Definition

Overview

Rossmann 是歐洲的一家連鎖藥妝店。擁有一個可以準確預測與分析銷售額的模型對於 Rossmann 這類的連鎖商店來說非常重要。透過模型分析以及預測銷售量，商店可以適時的引進足夠的貨源避免貨源不足的情況，或是避免進貨過多銷售不完得情形。此外，由建立得模型可以預測在什麼情況下可以有最大的獲利，並朝此方向改變營業方針；例如可由過去的銷售情形得知在什麼情況下做何種促銷可使銷售額增加，獲取更多的利潤。

在本項目中將使用 Machine learning algorithm 中的 Random forest regressor 建立模型，並用此模型來預測 Rossmann 商店的銷售量，並分析影響銷售量的淺在因素。

Problem Statement

本項目將根據 Rossmann 在 Kaggle 上提供的 2013年1月到2015年7月 1115間 store 的數據 [1] ：促銷、節日、過去銷售情況等，建立一個預測與分析銷售額的模型預測 Rossmann 商店的銷售額。利用 kaggle 上給的 train.csv 與 store.csv 當作數據集，將其依比例隨機分成 80% 的 train data set 與 20 % 的 test data set。利用 train data set 來訓練模型，並使用 test data set 來進行預測並評分。

Metrics

本項目將會依照 Kaggle 裡 Rossmann Store Sales 競賽的要求使用 Root Mean Square Percentage Error (RMSPE) 來評估準確度 [1]。RMSPE 的定義如下：

$$\mathrm{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2},$$

y_i 是 test data set 裡面實際的銷售額; yhat_i 是預測值。 任何商店在任意天的 sales 為 0 時，忽略不計算準確度。

Analysis

Data Exploration

數據集來自於 Rossmann 在 Kaggle 上提供的 2013年1月到2015年7月 1115 間 store 的數據以及銷售 [1]。從 Kaggle 的 Rossmann Sales 比賽下載檔案： train.csv 與 store.csv 之後用 Python 讀取成 Pandas 的 DataFrame： train_data 與 store_data。

train_data 與 store_data 個別 display 最前面的兩個 row 如下圖 (Figure 1 (a) and (b))：

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
0	1	c	a	1270.0	9.0	2008.0	0	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0

Figure 1. (a) store_data

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1

Figure 1. (b) train_data

分析 train_data 與 store_data 的資料總數：

train_data：

RangeIndex: 1017209 entries, 0 to 1017208

Data columns (total 9 columns):

Store 1017209 non-null int64

DayOfWeek 1017209 non-null int64

Date 1017209 non-null object

Sales 1017209 non-null int64

Customers 1017209 non-null int64

Open 1017209 non-null int64

Promo 1017209 non-null int64

StateHoliday 1017209 non-null object
SchoolHoliday 1017209 non-null int64

store_data：

RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
Store 1115 non-null int64
StoreType 1115 non-null object
Assortment 1115 non-null object
CompetitionDistance 112 non-null float64
CompetitionOpenSinceMonth 761 non-null float64
CompetitionOpenSinceYear 761 non-null float64
Promo2 1115 non-null int64
Promo2SinceWeek 571 non-null float64
Promo2SinceYear 571 non-null float64
PromoInterval 571 non-null object

Exploratory Visualization

據數據裡不同 StoreType 與 Assortment 的數目顯示於 Figure 2 (a) and (b)。 其中 StoreType 裡 type a 的數目最多， type b 的數目最少。而 Assortment 中 type a 的數目最多 type b 的數目最少。

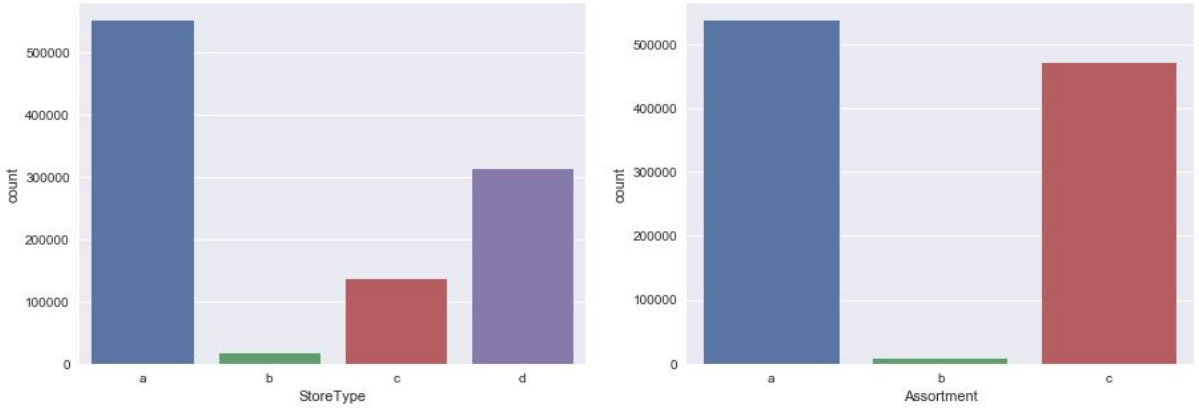


Figure 2. (a) StoreType vs. Count (b) Assortment vs. Count

將 StoreType 與 Assortment 合成同一個特徵 Storemode 後，可將不同類型的商店分成 9 類：aa、ac、dc、da、ca、cc、bb、ba 與 bc。由 Figure 3 可以看出， aa 的種類的數目遠遠高出其他種類，而 bc 種類的數目最低。

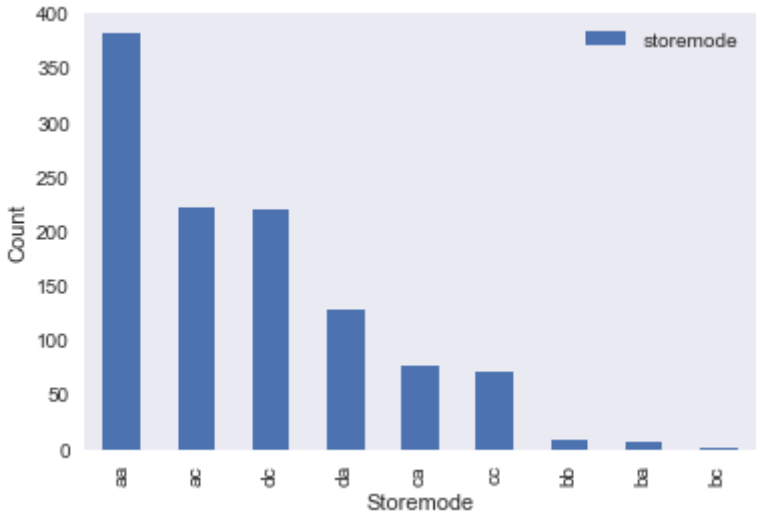


Figure 3 Storemode vs. Count

對於不同的 StoreType ， b 類型的 StoreType 不論是平均 Sales 或是平均 Customers 都是最高的。如 Figure 4 所示。

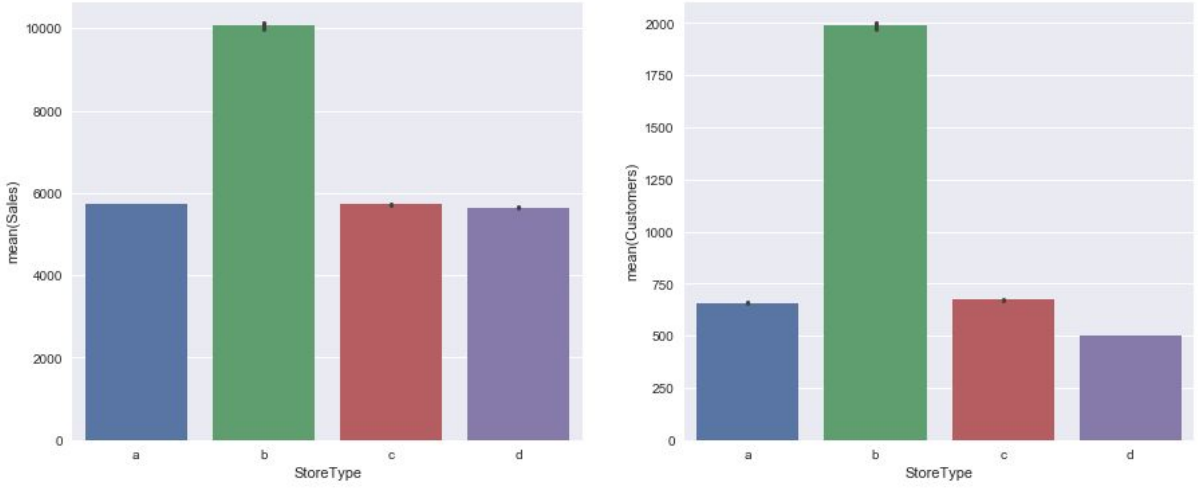


Figure 4. (a) StoreType vs. Mean Sales (b) StoreType vs. Mean Customers

將銷售額除以顧客人數後顯示於 Figure 5。 StoreType d 的平均銷售額與顧客數雖然遠小於 StoreType b，但是每個顧客的平均銷售額卻遠遠大於 StoreType b，這表示於 StoreType b 的

顧客的消費能力較為 StoreType d 低。但由於 StoreType b 的平均顧客人數比其他 StoreType 多很多，因此平均銷售額還是相較於其他 StoreType 高。

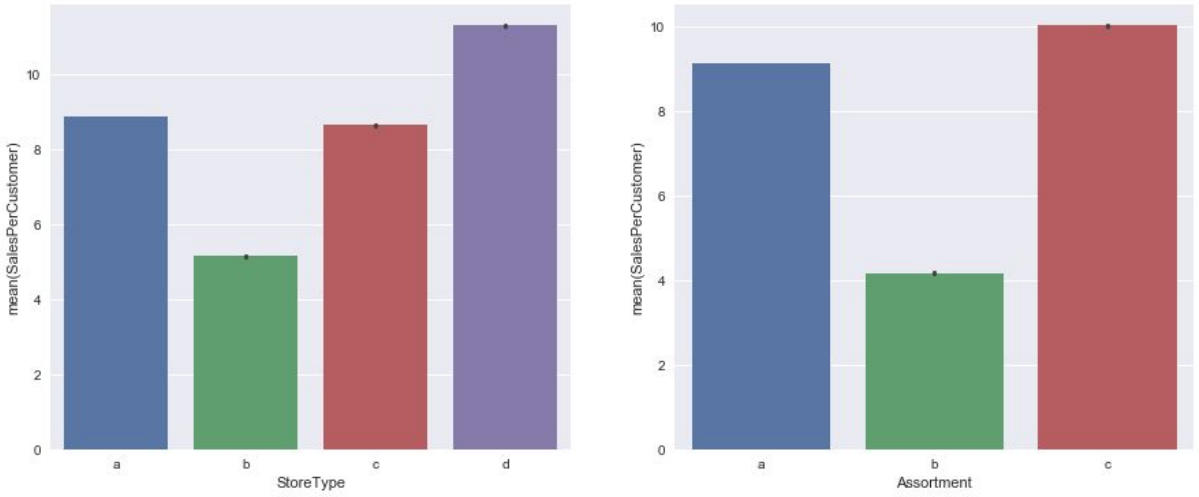


Figure 5. (a) StoreType vs. Mean SalsesPerCustomer (b) Assortment vs. Mean SalsesPerCustomer

然而對於 Sotremode 來說，每一種 Sotremode type 的平均顧客人數則差不多，如 Figure 6 所示

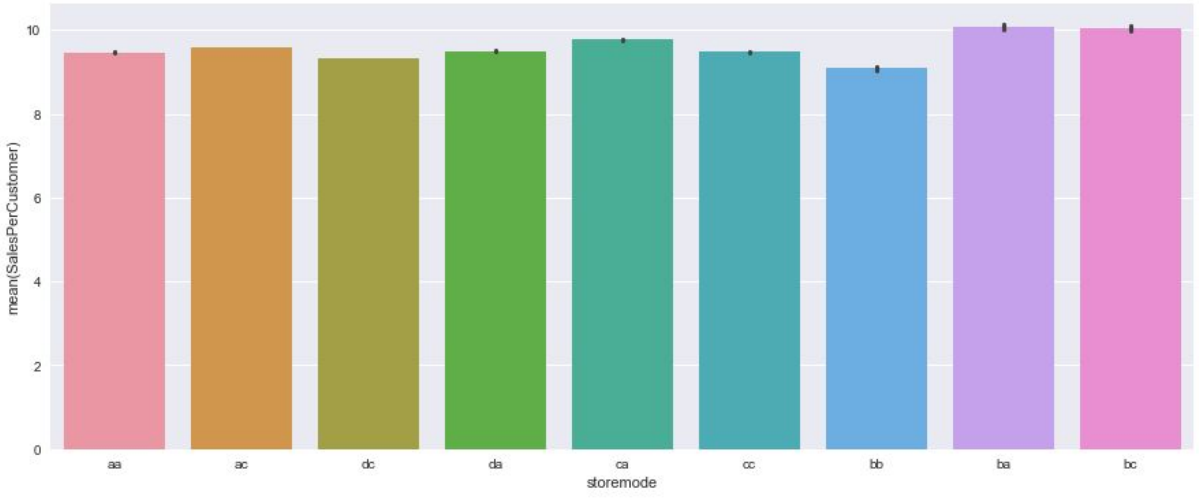


Figure 6. Storemode vs. Mean SalsesPerCustomer

然而從單日筆銷售額來看， Storemode bc 的最高銷售額明顯小於其他 Storemode ，如 Figure 7 所示。

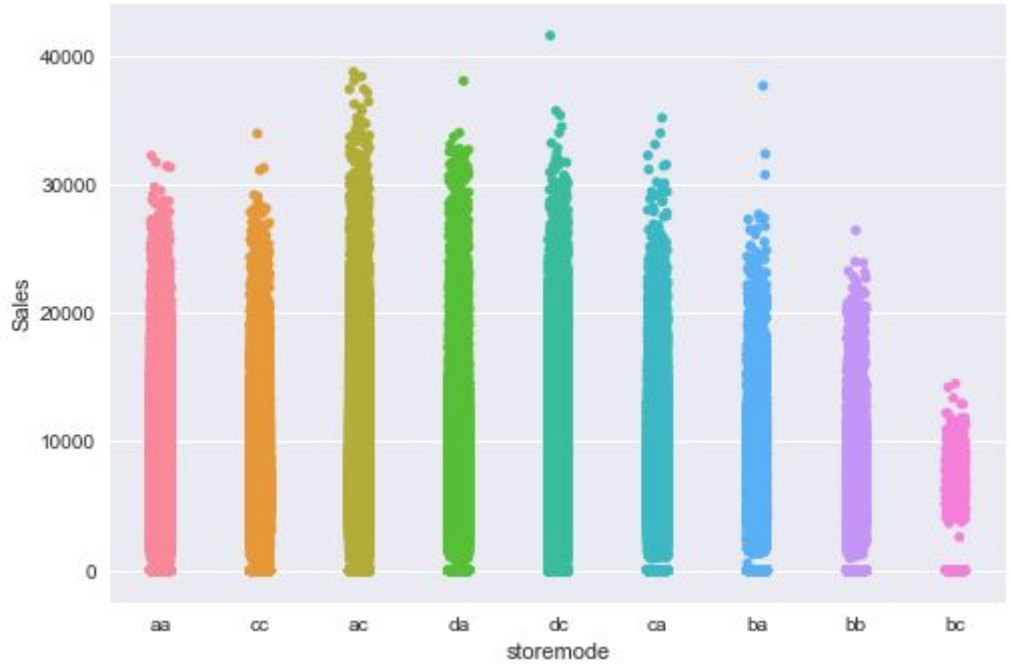


Figure 7 Storemode vs. Sales

Customers 與 Sales 的關係圖如 Figure 8 所示。可看出 Customer 數目大致上與 Sales 數目的 log 值成正比。

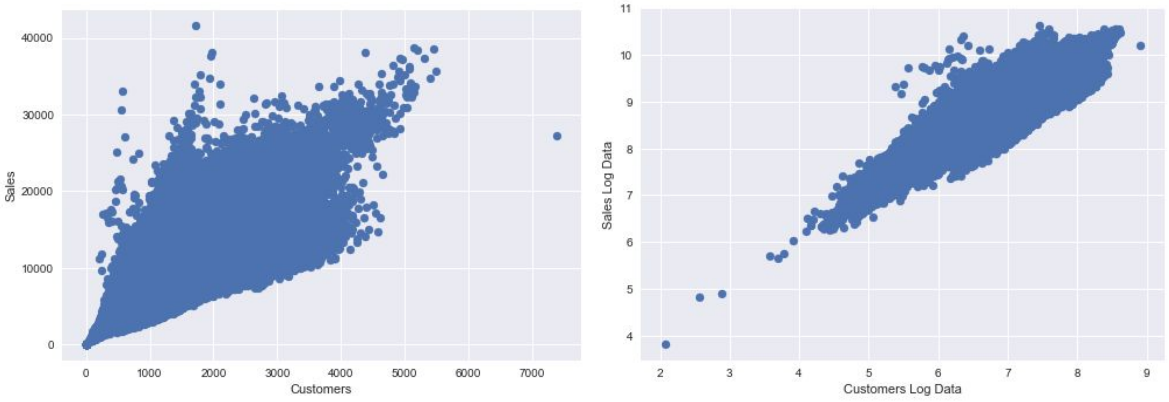


Figure 7. (a) Customers vs. Sales (b) Customers vs. Sales log data

Figure 8 為不同 CompetitionDistance 對 Sales 之圖。

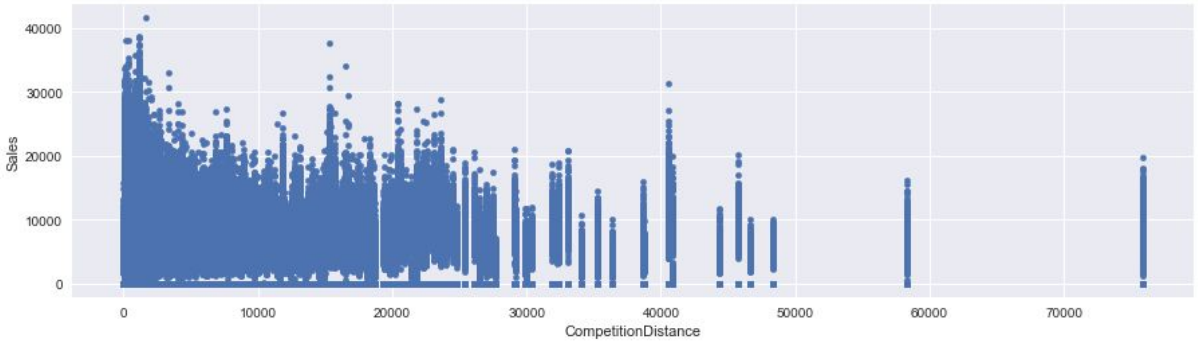


Figure 8. CompetitionDistance vs. Sales

Figure 9 為 CompetitionDistance 與不同特徵之關係圖。

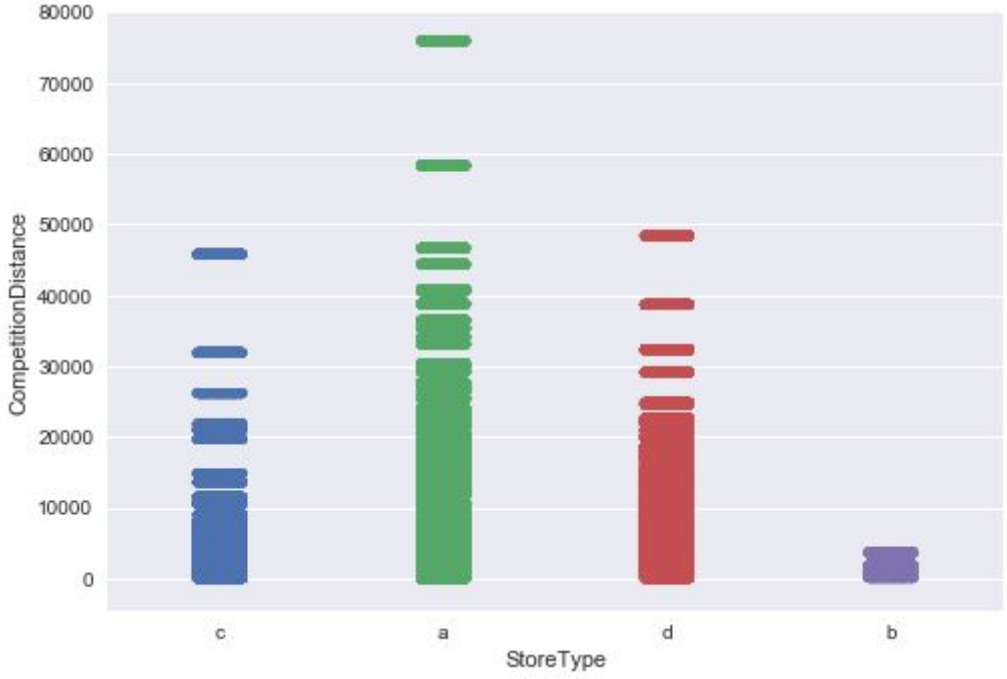


Figure 9. (a) StoreType vs. CompetitionDistance

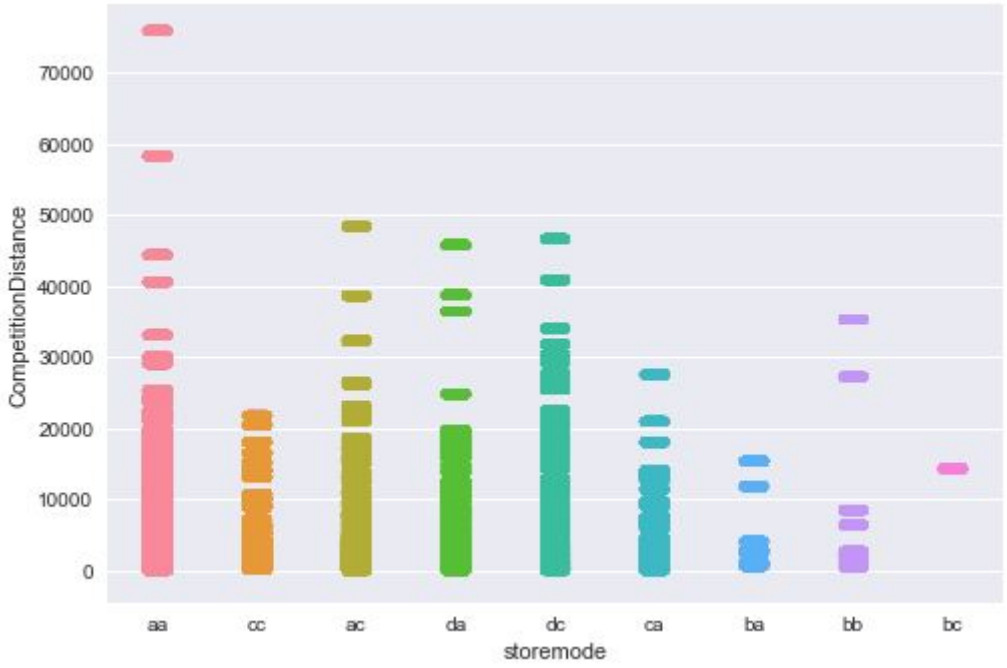


Figure 9. (b) Storemode vs. CompetitionDistance

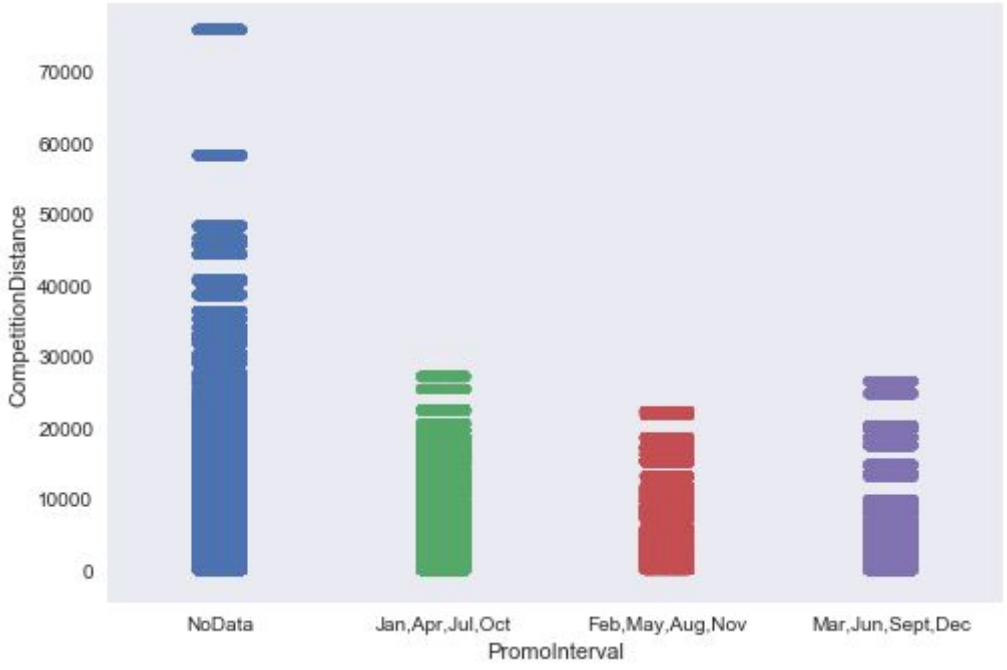


Figure 9. (c) PromoInterval vs. CompetitionDistance

Figure 10 為 Promo2 與 CompetitionDistance 和 Sales 之關係圖。 由圖可知有 Promo2 的商店競爭者的距離都較近，但銷售額卻一定比沒有 Promo2 的商店高。

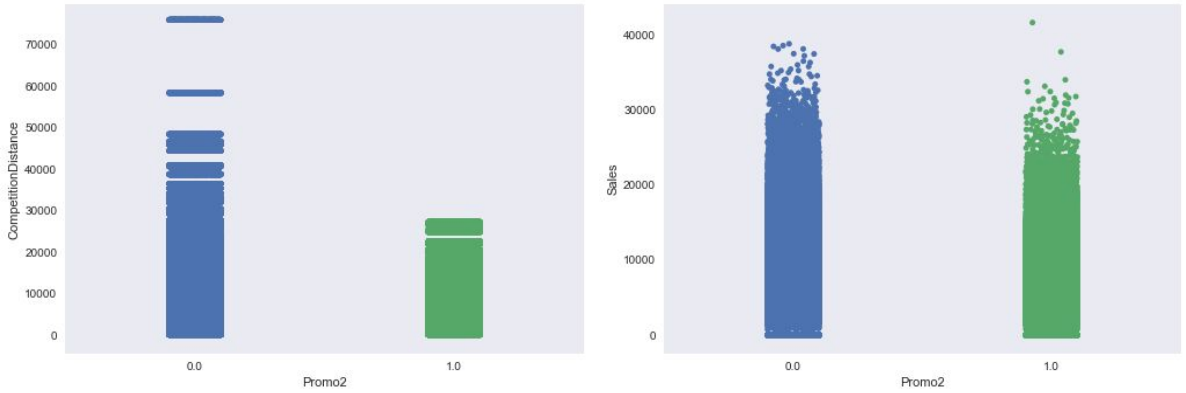


Figure 10. (a) Promo2 vs. CompetitionDistance (b) Promo2 vs. Sales

Figure 11 顯示除了 DayOfWeek 是 7 之外的店的 Open 數目，其他種類的 Open 數目皆差不多。

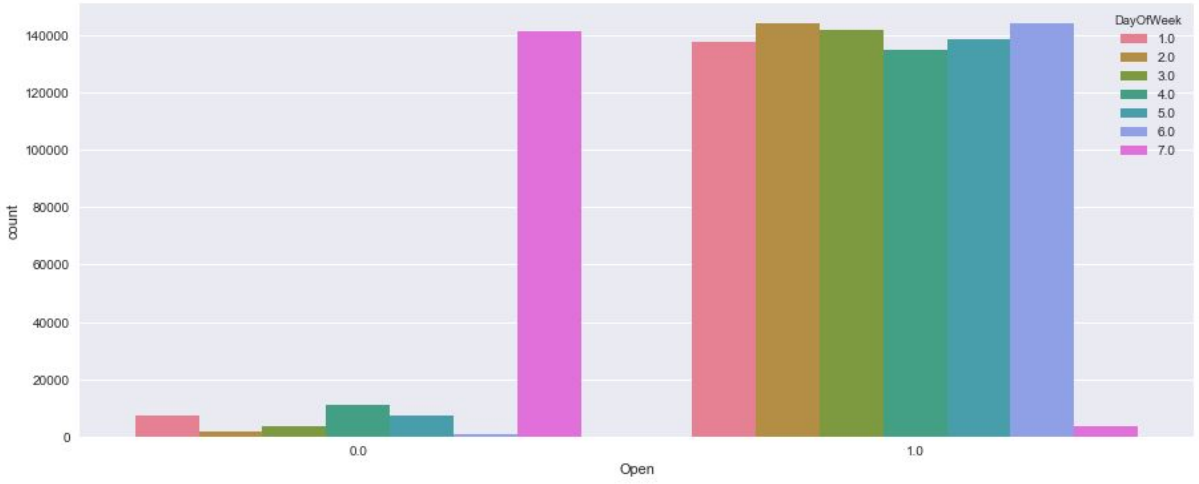


Figure 11. Open vs. Counts of DayOfWeek

每月總銷售量與總客人數對日期作圖顯示於 Figure 12。

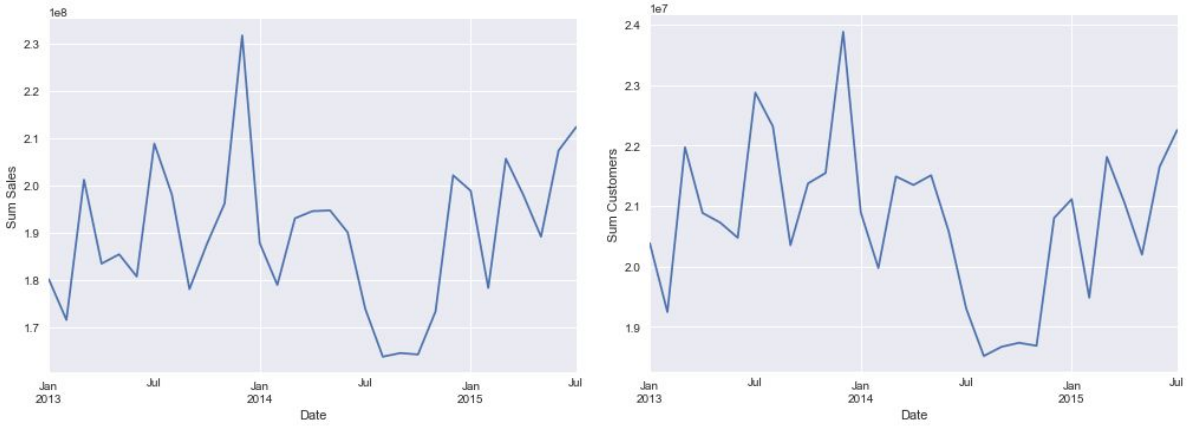


Figure 12. (a) Date vs. Sum Sales (b) Date vs. Sum Custmers

每月每個客人平均消費對日期作圖顯示於 Figure 13。 由圖可看出，12 月時每個可人的平均消費皆比其他月份高出許多。

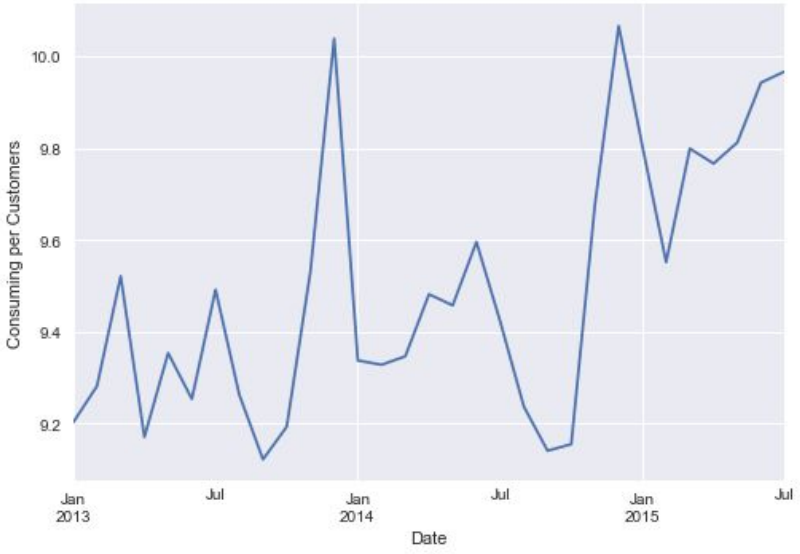


Figure 13 Date vs. Consuming per Customers

Figure 14 為不同 StoreType 與 Assortment 每月平均銷售額對日期圖。

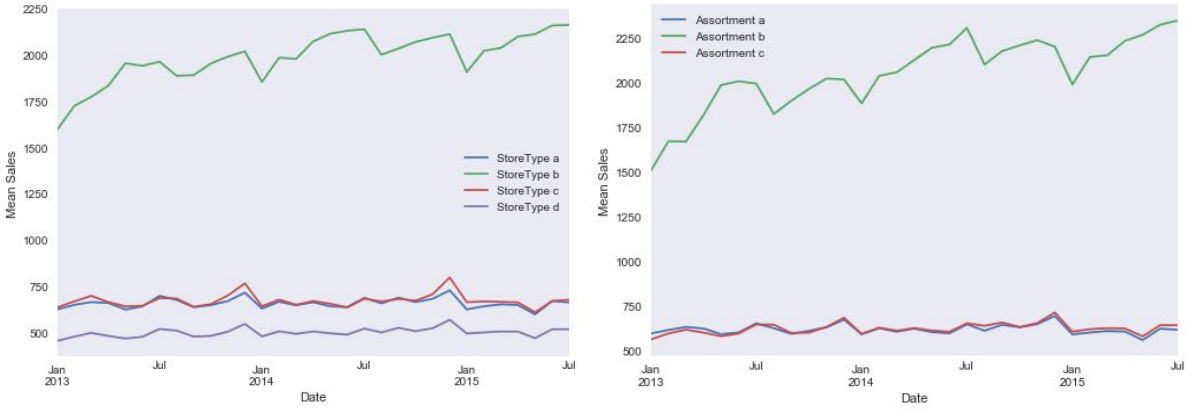


Figure 14. (a) Date vs. Mean Sales of StoreType (b) Date vs. Mean Sales of Assortment

Figure 15 為不同 Storemode 每月平均銷售額與每月總銷售額對日期圖。

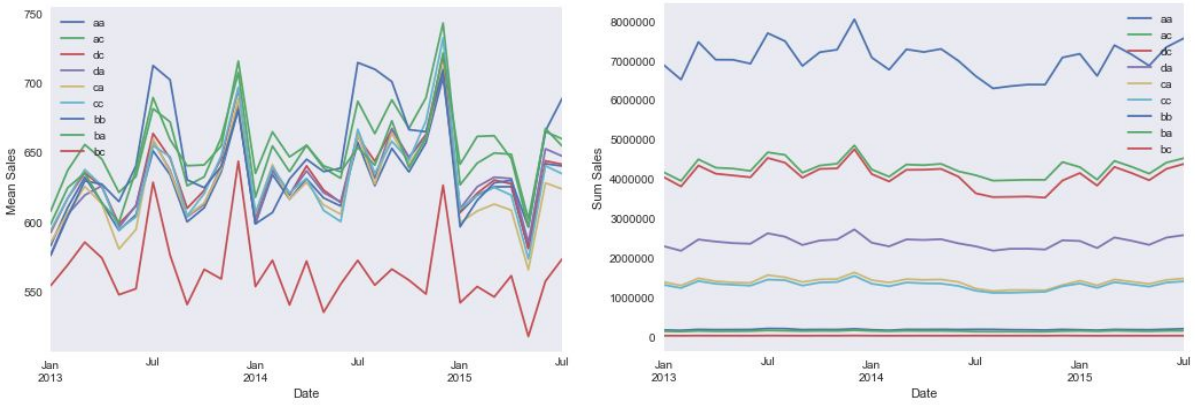


Figure 15. (a) Date vs. Mean Sales of Storemode (b) Date vs. Sum Sales of Storemode

Algorithms and Techniques

本篇報告使用 Python scikit-learn package 裡的 RandomForestRegressor 模型 [2] 來預測。模型中改動的參數如下：

- **n_estimators :** The number of trees in the forest.
- **max_depth :** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

並且使用預測後得到的 feature importances 來評估每個特徵的重要性。

Benchmark

本篇報告將每間店的平均值算出，作為評估模型性能的基礎模型。 由平均值評估出的結果為:
RMSPE Score = 0.454630163643

Methodology

Data Preprocessing

- 由於數據集裡的 “Date” 用 Python 讀取後為字串，顯示為：yy-mm-dd，為了將時間也考慮進去當作參數，在此將字串 “yy-mm-dd” 分成三組數字分別於三欄裡。 三欄分別為 “year”、 “month” 與 “date”。
- 根據 Metrics 的要求將 'Sales' 為 0 的數據去除。

- 由於 'Customers' 與 ‘Sales’ 一樣為未知數，因此將 'Customers' 由數據中去除。
- 由於 'StateHoliday'、’Store' 與 'Promo2SinceWeek' 為不重要的特徵，因此將其從數據中去除。
- 將 catagorical 與 numerical 的 data 區分出來。
- 用 ’nan’ 取代 catagorical 的 missing data ; 用 numerical data 的中位數取代 numerical 的 missing data。
- 對 catagorical 的 data 做 edcoding，並對 numerical 的 data 做歸一化。

Implementation

我使用 sklearn 裡面的 train_test_split 將 feature 與 sales 80% 的數據隨機分成訓練集與 20 % 的測試集，並將 random_state 固定設為 26。 使用 ShuffleSplit 將訓練集分成 10 份做 grid search method，找出最佳的 n_estimators 與 max_depth 的參數。
使用模型預測後得到的 feature importances 來評估每個特徵的重要性，並將不重要的 feature 刪除增加預測的準確度。

Result and Refinement

一開始我使用 Python scikit-learn package 裡的 RandomForestRegressor 模型，在並未更改 defalt 的參數的情況下得到：
RMSPE Score: 0.14257490707

Feature Importances:

Promo0 0.081223
CompetitionDistance 0.077850
DayOfWeek 0.064863
Promo1 0.054088
date 0.052364
month 0.046696
CompetitionOpenSinceYear 0.027683
Store816 0.021944
Promo2SinceWeek 0.020370
StoreType1 0.019793
CompetitionOpenSinceMonth 0.019771
Store1113 0.018070
year 0.015371
Promo2SinceYear 0.015252
Store261 0.014275
Store512 0.012955
Store250 0.012486

Store787 0.011469
Store841 0.010355
Assortment2 0.009064
Store382 0.008742
Assortment0 0.008205
Store697 0.007639
Store755 0.006587
Store379 0.006480
Store594 0.006316
StoreType0 0.005945
PromoInterval2 0.005603
Store561 0.005594
Store522 0.005333
... ...
Store459 NaN
Store234 NaN
Store316 NaN
Store620 NaN
Store370 NaN
Store85 NaN
Store243 NaN
Store1094 NaN
Store203 NaN

Feature Importances

由 Feature Importances 可以看出 year 造成的影響雖然不是最高的，但是依然會對模型造成不小的影響。因此我嘗試改變 year 的特徵表示方式，將原本 year 的絕對值改為與 2015的差值，得到結果小幅度增進的準確度：
RMSPE Score: 0.142493192859

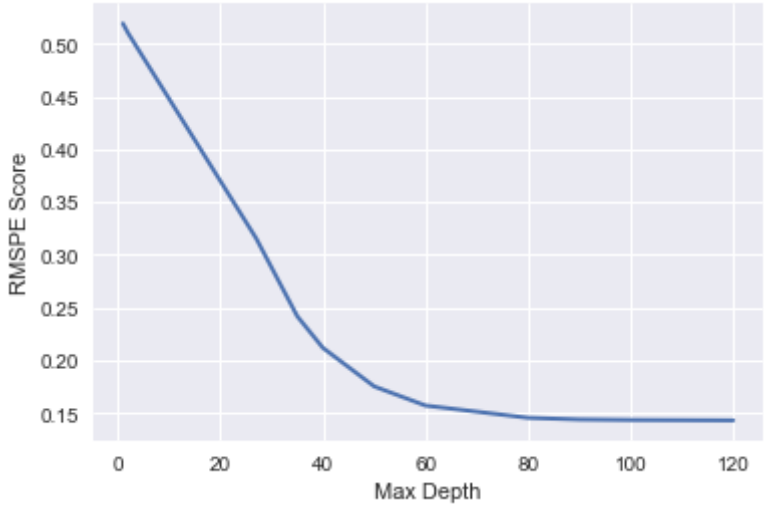
雖然 'StateHoliday' 與 'Promo2SinceWeek' 由 Feature Importances 看起來不會影響模型的準確度，然而捨棄 StateHoliday' 與 'Promo2SinceWeek' 之後，模型的準確率反而降低了：
RMSPE Score: 0.145531199504

Max Depth

在其他條件不變的情況下，我調整了 max_depth 得到 RMSPE Score 值列於下表：

Max Depth	RMSPE Score	Run Time
Defalt	0.142493192859	983.490607977
1	0.519101118894	X
2	0.510105350151	X
27	0.315338383689	X

35	0.241596348998	X
40	0.21165359609	X
50	0.175109607934	X
60	0.156937563399	X
80	0.145316793498	X
90	0.143742410698	978.979128122
100	0.14327112879	975.916754007
120	0.142898617358	1026.82690096



由表可以看出使用 defalt 的 max_depth 可以達到最優之結果，而實際 max_depth 的值應是介於 100-200之間。因此之後模型之訓練都使用 max_depth = defalt。

N Estimators

將 max_depth 設為 defalt，比較不同 n_estimators 的值結果顯示於下表：

N Estimators	RMSPE Score	Run Time
Defalt (n = 10)	0.142493192859	983.490607977
n = 100	0.136281158672	10260.5755961

由表可以看出，當 n 增加 10 倍時，Run Time 也大約增加了 10 倍。 而 RMSPE Score 明顯變得更好了。在此由於繼續增加 n 之後的運行時間太長，所以並未測試 n > 100 的 RMSPE Score。然而由於訓練集裡的特徵超過 1000 個，所以可以預期的是當 n 繼續增加時，應是可以再優化 RMSPE Score。

Conclusion

本篇報告優化模型後得到的最佳結果為 RMSPE Score = 0.136281158672，相較於基礎模型得到的 RMSPE Score: 0.454630163643 優化非常多。 而相較於 Rossmann 在 Kaggle 上的競賽一名得到的 RMSPE Score = 0.10021 還是差了一些。以下是我推測還可以改進模型的幾個方式：

- n_estimators 值過小，繼續增加 n_estimators 可以再優化模型的準確度。
- Kaggle 比賽裡預測的數據為其給予數據的之後 6 週的數據。 如果考慮到只預測最新 6 週的數據，由於每年的消費趨勢都有些不同將一些最早期的數據去除後有機會增加準確率。

Reference

[1] https://www.kaggle.com/c/rossmann-store-sales/data

[2]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html