方法也稱為 Ensemble Method，也就是「三個臭皮匠勝過一個諸葛亮」的概念。

2. KNN 演算法

KNN 演算法又稱 k 近鄰分類(k-nearest neighbor classification)演算法，是一種監督是演算法。它是根據不同特徵值之間的距離來進行分類的一種簡單的機器學習方法，它是一種簡單但是懶惰的演算法。KNN 演算法用於迴歸的核心思想是：找到近鄰的 k 個樣本，然後取平均值作為未知樣本的值，對其進行預測。
演算法步驟大致可以分成三步:1.算距離：給定未知物件，計算它與訓練集

中的每個物件的距離。2.找近鄰：圈定距離最近的 k 個訓練物件，作為未知物件的近鄰。2.做預測：這 k 個近鄰取平均值，作為未知樣本的值。

三、 案例研究

1. 資料前處理

Kaggle 提供的訓練資料共 10886 筆、12 個欄位，測試資料則是 6493 筆、9 個欄位，兩個資料集都沒有空值。資料的時間從 2011 到 2012，時間詳細到小時，有每個小時的天氣、使用量資料……等。可以說是很完整的資料

欄位名稱	說明
datetime	詳細到小時的日期
season	1:春天 2:夏天 3:秋天 4:冬天
holiday	0:非節日 1:節日
workingday	0:非工作日 1:工作日
weather	1:晴朗、微量的雲 2:有霧或雲 3:小雪、小雨、暴風 4:大雨、冰雹
temp	溫度(攝氏)
atemp	體感溫度(攝氏)
humidity	相對濕度
windspeed	風速
casual	沒註冊的使用者租借數量
registered	已註冊的使用者租借數量
count	總租借數量

首先檢查分類資料是否有異常值

```
#檢查分類資料是否合乎規範
print(dt["season"].unique())
print(dt["holiday"].unique())
print(dt["workingday"].unique())
```

```
[1 2 3 4]
[0 1]
[0 1]
```

處理時間資料

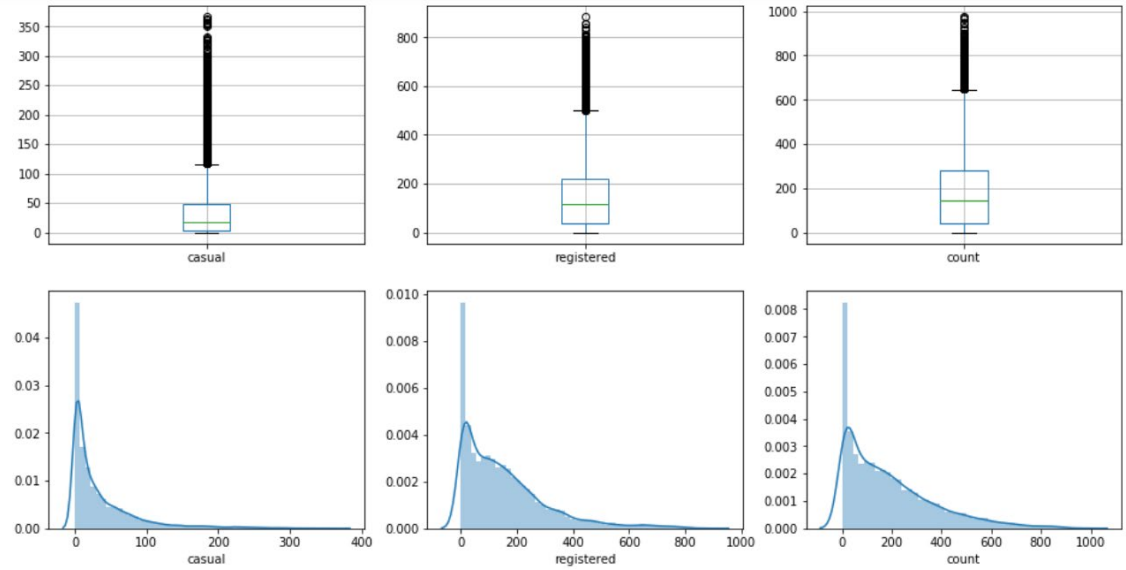
將資料以固定格式轉換，留下年分、月份、星期與小時

```
#轉換日期與時間資料
dt['datetime'] = pd.to_datetime(dt['datetime'],format='%Y-%m-%d %H:%M:%S') #訓練資料
dt_test['datetime'] = pd.to_datetime(dt_test['datetime'],format='%Y-%m-%d %H:%M:%S') #測試資料
```

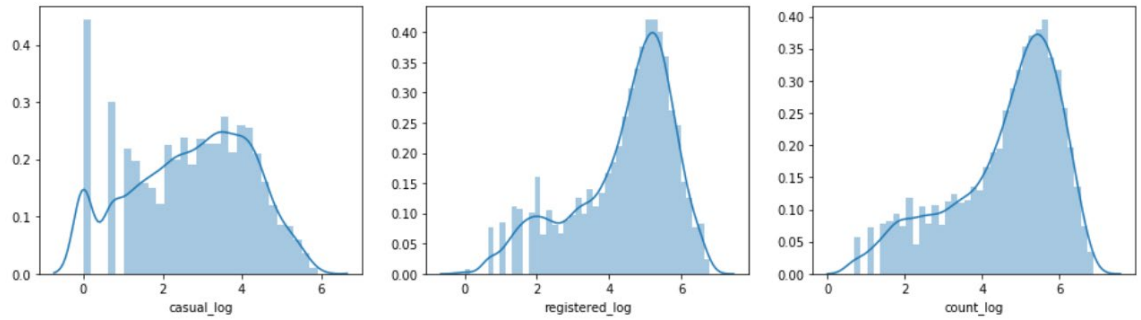
```
#留下年分、月份、星期與小時
dt["year"] = dt['datetime'].dt.year
dt["month"] = dt['datetime'].dt.month
dt["hour"] = dt['datetime'].dt.hour
dt["dayofweek"] = dt['datetime'].dt.dayofweek

dt_test["year"] = dt_test['datetime'].dt.year
dt_test["month"] = dt_test['datetime'].dt.month
dt_test["hour"] = dt_test['datetime'].dt.hour
dt_test["dayofweek"] = dt_test['datetime'].dt.dayofweek
```

歪斜的使用量資料



取 log 後的資料



可以看到除了 casual 外，其他資料正常很多，推測是因為未註冊的用戶較不穩定。

比較兩種資料的初步預測差異:

```
X_train, X_test, y_train, y_test = train_test_split(dt[factor], dt["count"], test_size = 0.3, random_s
rfmodel = RandomForestRegressor(n_estimators=3000, random_state=42, max_depth=30)
rfmodel.fit(X_train,y_train)
pred = rfmodel.predict(X_test)
MSE = mean_squared_error(pred, y_test)
MSLE = mean_squared_log_error(pred, y_test)
#測試集分數
print("RMSE =", MSE**0.5)
print("RMSLE =", MSLE**0.5)
```

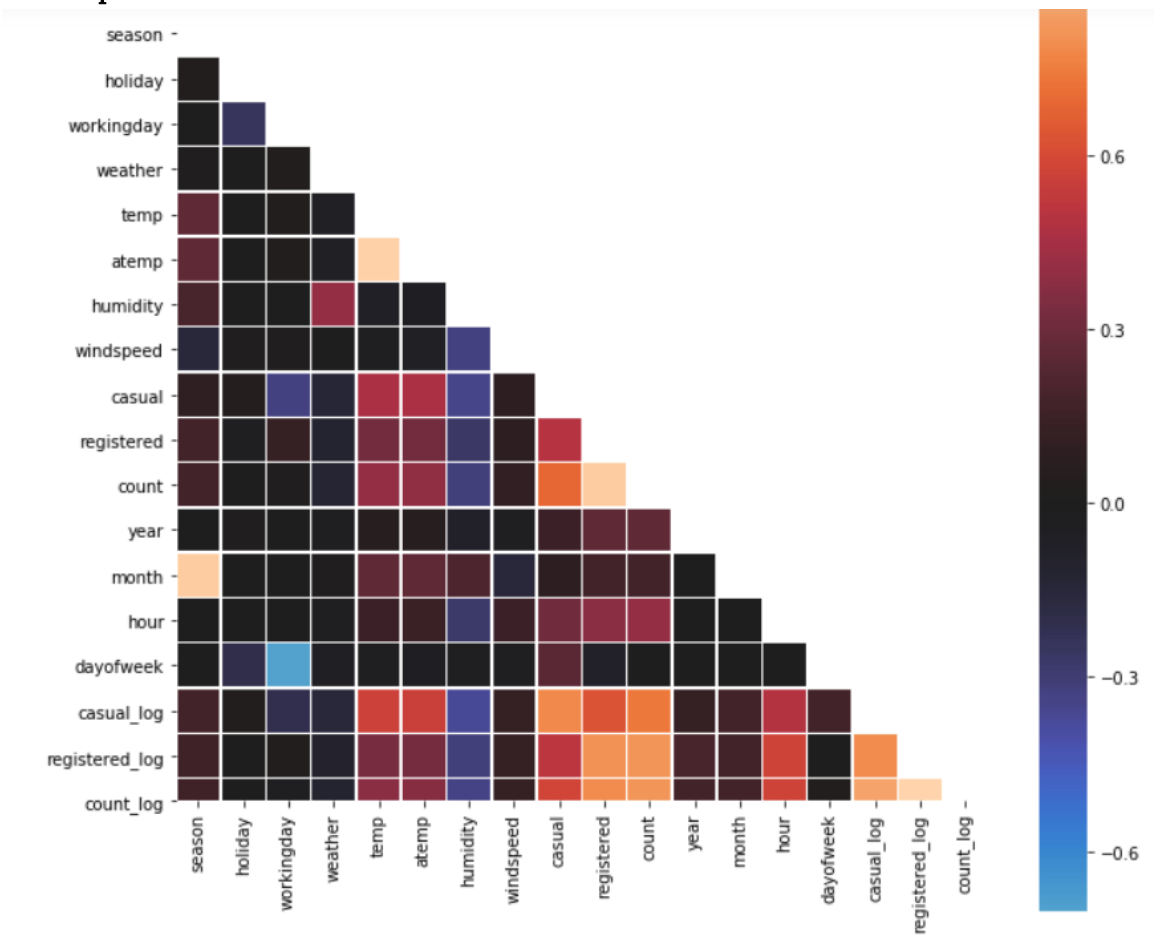
```
RMSE = 154.13067529171568
RMSLE = 1.2845150860833485
```

```
pred = rfmodel.predict(X_test)
MSE = mean_squared_error(pred, y_test)
MSLE = mean_squared_log_error(pred, y_test)
#測試集分數
print("RMSE =", MSE**0.5)
print("RMSLE =", MSLE**0.5)
```

```
RMSE = 1.2691033479515628
RMSLE = 0.30814407619288725
```

上圖為未處理資料的結果，RMSE 為 154.13、RMSLE 為 1.28。
下圖為資料取 log 的結果，RMSE 為 1.27、RMSLE 為 0.31。
使用的模型是隨機森林，可以觀察到預測結果有顯著的提升。

Heatmap

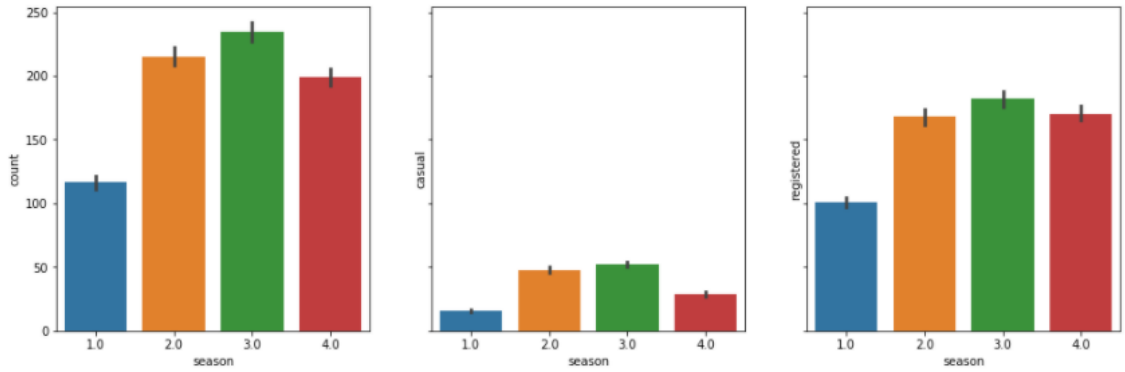


每個欄位對使用量皆沒有顯著影響，需要進一步做處理和排除。

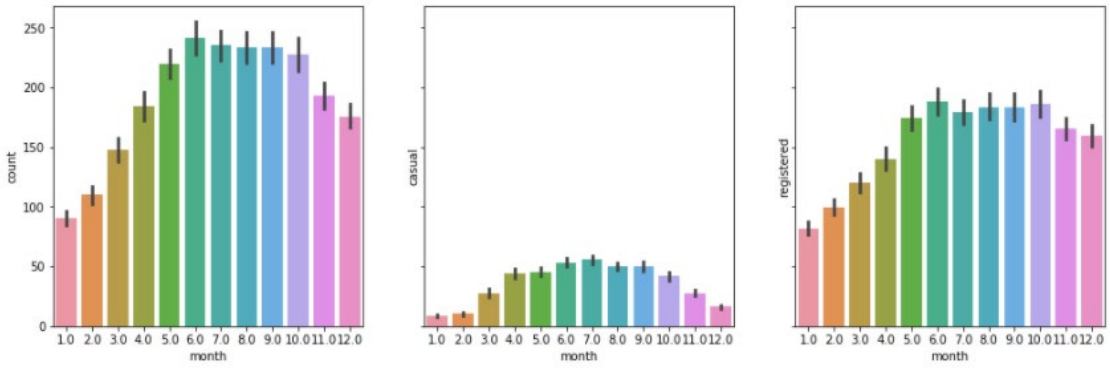
觀察各因子作用

依照使用者種類:count(total), casual, registered，畫出各欄位的分布圖。

季節

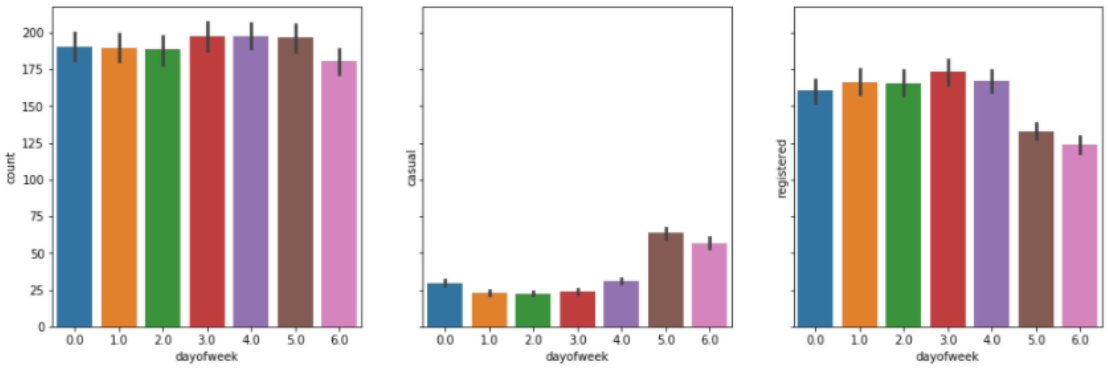


月份



三張圖趨勢相同，春天最低，高峰在夏、秋

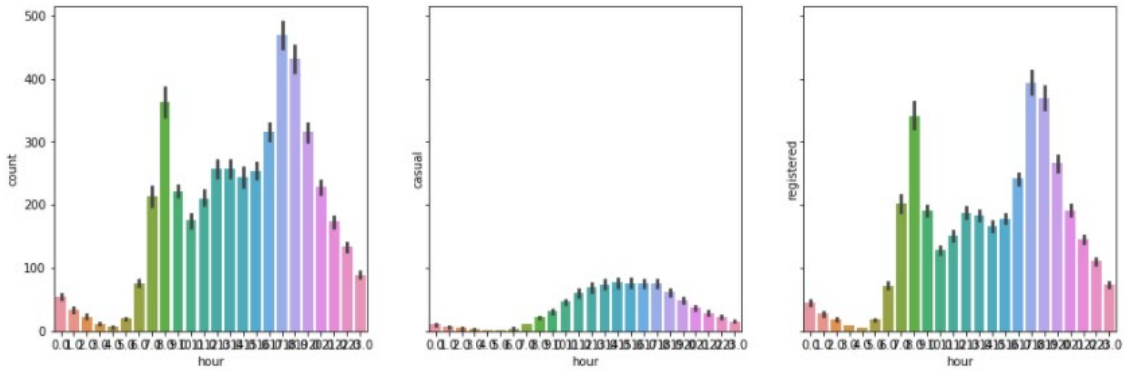
星期



總體來說，使用量不受星期影響。但未註冊的使用者在周末用量較大，已

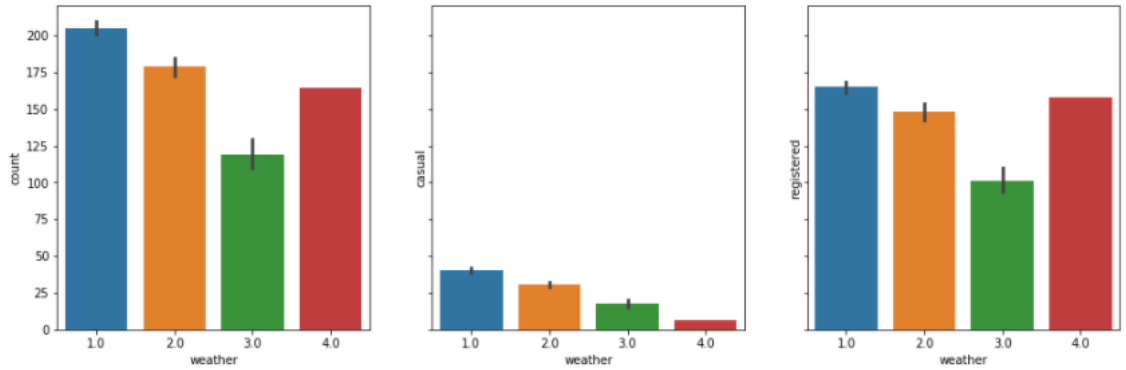
註冊的使用者則相反。

小時



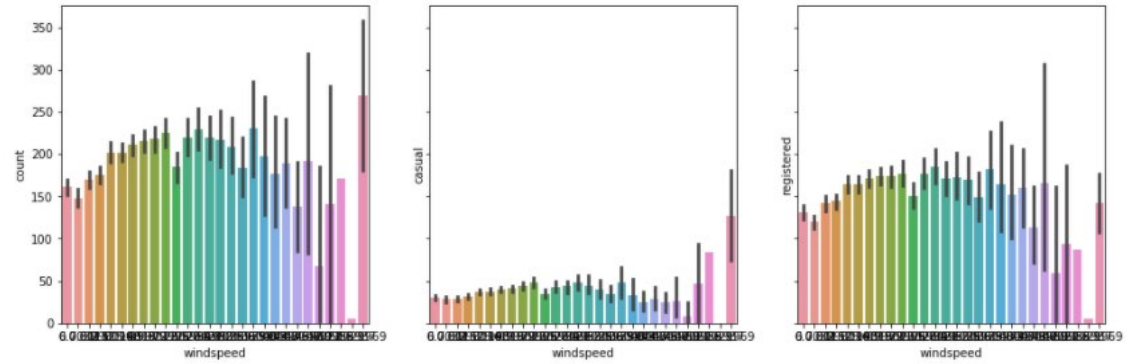
未註冊使用者使用高峰在 12-18，已註冊使用者的高峰則在上下班時間。

天氣



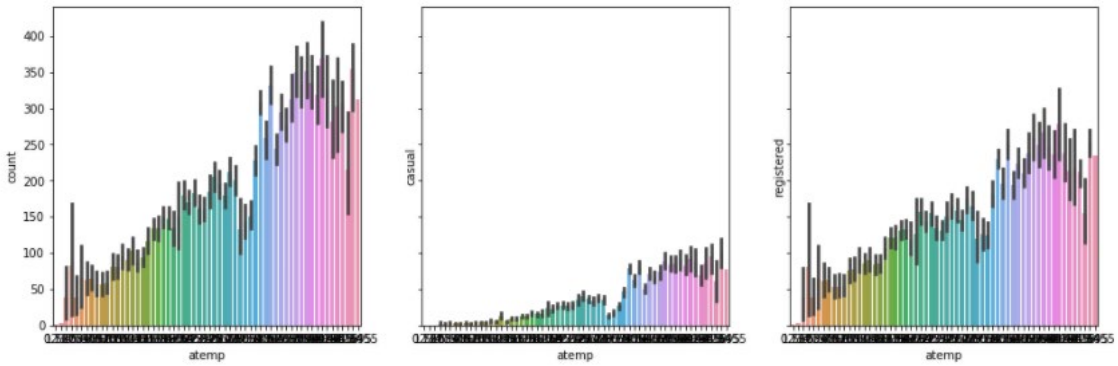
未註冊使用者傾向在好天氣使用，已註冊使用者除了 3(小雪)外，使用量差不多。

風速



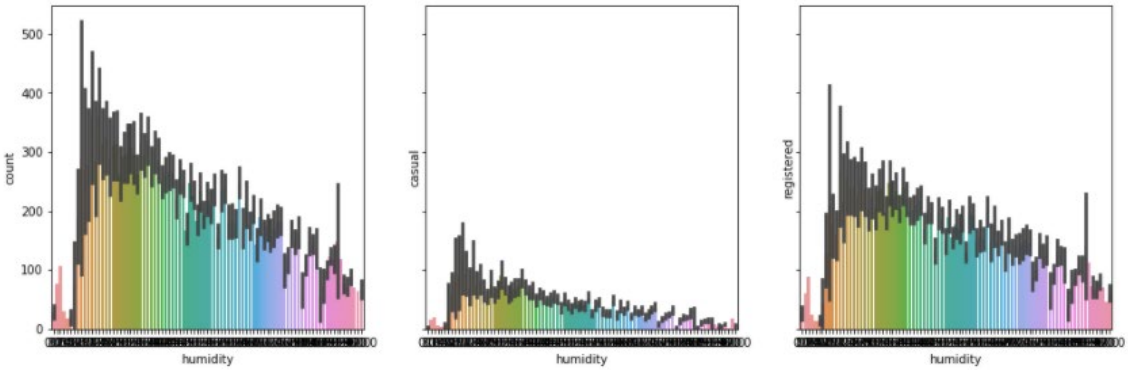
使用量大致相同除了某些離群值。

體感溫度



趨勢與溫度相符。

濕度



傾向在低濕度時使用，但過於乾燥時使用量會下降

考量天氣、節日、星期等欄位大多互相有關連，加上有些因子的影響不顯著。最終決定使用工作日、溫度、濕度、風速、年分、小時、季節、天氣 8 個因子。其中季節和天氣因子採用 one-hot encoding。

2. 模型建構與結果

本研究採用兩個模型:KNN 演算法與 Random Forest。

KNN 演算法

```
from sklearn.preprocessing import StandardScaler
#轉換資料
scaler = StandardScaler()
general_data = data[["casual_log", "count_log", "registered_log"]]
scaler.fit(general_data)
scaled = scaler.transform(general_data)
scaled = pd.DataFrame(data=scaled, columns = general_data.columns)

#轉換Log資料(KNN不支援)
scaled["casual_log"] = (scaled["casual_log"] * 10) .apply(np.floor)
scaled["registered_log"] = (scaled["registered_log"] * 10) .apply(np.floor)
scaled["count_log"] = (scaled["count_log"] * 10) .apply(np.floor)


general_data = general_data.join(data.drop(["casual_log", "count_log", "registered_log"],axis=1))
```

```
def build_kmodel(data, tuning_parameter):

    X_train, X_test, y_train, y_test = train_test_split(data.drop(["count_log", "registered_log","casu

    kmodel = KNeighborsRegressor(n_neighbors = tuning_parameter)
    kmodel.fit(X_train,y_train)
    pred=kmodel.predict(X_test)

    MSE = mean_squared_error(pred, y_test)

    return kmodel, MSE
```

其中 K value 以迴圈測試最佳解。

```
#finding the best k value
k_values = [3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30]

MSEs =[]
models = []

for k_value in k_values:
    model, MSE = build_kmodel(general_data, tuning_parameter = k_value)
    MSEs.append(MSE)
    models.append(model)
```

其中 k=6 的 MSE 為 0.56，為測試出的最佳解。

```
kmodel = KNeighborsRegressor(n_neighbors = 6)
kmodel.fit(general_data.drop(["count_log", "registered_log","casual_log"],axis=1), general_data["count_
pred=kmodel.predict(data_test)

submission = pd.DataFrame({
        "datetime": dt_test["datetime"],
        "count": [max(0, x-1) for x in np.exp(pred)]
    })
submission.to_csv('KNN.csv', index=False)
```

使用測試集資料進行預測，並提交到 Kaggle，所得的分數(RMSLE)為 0.78，在排行榜上約前 20%。

Random Forest

```
X_train, X_test, y_train, y_test = train_test_split(data.drop(["count_log", "registered_log","casual_l
rfmodel = RandomForestRegressor(n_estimators=3000, random_state=42, max_depth=30)
rfmodel.fit(X_train,y_train)
pred = rfmodel.predict(X_test)
MSE = mean_squared_error(pred, y_test)
MSLE = mean_squared_log_error(pred, y_test)
#測試集分數
print("RMSE =", MSE**0.5)
print("RMSLE =", MSLE**0.5)
```

```
RMSE = 0.3493234181071481
RMSLE = 0.09199942633926979
```

使用訓練集，得出 RMSE 為 0.35、RMSLE 為 0.09

```
rfmodel = RandomForestRegressor(n_estimators=3000, random_state=42, max_depth=30)
rfmodel.fit(data.drop(["count_log", "registered_log","casual_log"],axis=1),data["count_log"])

pred = rfmodel.predict(data_test)
submission = pd.DataFrame({
        "datetime": dt_test["datetime"],
        "count": [max(0, x-1) for x in np.exp(pred)]
    })
submission.to_csv('RF.csv', index=False)
```

在多次測試中，發現參數對模型表現影響不大，因此不設計參數調整。


```
rfmodel = RandomForestRegressor(n_estimators=3000, random_state=42, max_depth=30)
rfmodel.fit(data.drop(["count_log", "registered_log","casual_log"],axis=1),data["count_log"])

pred = rfmodel.predict(data_test)
submission = pd.DataFrame({
        "datetime": dt_test["datetime"],
        "count": [max(0, x-1) for x in np.exp(pred)]
    })
submission.to_csv('RF.csv', index=False)
```


使用測試集資料進行預測，並提交到 Kaggle，所得的分數(RMSLE)為 0.41，在排行榜上約前 10%。

本研究有嘗試將 casual 與 registered 使用者分開進行預測，但嘗試後的結果多不盡人意，因此不採用此種方法。

四、 結論與未來展望

因為共享單車此產業對於電子系統的依賴度很高，其資料的來源穩定、正確性也很高，因此本專案的可行性相當高。本專案的成果並不是最佳模型，未來希望能進一步改善。

惟此資料收集時間為近 10 年前，因子與模型的可靠性高低，我們無法知曉；再者，資料來源為美國的華盛頓特區，不一定合乎各地區的消費者行為，期望未來能夠與其他供應商合作，收集其他地區的相關數據，並且長遠且穩定的合作才能為企業與消費者創造雙贏局面。

參考資料

https://www.kaggle.com/c/bike-sharing-demand/