Data Analysis of Sales of Australian Grocery Store

Author: Zilin Huang, Vivian Han, Albert Hutiancong Wang, Daniel Luo

Background

All Foods is an Australian grocery store (similar to Costco) that sells a variety of groceries including fresh farm products.

In 2017, the store operated 7/24. However, in 2018, the store changed its strategy by only opening from 7.am to 7.pm daily. The problem is that this dropped their revenues by more than $50,000 and profits by more than $40,000.

Our objective is to analyze the past sales data to identify some trends that explain the loss of earnings.

Data Definition

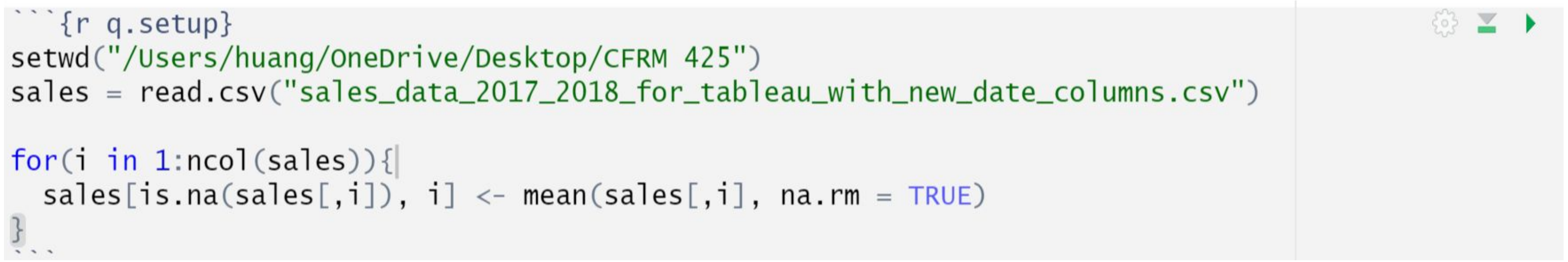
The dataset below provides details of Australian Grocery Store All Goods in 2017-2018:

sales_data_2017_2018_for_tableau_with_new_date_columns.csv

The dataset contains 23 variables, including:

- Items bought per receipt id
- selling time, date, and year
- Item prices and profits
- Main, sub category

Data Cleaning


~~~
```{r q.setup}
setwd("/Users/huang/OneDrive/Desktop/CFRM 425")
sales = read.csv("sales_data_2017_2018_for_tableau_with_new_date_columns.csv")

for(i in 1:ncol(sales)){
  sales[is.na(sales[,i]), i] <- mean(sales[,i], na.rm = TRUE)
}
```
~~~

Fill all the null values inside the data set with the mean values of the corresponding column

(inside an iterated for loop)

Data Analysis

Question: Which items should the store stop selling? Why? (this somehow explains the loss of profits for All Foods)

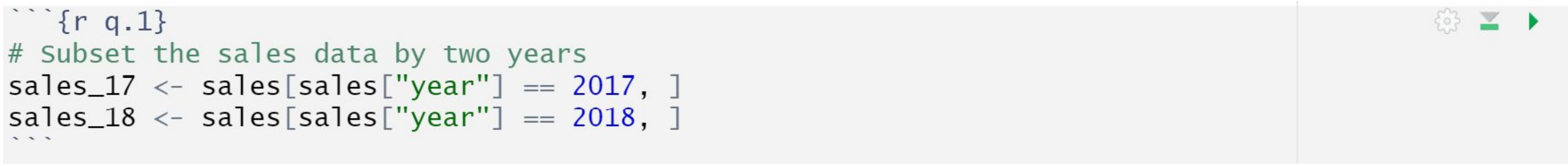
Metrics for items to not be sold are:

1) their purchasing frequency

2) their total profits generated

For simplicity, we only analyze the former case.

Data Analysis

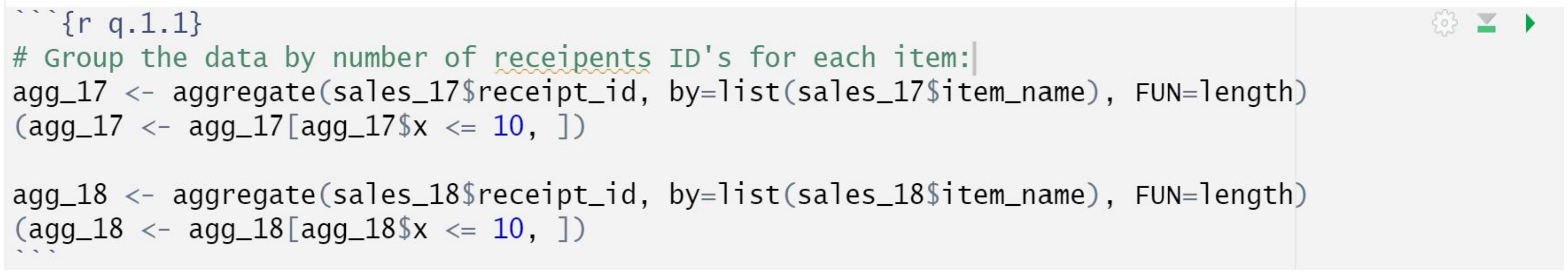


~~~
```{r q.1}
# Subset the sales data by two years
sales_17 <- sales[sales["year"] == 2017, ]
sales_18 <- sales[sales["year"] == 2018, ]
```
~~~

Subset the data by years to observe the yearly differences of sales

(these data will also be utilized for the visualization part)

Data Analysis

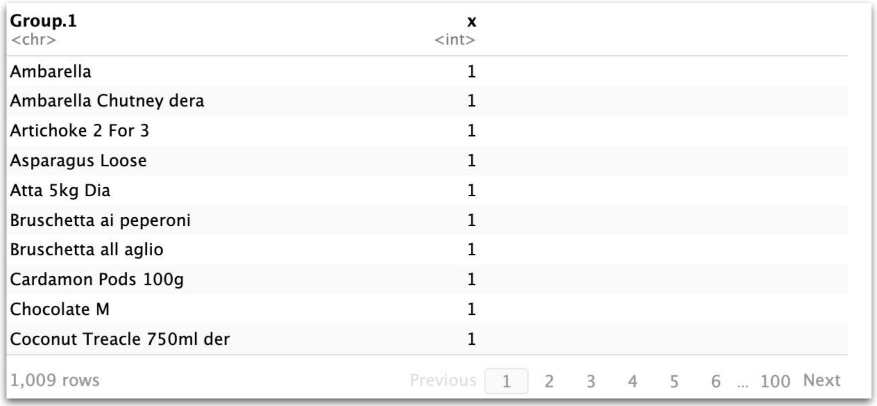


~~~
```{r q.1.1}
# Group the data by number of receipents ID's for each item:
agg_17 <- aggregate(sales_17$receipt_id, by=list(sales_17$item_name), FUN=length)
(agg_17 <- agg_17[agg_17$x <= 10, ])

agg_18 <- aggregate(sales_18$receipt_id, by=list(sales_18$item_name), FUN=length)
(agg_18 <- agg_18[agg_18$x <= 10, ])
```
~~~

For both years' sales data, group by number of receipt ID (which indicates how much time is the item sold) for each item

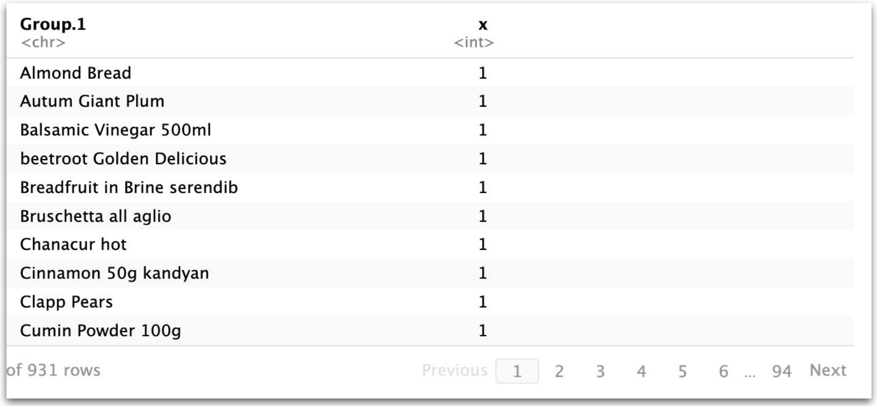
Assume that items with less than **10** purchasing times in a year should not be considered to be further sold. (this is a really low purchasing frequency regardless of the actual price of the items)

Data Analysis



Group.1 <chr>	x <int>
Ambarella	1
Ambarella Chutney dera	1
Artichoke 2 For 3	1
Asparagus Loose	1
Atta 5kg Dia	1
Bruschetta ai peperoni	1
Bruschetta all aglio	1
Cardamon Pods 100g	1
Chocolate M	1
Coconut Treacle 750ml der	1

1,009 rows Previous 1 2 3 4 5 6 ... 100 Next



Group.1 <chr>	x <int>
Almond Bread	1
Autum Giant Plum	1
Balsamic Vinegar 500ml	1
beetroot Golden Delicious	1
Breadfruit in Brine serendib	1
Bruschetta all aglio	1
Chanacur hot	1
Cinnamon 50g kandyan	1
Clapp Pears	1
Cumin Powder 100g	1

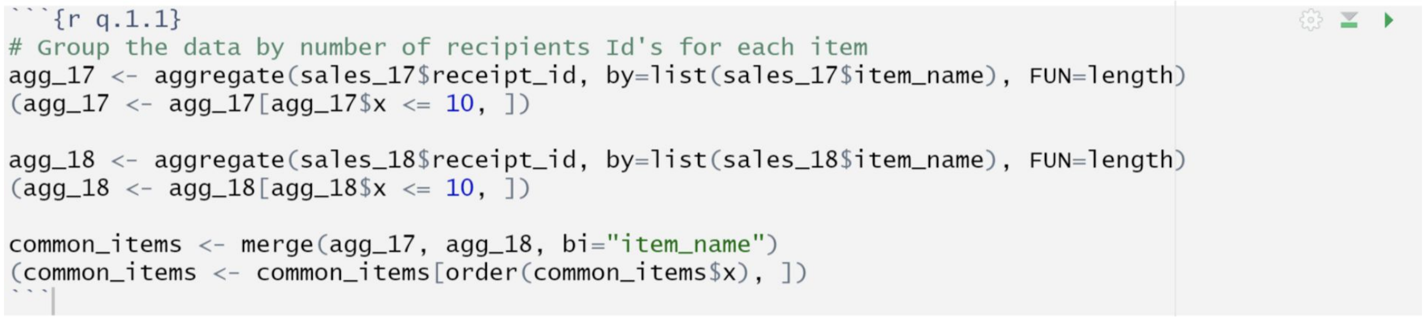
of 931 rows Previous 1 2 3 4 5 6 ... 94 Next

One issue to notice in the output is that in both years, many items have the same **x** value (number of times it is sold), implying that they were all only appeared once in receipt.

To avoid repetitions, we decide to set a range and merge data from 2017 and 2018 to retrieve the common items appearing in 2017 and 2018’s sales data.

Before editing the code, we **estimate** that in both years, around ⅓ of the goods in All Foods were purchased less than 10 times per year, which is considered to not be sold.

Data Analysis

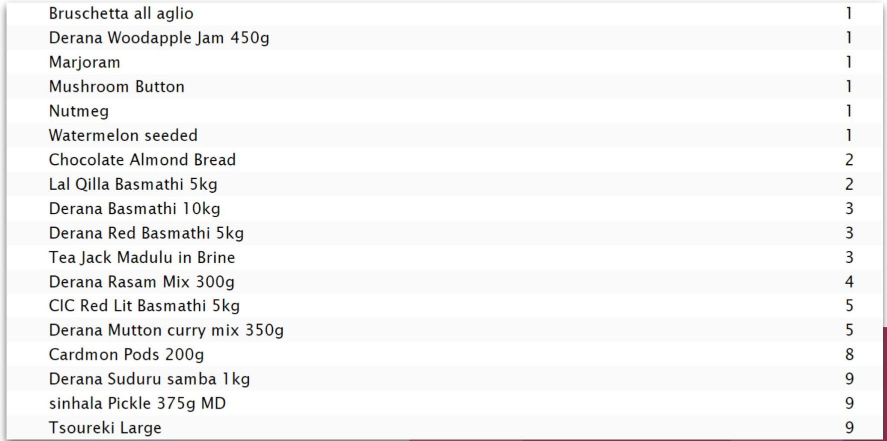


~~~
```{r q.1.1}
# Group the data by number of recipients Id's for each item
agg_17 <- aggregate(sales_17$receipt_id, by=list(sales_17$item_name), FUN=length)
(agg_17 <- agg_17[agg_17$x <= 10, ])

agg_18 <- aggregate(sales_18$receipt_id, by=list(sales_18$item_name), FUN=length)
(agg_18 <- agg_18[agg_18$x <= 10, ])

common_items <- merge(agg_17, agg_18, bi="item_name")
(common_items <- common_items[order(common_items$x), ])
```
~~~

Applying common_item and merge, we found 18 rows, which is 18 items should be stopped selling in All Goods. (see right side), which is less than ⅓ in our estimation.

These items share the smallest value in receipt id in 2017 - 2018.

We **recommend** that All Foods should make decision not only based on current year, but also combining previous years.



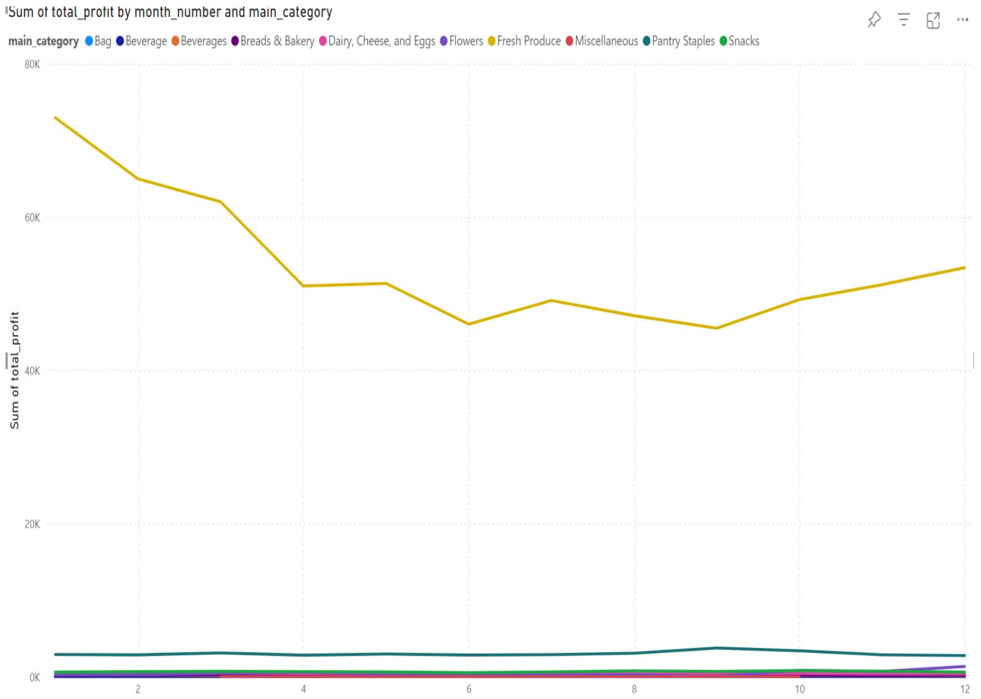
Bruschetta all aglio	1
Derana Woodapple Jam 450g	1
Marjoram	1
Mushroom Button	1
Nutmeg	1
Watermelon seeded	1
Chocolate Almond Bread	2
Lal Qilla Basmathi 5kg	2
Derana Basmathi 10kg	3
Derana Red Basmathi 5kg	3
Tea Jack Madulu in Brine	3
Derana Rasam Mix 300g	4
CIC Red Lit Basmathi 5kg	5
Derana Mutton curry mix 350g	5
Cardmon Pods 200g	8
Derana Suduru samba 1kg	9
sinhala Pickle 375g MD	9
Tsoureki Large	9

Data Visualization

Question: What trends do you notice in item categories & sub-categories?

We will answer this question by analyzing monthly trends of total profits for each (sub)category. We will also summarize the percentage of quantity purchased in weekdays or weekends for each (sub)category in Power BI.

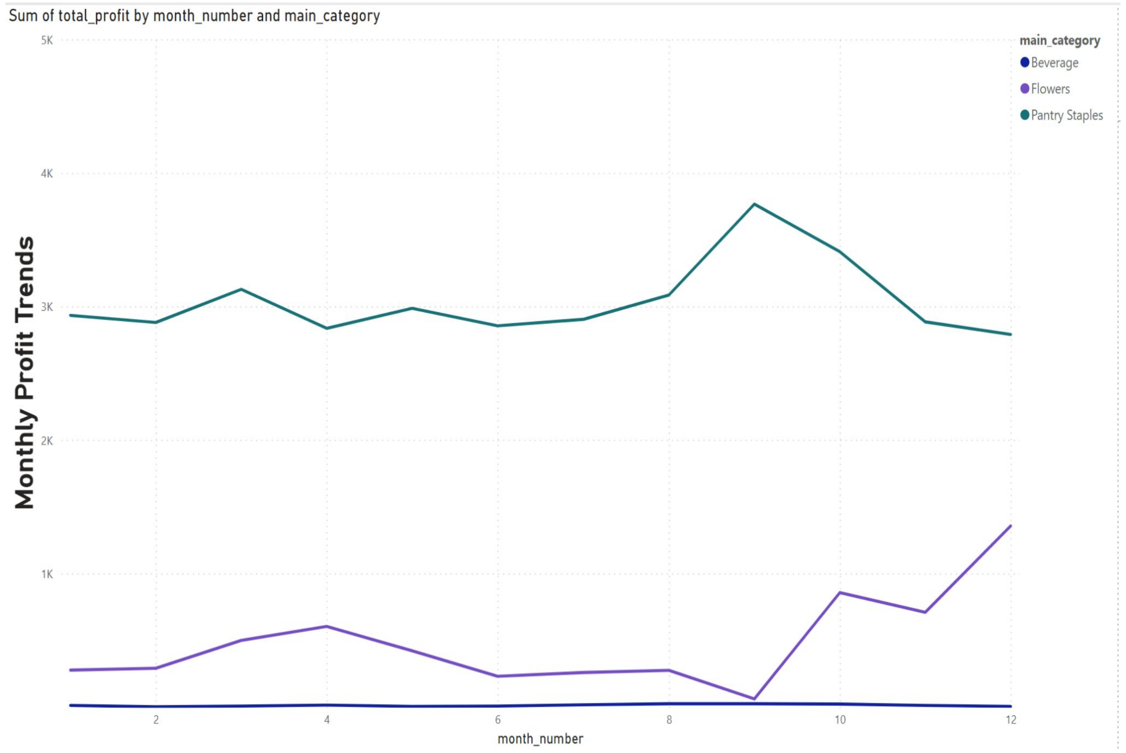
Data Visualization: Line Chart



This chart shows that the total profit of **Fresh Produce** per month is way more higher than the remaining categories. This indicates that the main business of All Foods is producing fresh goods.

Notice that the trend of Fresh Produce profit is decreasing from January to June, which corresponds to the fact that farming in Australia's winter (June to August) is less profitable.

Data Visualization: Line Chart (Partial)



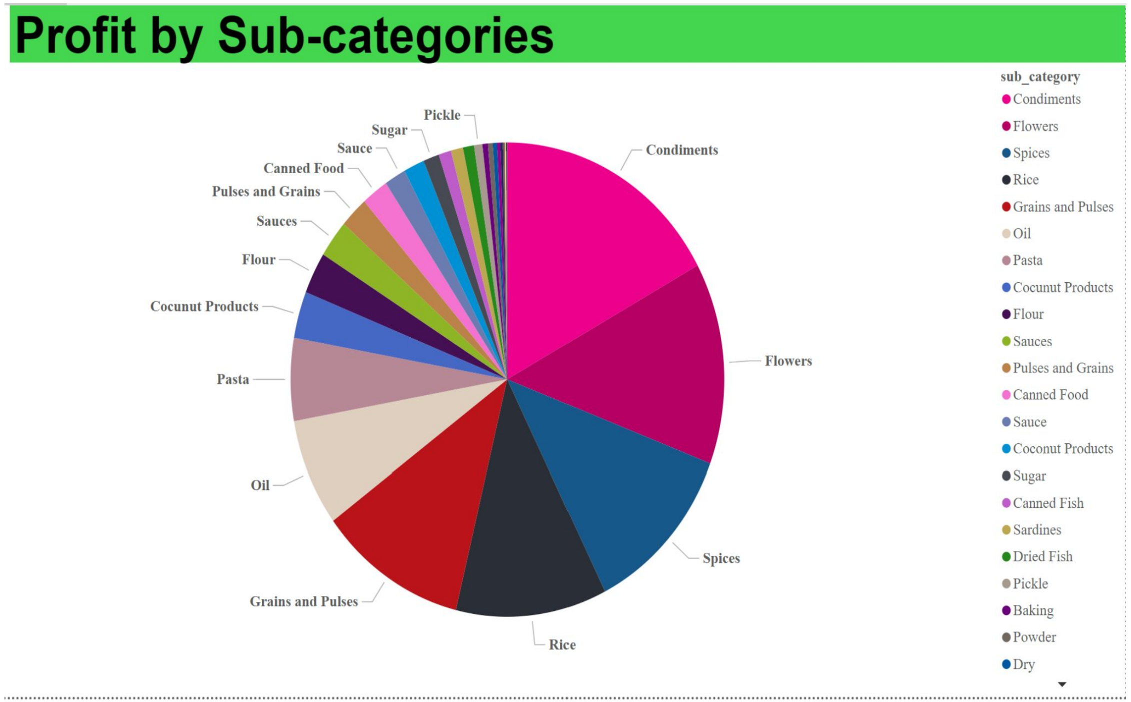
By just analyzing the monthly profit trends of these categories:
Beverage, Flowers, Pantry Staples,

We found that All Foods focus least on selling Beverages (and is invariant of the seasons).

Meanwhile, All Foods tend to sell more flowers in Australia's Summer, considering the fact that flower's agriculture is more productive during summer.

Besides, All Foods put more efforts in selling Pantry Staples than other categories of goods, indicating that this category is also agriculture-related (its profit is only less than that of Fresh Produce)

Data Visualization: Pie Chart



This chart shows that for all sub-categories, spices-related products (Spices and Condiments) occupy the largest percentage of total profits in total, followed by Rice and Grains.

This somehow agrees with the line chart, which shows that Fresh Produces and Spices generate the highest profits among all main categories.

Machine Learning

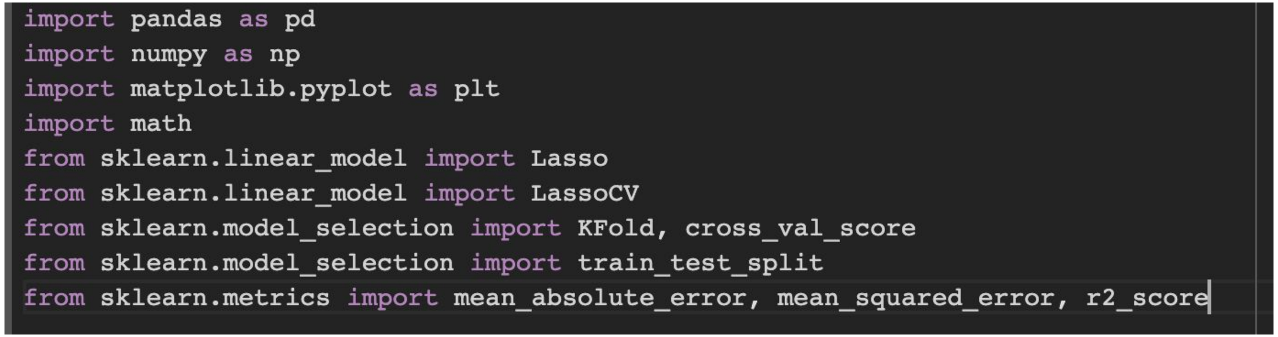
Objective: Predict the total amount of sales (quantity/selling price/buying price) in the next 3 months. => the store as a whole

1. The ML models we will be using (based on data speciality)
   a. Considering:
      i. Decision Trees
      ii. Regularization Regression Models (LASSO/Ridge)
      iii. XGBoost
      iv. Random Forest
   b. We choose LASSO with Cross Validation since we decide to include too many parameters needing to be filtered throughout the process
      i. Cross Validation helps model selection

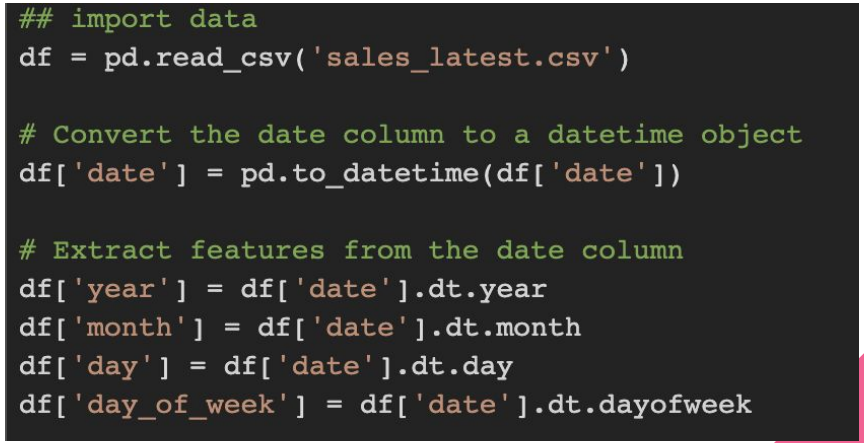
LASSO CV Model Procedure

1. Using LASSO CV to predict sales quantity of the store
2. Steps:
   a. Start-up
      i. Import necessary Python Packages
      ii. Data Processing and Cleaning
   b. Set up Machine Learning Model
      i. Model Selection
      ii. Train & Test
      iii. Model Evaluation (RMSE as margin of error & R^2 as accuracy)
         1. CV Plots
   c. Forecasting Based on ML Model
      i. Extension: Deep Learning with LSTM Model for Time-Series Data

1. Start-up and feature creation

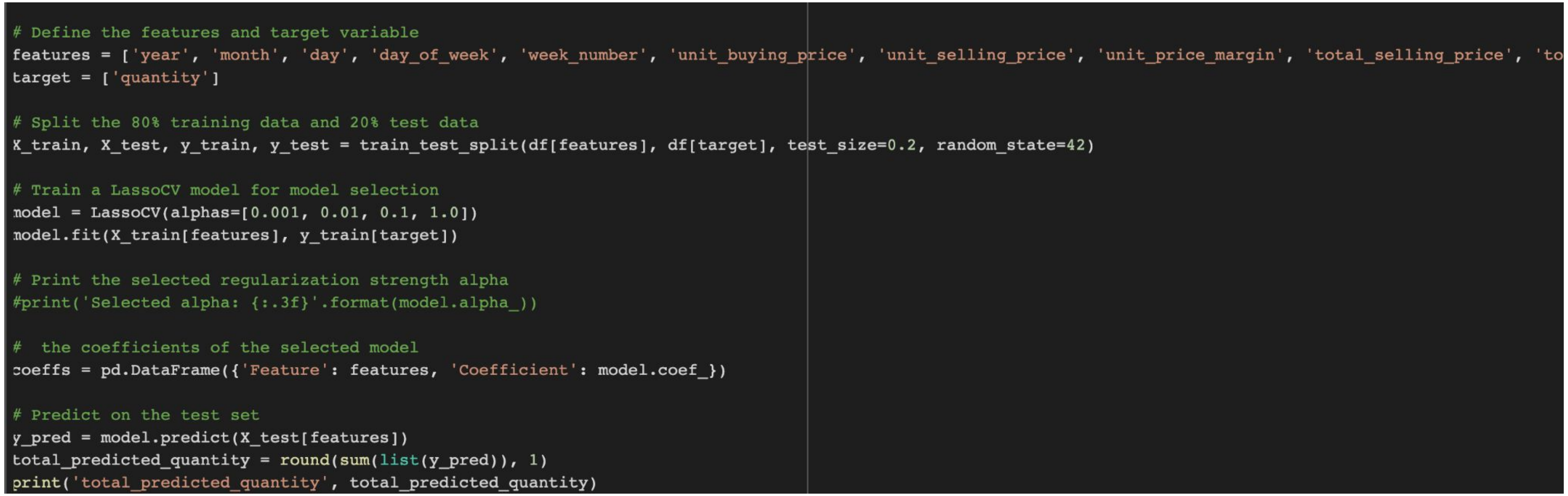


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```



```python
## import data
df = pd.read_csv('sales_latest.csv')

# Convert the date column to a datetime object
df['date'] = pd.to_datetime(df['date'])

# Extract features from the date column
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
df['day_of_week'] = df['date'].dt.dayofweek
```

2. ML Model



```
# Define the features and target variable
features = ['year', 'month', 'day', 'day_of_week', 'week_number', 'unit_buying_price', 'unit_selling_price', 'unit_price_margin', 'total_selling_price', 'to
target = ['quantity']

# Split the 80% training data and 20% test data
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)

# Train a LassoCV model for model selection
model = LassoCV(alphas=[0.001, 0.01, 0.1, 1.0])
model.fit(X_train[features], y_train[target])

# Print the selected regularization strength alpha
#print('Selected alpha: {:.3f}'.format(model.alpha_))

#  the coefficients of the selected model
coeffs = pd.DataFrame({'Feature': features, 'Coefficient': model.coef_})

# Predict on the test set
y_pred = model.predict(X_test[features])
total_predicted_quantity = round(sum(list(y_pred)), 1)
print('total_predicted_quantity', total_predicted_quantity)
```

We select essential features as independent variables and “quantity” as target variable.

Fit the model using LassoCV.

3. Model Evaluation and Forecasting

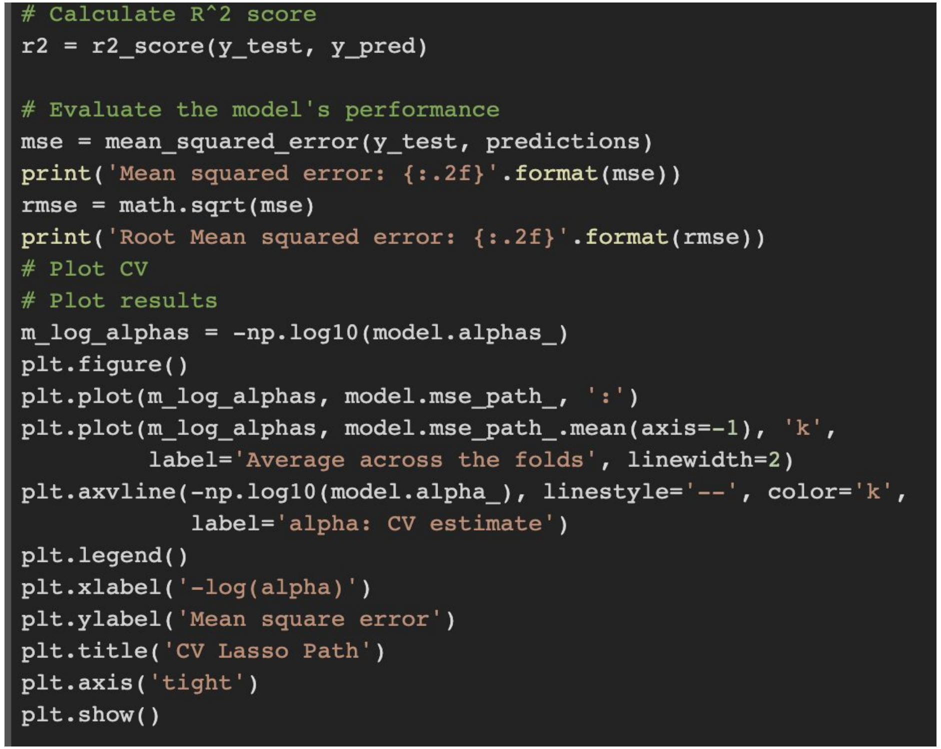
Since we approach this question by regression strategy, we decide to evaluate the model based on RMSE and R^2.

RMSE for margin of errors

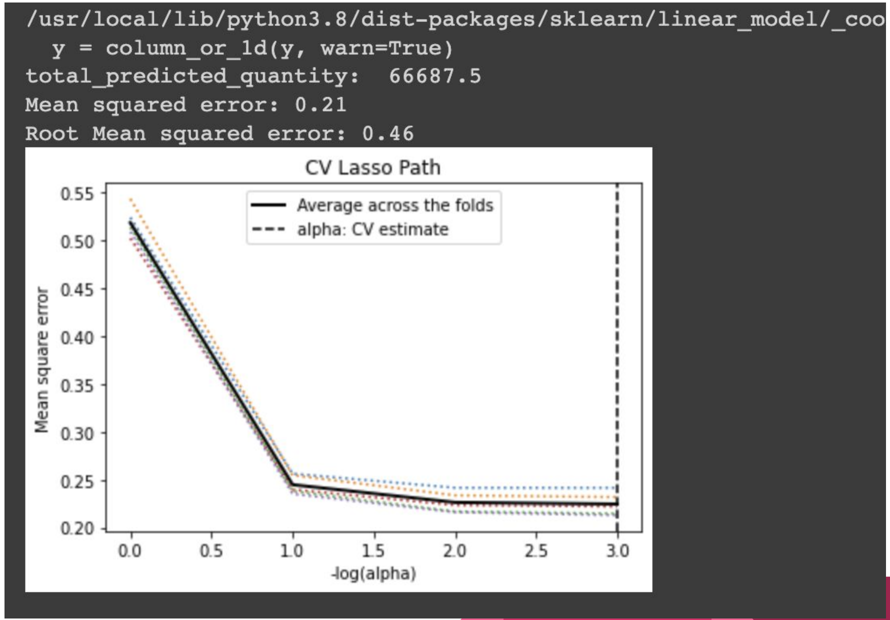
R^2 for proportion of explanation of variance

Based on independent variables

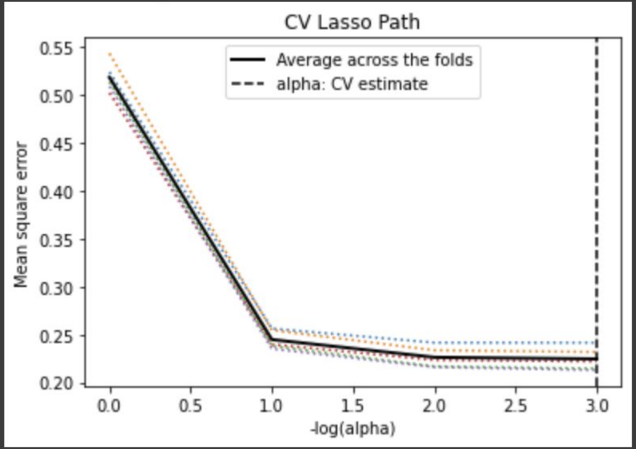
Evaluation of the model



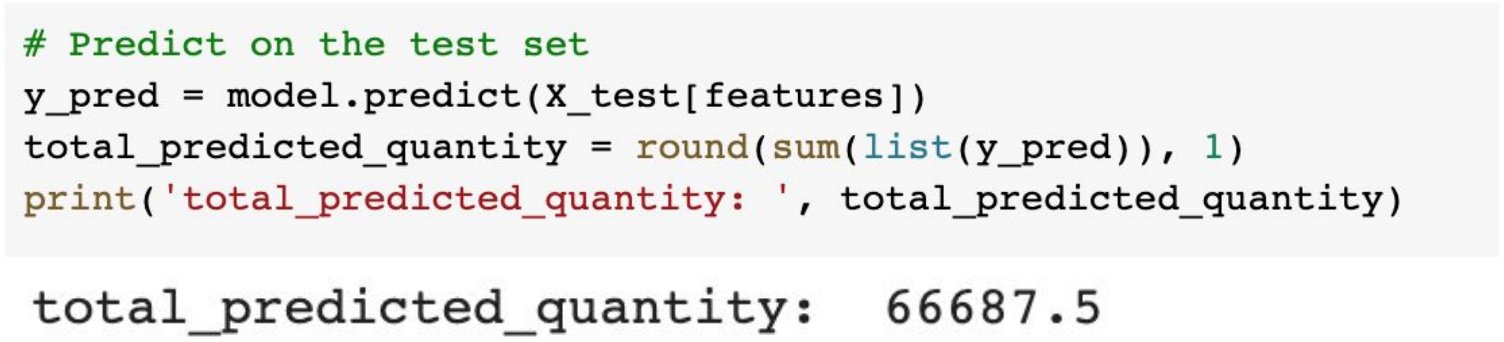
```
# Calculate R^2 score
r2 = r2_score(y_test, y_pred)

# Evaluate the model's performance
mse = mean_squared_error(y_test, predictions)
print('Mean squared error: {:.2f}'.format(mse))
rmse = math.sqrt(mse)
print('Root Mean squared error: {:.2f}'.format(rmse))
# Plot CV
# Plot results
m_log_alphas = -np.log10(model.alphas_)
plt.figure()
plt.plot(m_log_alphas, model.mse_path_, ':')
plt.plot(m_log_alphas, model.mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha: CV estimate')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean square error')
plt.title('CV Lasso Path')
plt.axis('tight')
plt.show()
```



```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_coo
  y = column_or_1d(y, warn=True)
total_predicted_quantity:  66687.5
Mean squared error: 0.21
Root Mean squared error: 0.46
```



Predict the total amount of quantity in next 3 months



```
# Predict on the test set
y_pred = model.predict(X_test[features])
total_predicted_quantity = round(sum(list(y_pred)), 1)
print('total_predicted_quantity: ', total_predicted_quantity)
```

```
total_predicted_quantity:  66687.5
```

Extension: Deep Learning with LSTM

1. Necessary Package Installation
2. Data Processing
   a. Resampling sales data into daily scales (segmentation)
   b. Split data into 80% Training data and 20% Test Data
   c. Scale and Standardized Data
   d. Set up LSTM Model for Time-Series Use
      i. Build
      ii. Train
      iii. Predictions
      iv. Evaluation
      v. Visualization
   e. Our RMSE : 0.19