Retail Sales Forecasting

Springboard Data Science Career Track
Dylan C Neal
9/1/20

Introduction:

Sales forecasts are essential information for financial planning. Without accurate sales forecasting, it is difficult to plan effectively for the future. By anticipating future sales, companies are able allocate resources to reduce waste and optimize revenue. Executives will be able to plan for major purchases while having confidence they will make the sales to be able to afford them.

Problem:

A major national retail chain has been keeping weekly sales data on each department in 45 of its stores. Historic sales data is limited, and only goes back ~3 years. Can a model be built to forecast sales data for each department in every store?

Step 1 – Data Wrangling:

- The data consist of 3 tables:
    1. SalesDF - Weekly sales data on every department in each store. Includes IsHoliday feature which indicates whether there is a major holiday that week. Holidays include the Super Bowl, Labor Day, Thanksgiving, and Christmas.

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	05/02/2010	24924.50	False
1	1	1	12/02/2010	46039.49	True
2	1	1	19/02/2010	41595.55	False
3	1	1	26/02/2010	19403.54	False
4	1	1	05/03/2010	21827.90	False

Figure 1: SalesDF.head()

```
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  object
 3   Weekly_Sales  421570 non-null  float64
 4   IsHolidav     421570 non-null  bool
```

Figure 2: SalesDF.info()

2. StoreDF - Details including the size and type of each store. Stores are categorized as type A, B, or C stores.

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

Figure 3: StoreDF.head()

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Store   45 non-null     int64
 1   Type    45 non-null     object
 2   Size    45 non-null     int64
```

Figure 4: StoreDF.info()

3. FeaturesDF - Additional weekly data for each store. Data include the average temperature and fuel price in the region, consumer price index, and unemployment rate. There are also 5 markdown features with numerical data. However, markdown data is not present for every store and is only available after November 2011.

#	Column	Non-Null Count	Dtype
0	Store	8190 non-null	int64
1	Date	8190 non-null	object
2	Temperature	8190 non-null	float64
3	Fuel_Price	8190 non-null	float64
4	MarkDown1	4032 non-null	float64
5	MarkDown2	2921 non-null	float64
6	MarkDown3	3613 non-null	float64
7	MarkDown4	3464 non-null	float64
8	MarkDown5	4050 non-null	float64
9	CPI	7605 non-null	float64
10	Unemployment	7605 non-null	float64
11	IsHoliday	8190 non-null	bool

Figure 5: FeaturesDF.info()

- The date columns were converted to datetime type. FeaturesDF was joined to SalesDF on the Date, Store, and IsHoliday features. StoresDF was joined to the main dataframe on the Store feature. Info on the joined dataframe is displayed below.

#	Column	Non-Null Count	Dtype
0	Store	421570 non-null	int64
1	Dept	421570 non-null	int64
2	Date	421570 non-null	datetime64[ns]
3	Weekly_Sales	421570 non-null	float64
4	IsHoliday	421570 non-null	bool
5	Temperature	421570 non-null	float64
6	Fuel_Price	421570 non-null	float64
7	MarkDown1	150681 non-null	float64
8	MarkDown2	111248 non-null	float64
9	MarkDown3	137091 non-null	float64
10	MarkDown4	134967 non-null	float64
11	MarkDown5	151432 non-null	float64
12	CPI	421570 non-null	float64
13	Unemployment	421570 non-null	float64
14	Type	421570 non-null	object
15	Size	421570 non-null	int64

Figure 6: .info() of Joined Data

- The only features containing null values are MarkDowns. Rather than removing nearly 75% of the data, it was decided to ignore markdown data and build the model without it.

Step 2 – Exploratory Data Analysis:

- The data were sorted by date, and then grouped by Store & Date. Weekly sales mean was aggregated and then plotted by store.
- A few of the line plots are seen below. While there is seasonality, it does not look like there is much trend in the data. If the data were over a longer span of time, there would most likely be observable trend.

Figure 7: Aggregate Mean Weekly Sales by Store – Line Plot Snippet

- To get a sense of store sales spread, a distribution plot was also made.

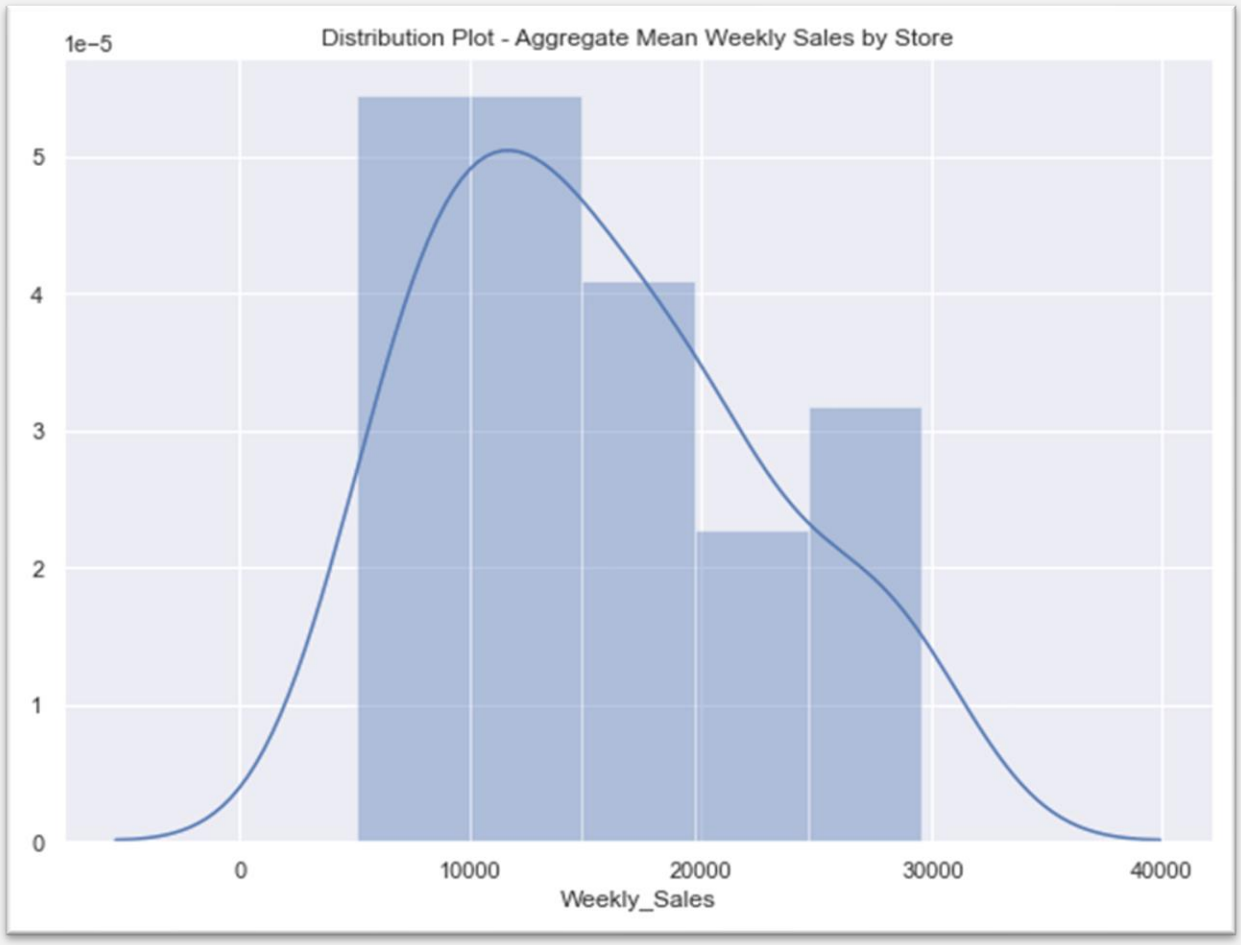

Figure 8: Aggregate Mean Weekly Sales by Store - Distribution Plot

- The data were then grouped by Store, Type, and Size, and weekly sales means were aggregated. From this a scatterplot was created to see the relationship between store size and weekly sales. Points were hued by type.

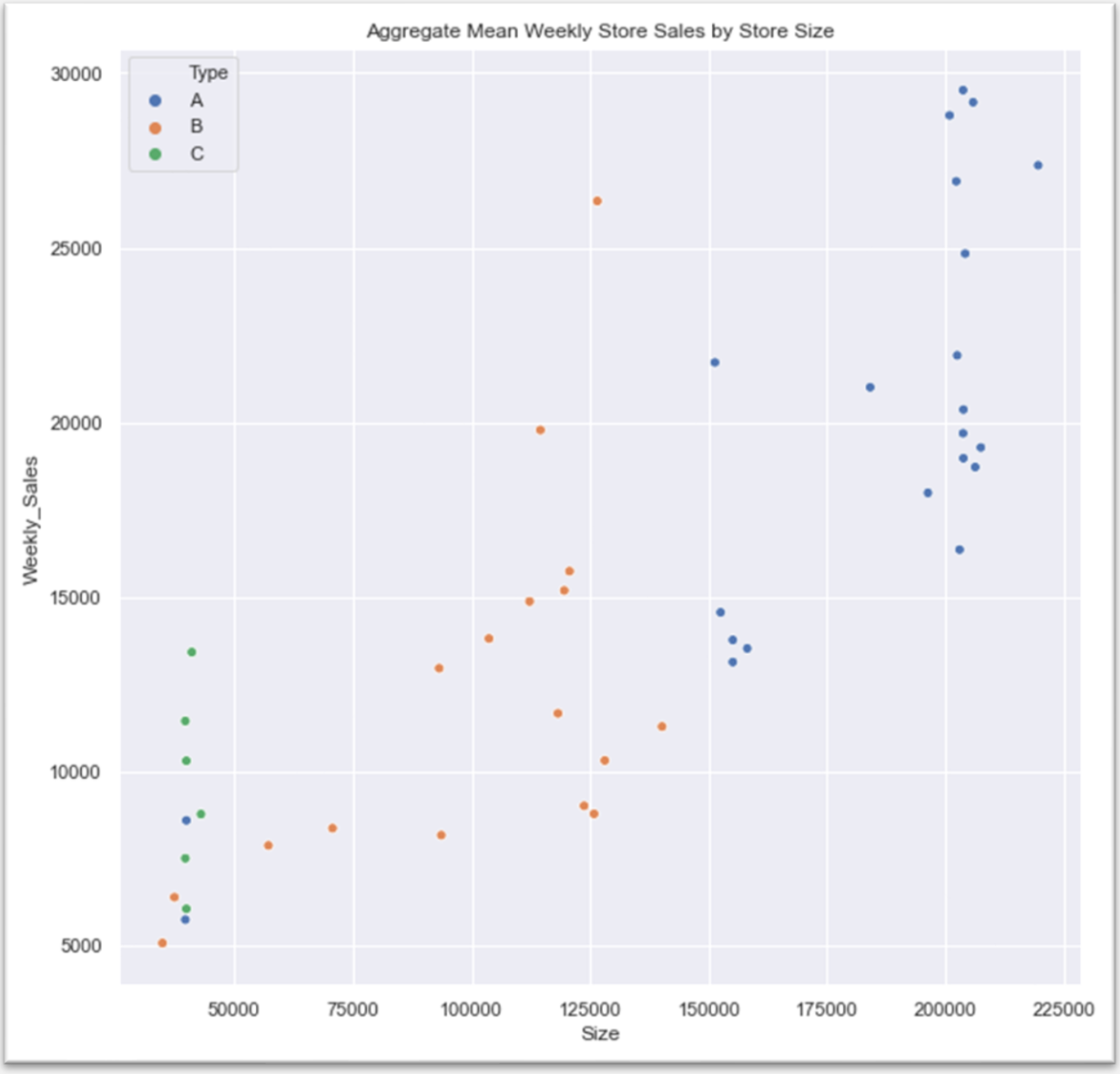

Figure 9: Aggregate Mean Weekly Sales by Store Size Hued by Store Type - Scatterplot

- A strip plot of aggregate mean weekly sales by store type was made.

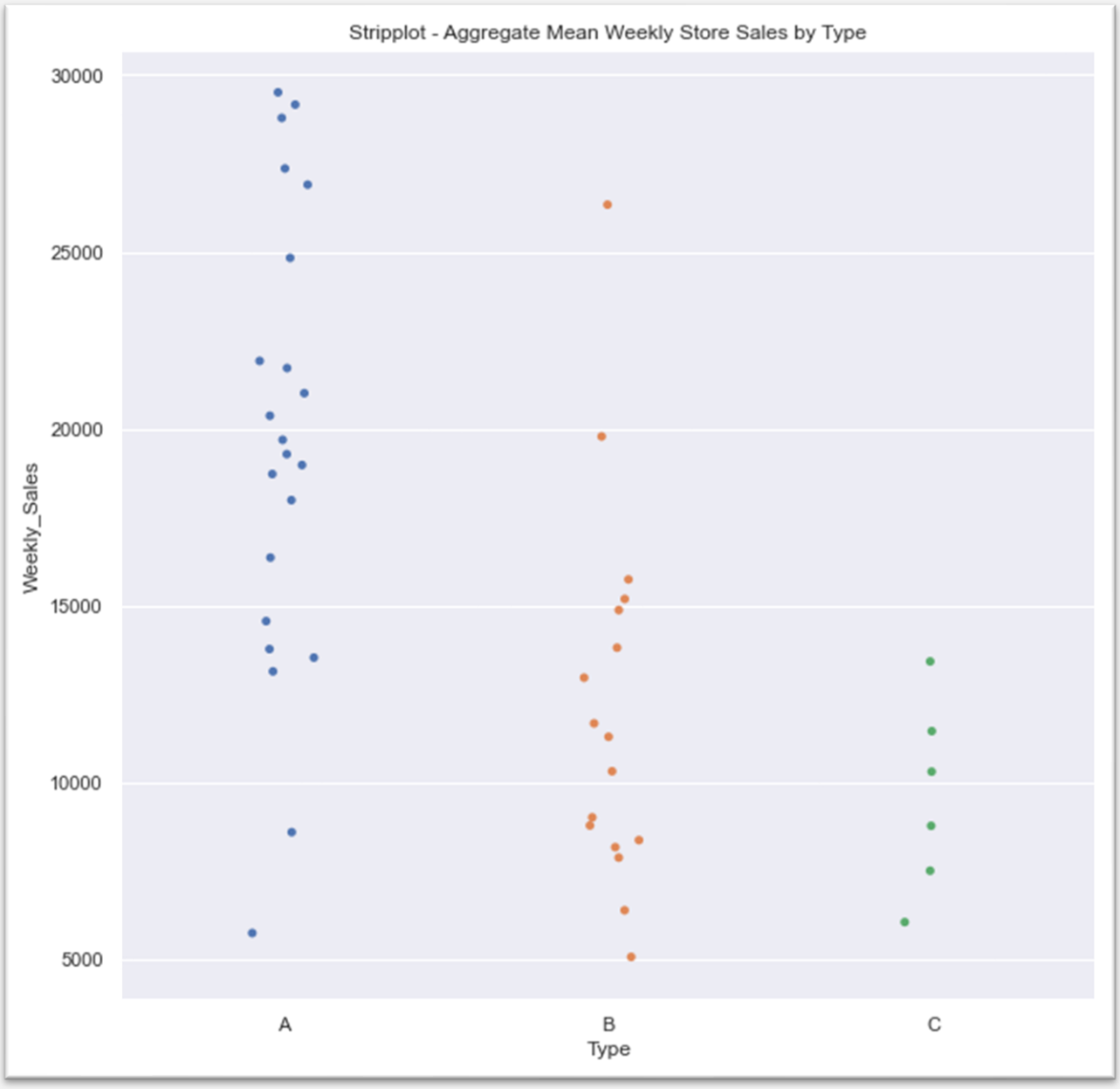

Figure 10: Aggregate Mean Weekly Sales by Store Type – Strip Plot

- There is a relationship between size of store, store type, and weekly sales.

- Line plots of aggregate mean weekly sales by store type were plotted by date as well.

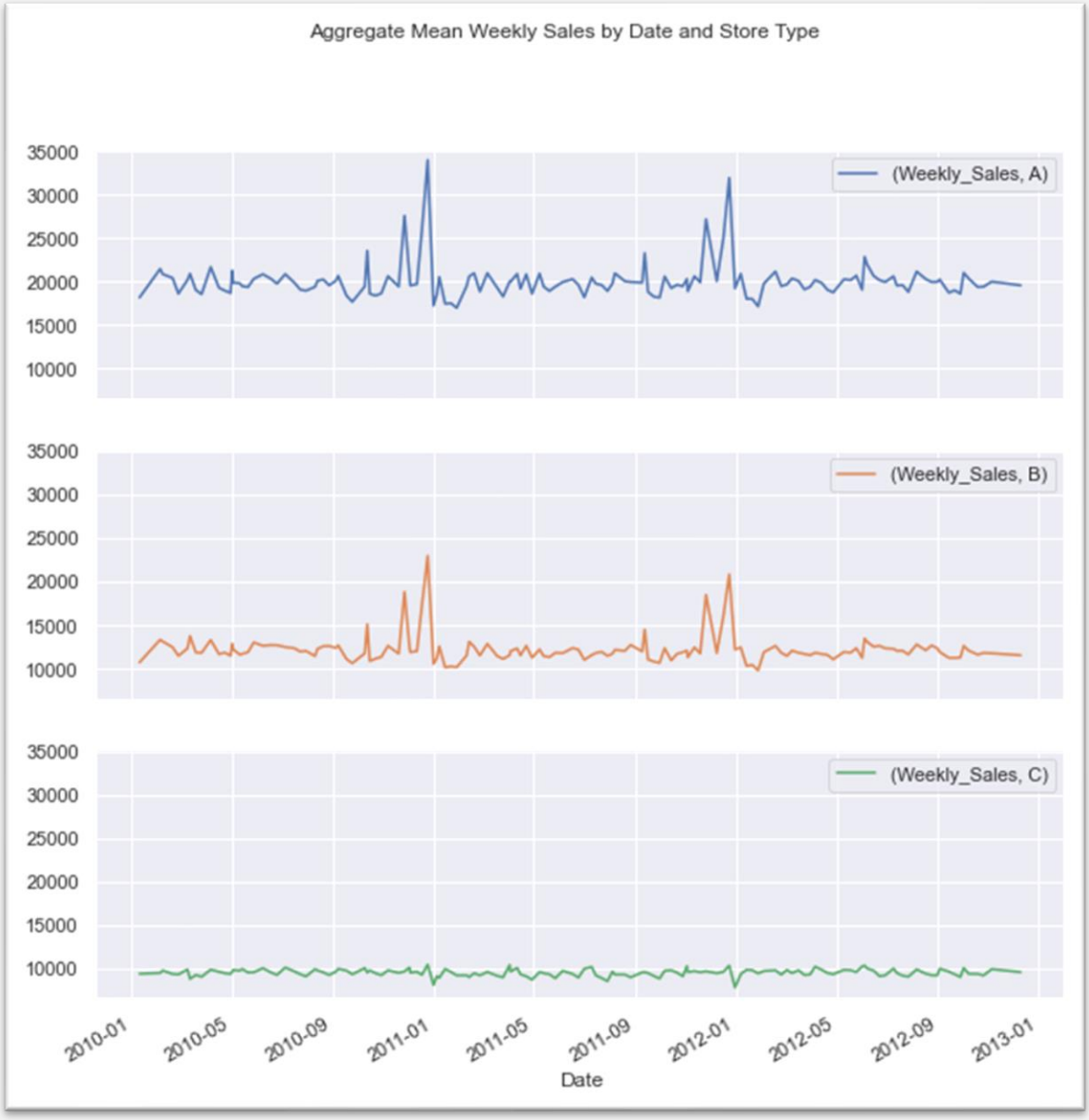

Figure 11: Aggregate Mean Weekly Sales by Store Type – Line Plots

- To get a sense of department sale spread, a distribution plot for departments was made.

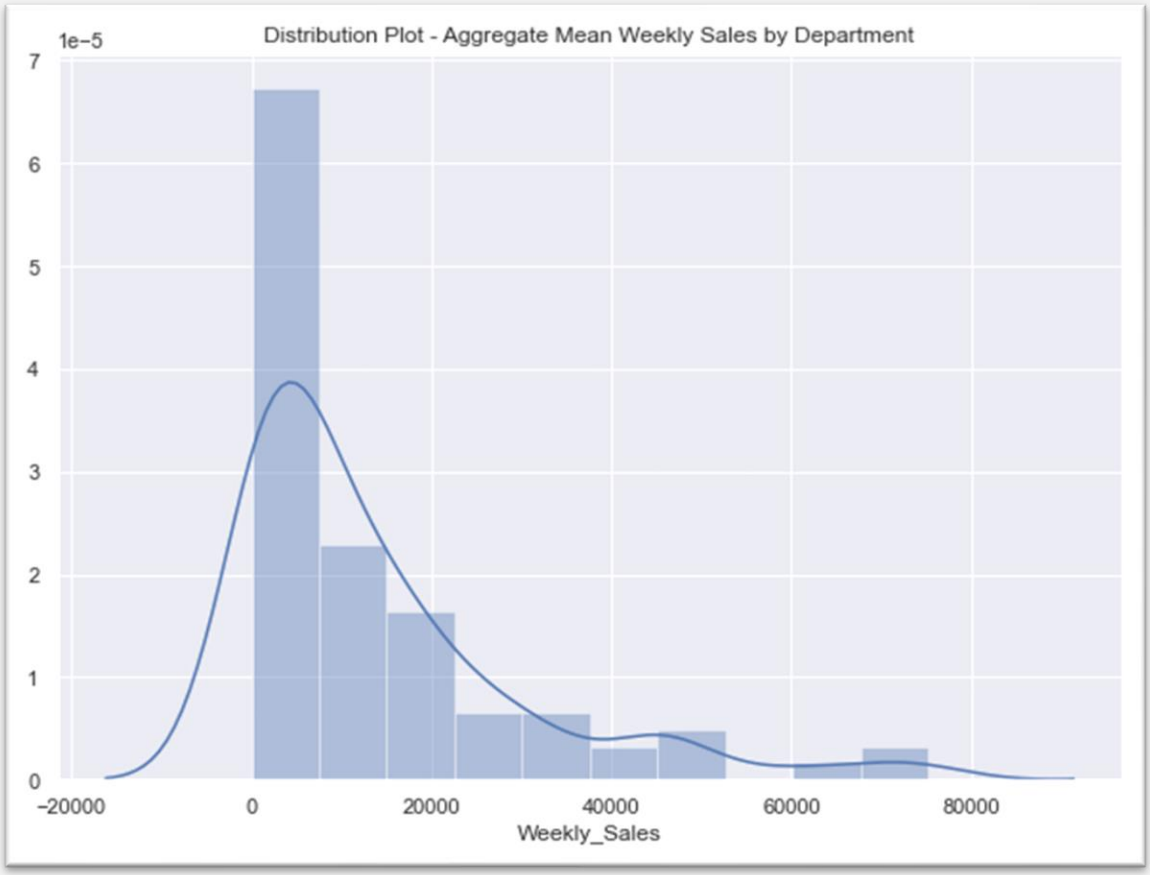

Figure 12: Aggregate Mean Weekly Sales by Department - Distribution Plot

- Most departments company-wide have weekly sales < $20,000. However, there are departments that have upwards to $80,000 in average weekly sales.

- A correlation coefficient heatmap was made for the continuous and Boolean features.

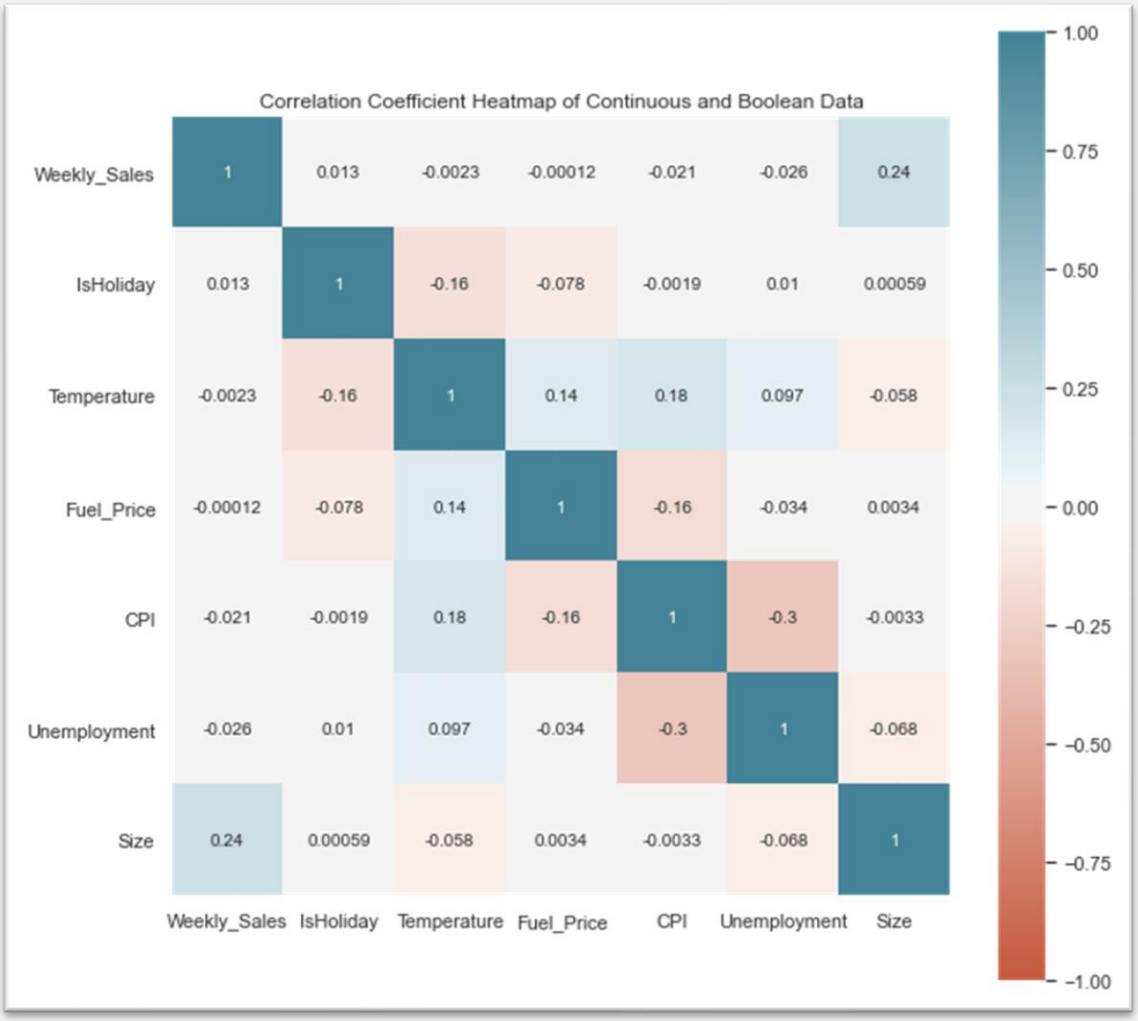

Figure 13: Heatmap of Continuous and Boolean Features

- Store size seems to have the biggest correlation with weekly sales. Weekly sales must primarily depend on store, department, and date. It does seem that temperature, fuel price, consumer price index, and unemployment all negatively correlate with weekly sales, which all could make sense logically.

Step 3 – Pre-Processing:

- Store Type values were converted from letters to numbers A:2, B:1, C:0
- Month and Week features were extracted from the Date feature.
- Markdown columns were dropped
- Dummy features were created for Department, Month, and Week.
- Because Type seemed to have a major impact on the weekly sales, it was decided to split the data out by type and create separate models for each store type.
- Once the data were split by type, dummy features were also created for Store.
- Before dropping the Date feature, a time-series train test split was performed, setting the training data before January 1st, 2012 and the test data after.
- A MinMaxScaler was used to scale the continuous features. The training data were fit to it and then the test data were transformed by it. This was done separately for each data set.
- 5-Fold cross-validation segments were generated from the training data for each store type. This was done manually via custom function. The cross-validation split was done in an appropriate manner for time-series, in which each training set consists only of observations that occurred prior to the test observation that forms the test set. This is called walk-forward validation.

Step 4 – Modeling:

- 3 types of models were trialed for each store type – Linear Regression, Ridge Regression, and Random Forest Regression
- Each model was run using a manually defined cross-validation function.
- Hyper-parameter tuning was performed using model specific functions defined to take parameter range inputs and iterate over the values.
- In addition to using the manually defined Hyper-parameter cross-validation tuning functions, scikit-learn's GridSearchCV and RandomizedSearchCV were also used as a comparison.
- Standard Linear Regression performed very poorly. It overfit to the training data and did not translate to modeling test data.
- Ridge Regression performed much better.
    - CV R^2 test scores maxing out at 0.9061, 0.6825, and 0.8177 for Store Types 0, 1, and 2, respectively.
- Random Forest Regression performed the best but took significantly more resources to fit the data and perform hyper-parameter tuning.
    - CV R^2 test scores maxed out at 0.9809, 0.9009, and 0.9509 for Store Types 0, 1, and 2, respectively.
- In an attempt to improve modeling of Store Type 1 data, trials performing PCA dimensionality reduction prior to Random Forest Regression were performed.
    - The best CV R^2 test score achieved was 0.8874, so PCA was not used in the final models.
- Cross-validation results are seen in the figures below.

Model	R2 Score	RMSE Score
Linear Regression	-8.93e+23	$9.38e+15
Ridge Regression	0.9061	$4,836
Random Forest Regression	0.9809	$2,165

Figure 14: Cross-Validation Results - Store Type 0 (C)

Model	R2 Score	RMSE Score
Linear Regression	-2.56e+24	$2.00e+16
Ridge Regression	0.6825	$9,380
Random Forest Regression	0.9009	$5,049
PCA - Random Forest Regression	0.8778	$5,772

Figure 15: Cross-Validation Results - Store Type 1 (B)

Model	R2 Score	RMSE Score
Linear Regression	-3.10e+24	$3.93e+16
Ridge Regression	0.8177	$11,141
Random Forest Regression	0.9509	$5,623

Figure 16: Cross-Validation Results - Store Type 2 (A)

Final Results:

- The best models were selected for each store type and run on the test data to gauge performance.
- y_test was plotted versus y_prediction for each model to visualize model performance.

Store Type 0

- Model Selected: Random Forest Regressor
- Hyperparameters: max_depth:80, min_samples_split:10, min_samples_leaf:1, max_features:auto
- R^2 Score: 0.9787
- RMSE Score: $2,393

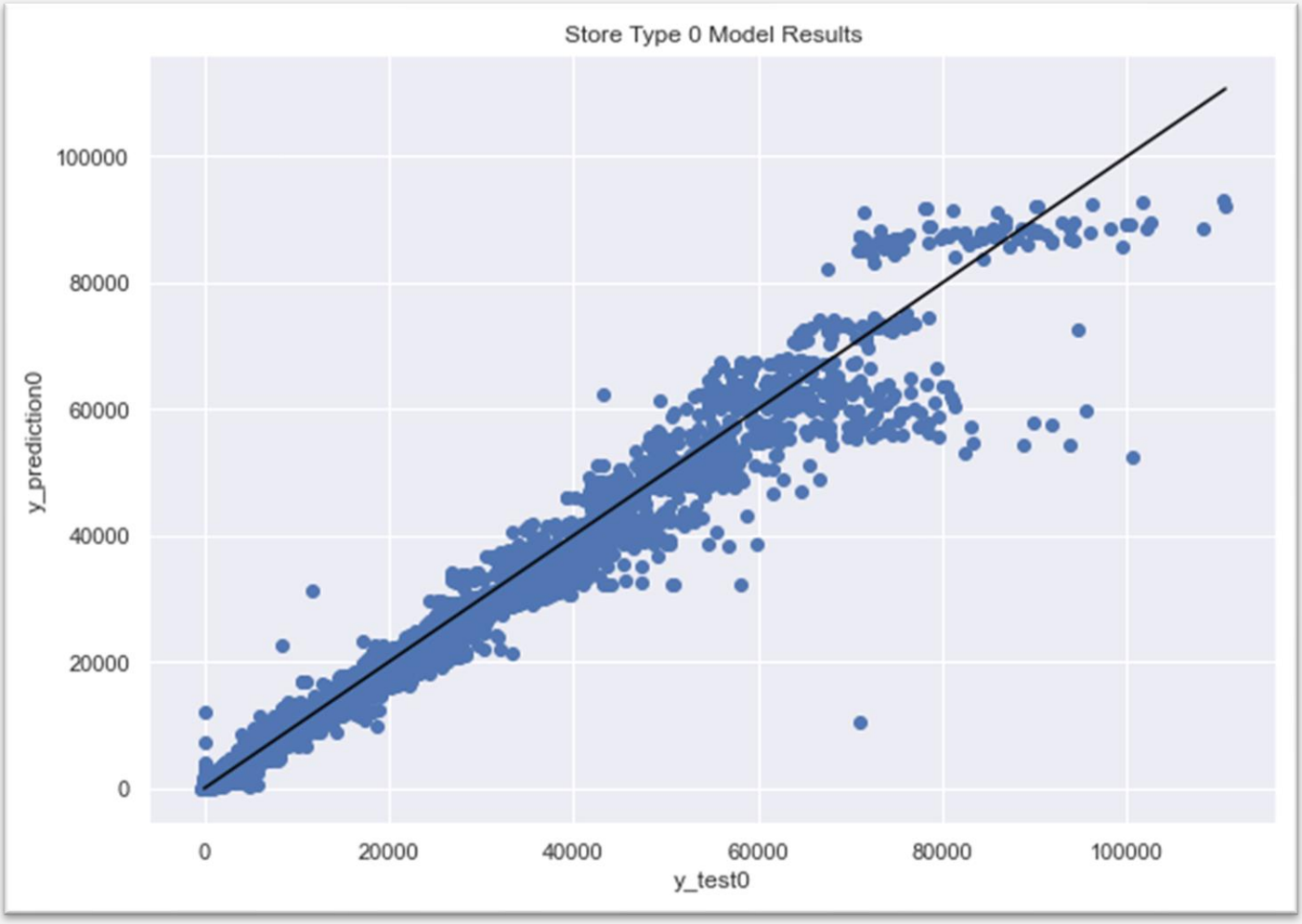

Figure 17: y_test vs y_prediction - Store Type 0

Store Type 1

- Model Selected: Random Forest Regressor
- Hyperparameters: max_depth:80, min_samples_split:35, min_samples_leaf:1, max_features:auto
- R^2 Score: 0.8403
- RMSE Score: $6,329

Figure 18: y_test vs y_prediction - Store Type 1

Store Type 2

- Model Selected: Random Forest Regressor
- Hyperparameters: max_depth:100, min_samples_split:5, min_samples_leaf:1, max_features:auto
- R^2 Score: 0.9370
- RMSE Score: $6,543

Figure 19: y_test vs y_prediction - Store Type 2

Store Type	R2 Score	RMSE Score
0 (C)	0.9787	$2,393
1 (B)	0.8403	$6,329
2 (A)	0.9370	$6,543

Figure 20: Final Metrics