ROSSMAN DATA SALES PREDICTION

Table of Contents

Introduction

This report mainly deals with pre-processing three sets of raw data for a sales forecasting problem across 1,115 drug stores in Germany to enhance the prediction's accuracy and reliability by cleaning and preparing the raw data. The raw datasets include 'store' supplementary information in CSV form with continuing promotion and competitor information and a train and test dataset in excel form with historical sales data with promotion and holiday status. The pre-processing steps include data review, missing data treatment, correlation check, visualization, and key factor identification. The goal is to ensure the reliability of the data and improve the accuracy and consistency of the predicted outcome. The report concludes with a brief model-fitting example to simulate the performance of the prediction and evaluation with the pre-processed data.

1. Methodology - Data Pre-processing & Model Fitting

1.1. Data Review

The initial step in the data pre-processing process is to review the data and structure to understand the features and their types clearly. A detailed examination of the features found in *Figure 1* showed that the 'train' dataset has features such as a store_id, sales, and customer. The historical sales data contained information on sale amount, date, holiday status, and promotional status. The data named 'test' has the same fields as the 'train' dataset but does not include sales and customers.

```
1  train.info()                                    ## No

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   Store          1017209 non-null  int64
 1   DayOfWeek      1017209 non-null  int64
 2   Date           1017209 non-null  object
 3   Sales          1017209 non-null  int64
 4   Customers      1017209 non-null  int64
 5   Open           1017209 non-null  int64
 6   Promo          1017209 non-null  int64
 7   StateHoliday   1017209 non-null  object
 8   SchoolHoliday  1017209 non-null  int64
dtypes: int64(7), object(2)
memory usage: 69.8+ MB
```

```
1  test.info()                                     ## N

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41088 entries, 0 to 41087
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Store          41088 non-null  int64
 1   DayOfWeek      41088 non-null  int64
 2   Date           41088 non-null  object
 3   Sales          0 non-null      float64
 4   Customers      0 non-null      float64
 5   Open           41077 non-null  float64
 6   Promo          41088 non-null  int64
 7   StateHoliday   41088 non-null  object
 8   SchoolHoliday  41088 non-null  int64
dtypes: float64(3), int64(4), object(2)
memory usage: 2.8+ MB
```

Figure 1. Data type and structure of Train and Test datasets

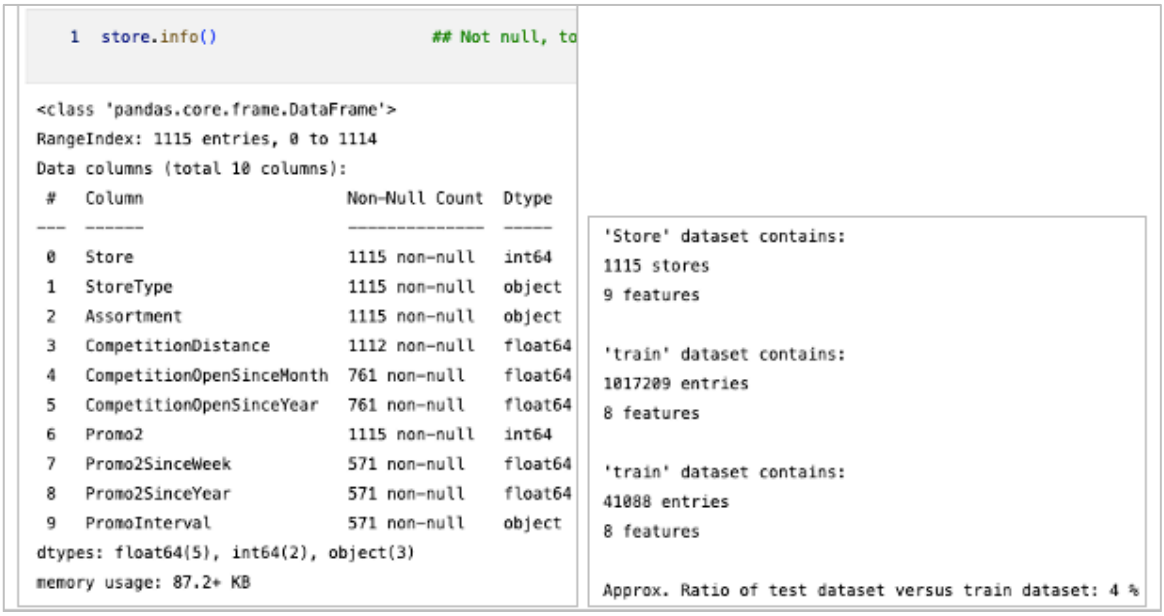

Figure 2. Data type and structure of store dataset

The 'stores' dataset shown in *Figure 2* includes store_id, store type, assortment, competition distance, a competition open date, continuing promotion, Etc. Variables in these datasets are mixed forms of both categorical and quantitative. Following the data structure check, the next step is missing data, and the outlier's treatment, data transformation, and finally, irrelevant instances or features should be filtered out as well.

1.2. Missing Data treatment, Instance reduction & Outliers detection

Missing values, their type, and some meaningless observations are to be checked and excluded from the data to reduce the data's size and improve the analysis's accuracy. The 'train' and 'test' datasets are merged as 'data' dataset to ensure they have the same pre-processing treated features.

It was found that there are NA values in 'sales', 'Customers', and 'Open'. Since the NA values in 'sales' and 'Customers' will be excluded later during the model prediction stage (test dataset), only NA values in 'Open' should be handled.

```
1  data.isna().sum()
✓ 0.3s

Store            0
DayOfWeek        0
Date             0
Sales        41088
Customers    41088
Open            11
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

Figure 3. Missing values in 'data' dataset

Besides NA values, zero values were also checked, and several were found in multiple variables. Zero values in 'sales', 'Customer' and 'Open' mean there are no sales or open history. As the target is predicting sales amount, those instances are not valuable in this case; therefore, these instances have been excluded from the dataset (approx. 17% of observations were excluded). Zero values in 'Promo' and 'SchoolHoliday' remain since zero value means "No" as they are categorical variables.

As a result, 'data2' only stores instances which include the 'Open' feature equal to one.

```
1  #Check 0 values
2  #Empty list
3  zero_c = []
4  zero_p = []
5
6  #Append the list of count and ratio
7  for col in data.columns:
8      zero= data[data[col]==0].shape[0]
9      zero_c.append(zero)
10     zero_p.append(np.round(zero/data.shape[0]*100,2))
11
12 zero_table = pd.DataFrame([zero_c, zero_p], columns=data.columns, index=['Count', 'Percent(%)'])
13 zero_table
✓ 0.3s
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
Count	0.0	0.0	0.0	172871.00	172869.00	178801.0	653953.00	0.0	858354.00
Percent(%)	0.0	0.0	0.0	16.33	16.33	16.9	61.79	0.0	81.11

Figure 4. Zero Value Detection in 'data' dataset

In the 'store' dataset, as zero values are only found in 'Promo2', a categorical variable meaning "No" continuing promotion, no elimination of zero value is required. However,

there are some missing values in some features related to competition and promotion 2. Where date features such as 'Week', 'Month', and 'Year' are present, it can be interpreted as no competition and promotion2 for that store.

```
Total Number of NA values in Store dataset:
 Store                         0
StoreType                      0
Assortment                     0
CompetitionDistance            3
CompetitionOpenSinceMonth    354
CompetitionOpenSinceYear     354
Promo2                         0
Promo2SinceWeek              544
Promo2SinceYear              544
PromoInterval                544
dtype: int64
```

Figure 5. Missing value in 'store' dataset

On the other hand, it was found that 'CompetitionDistance', a numerical variable, had numerous outliers while checking the statistical summary. It provides evidence that the mean would be skewed to the left of the distribution graph while several outliers would be spread to the right side. Outliers and missing values of this feature will be handled during the EDA step in section 2.3.

```
1 store.describe()
✓ 0.1s
```

	Store	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear
count	1115.00000	1112.000000	761.000000	761.000000	1115.000000	571.000000	571.000000
mean	558.00000	5404.901079	7.224704	2008.668857	0.512108	23.595447	2011.763573
std	322.01708	7663.174720	3.212348	6.195983	0.500078	14.141984	1.674935
min	1.00000	20.000000	1.000000	1900.000000	0.000000	1.000000	2009.000000
25%	279.50000	717.500000	4.000000	2006.000000	0.000000	13.000000	2011.000000
50%	558.00000	2325.000000	8.000000	2010.000000	1.000000	22.000000	2012.000000
75%	836.50000	6882.500000	10.000000	2013.000000	1.000000	37.000000	2013.000000
max	1115.00000	75860.000000	12.000000	2015.000000	1.000000	50.000000	2015.000000

Figure 6. Store dataset Statistical summary

1.3. Visualisation (EDA)

Graphs highlighted in *Figure 7* help to understand the distribution in 'data2' dataset, and they identify 'sales' and 'customers' features are skewed to the left while their distribution is assembled. 'Day-Of-Week' shows that they only rarely open on the day 7, a Sunday.

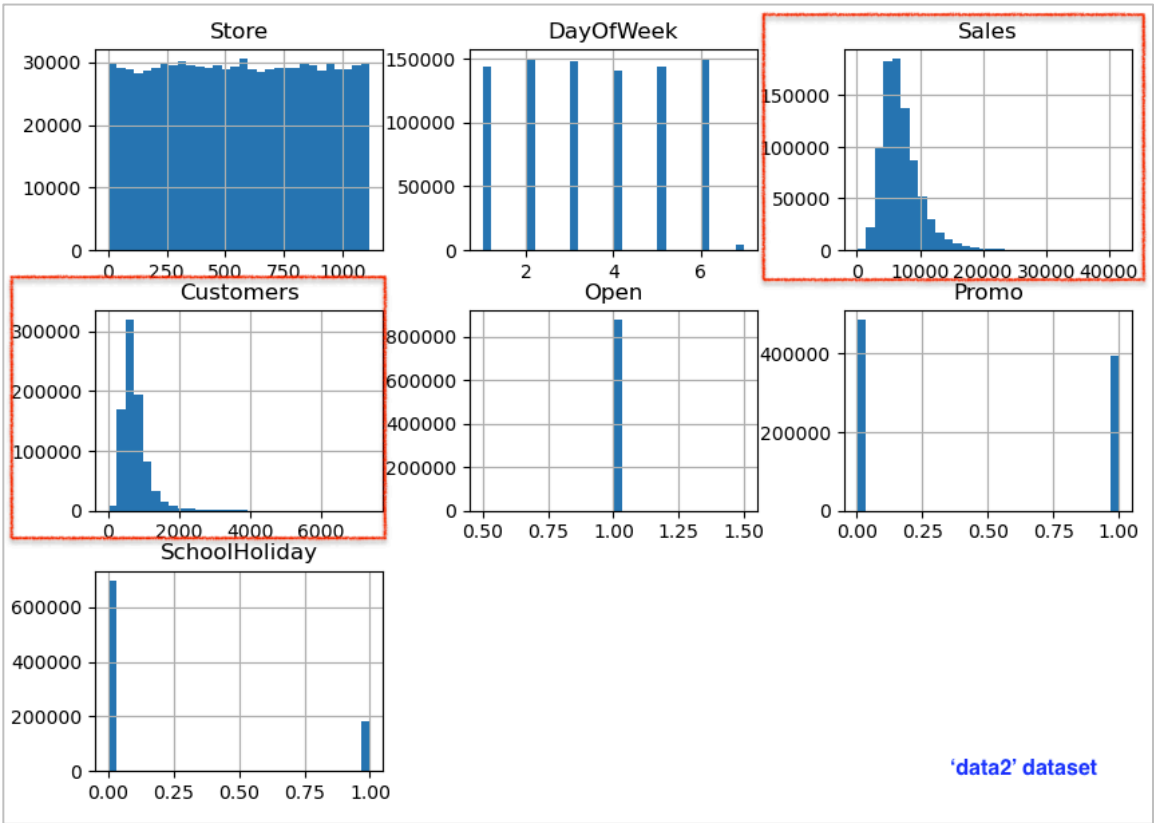

Figure 7. 'data2' (train + test datasets) histogram graphs for numerical variables

As state holiday is not a numerical variable, it was not present on previous histogram graphs. *Figure 8* presents the distribution of sales amount for each state holiday category. The density in category "0" shows a high value, however, "0" means no holiday. For the other holidays, sales amount is mostly distributed below the 23,000 range.

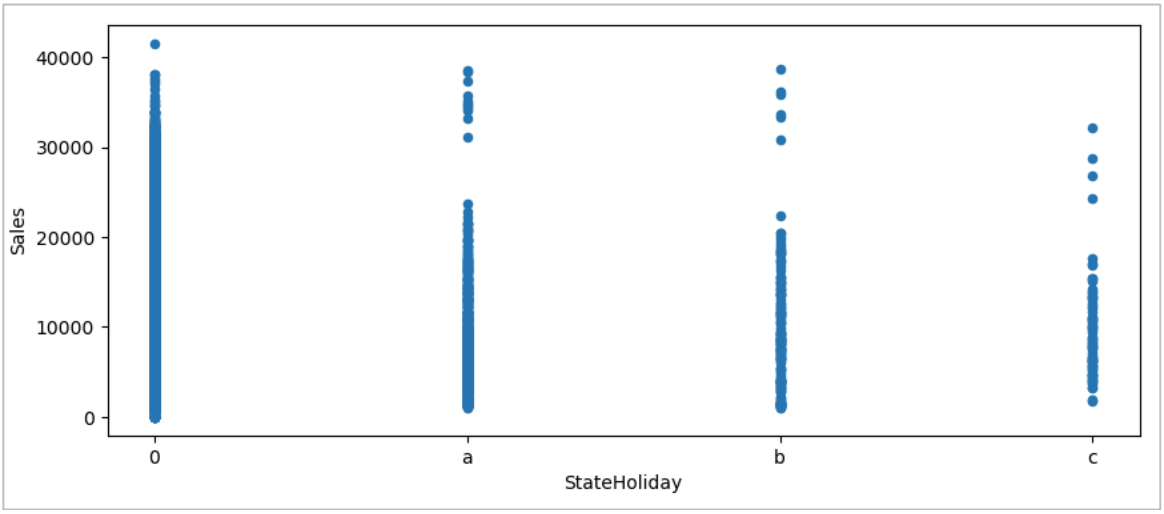

Figure 8. Scatter plot of Sales by StateHoliday

Figure 9 shows that the sales amount over the week is quite regularly distributed, though Wednesday and Saturday have a slightly lower number of sales than other days.

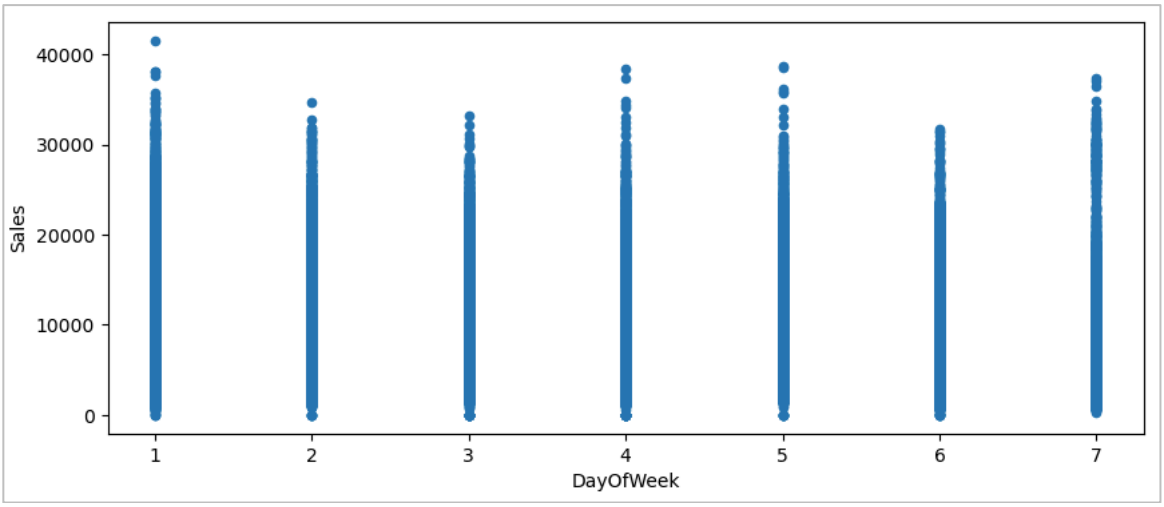

Figure 9. Scatter plot of Sales by DayOfWeek

'Promo' and 'SchoolHoliday' are also categorical variables, though consisting of 0 and 1, which is a binary type. In *Figure 7*, the number of observations of no 'Promo' and no 'SchoolHoliday' appeared higher. In contrast, the empirical cumulative distribution plots *in Figures 10* and *11* show that 'Promo' and 'SchoolHoliday' have greater effects on the number of sales.

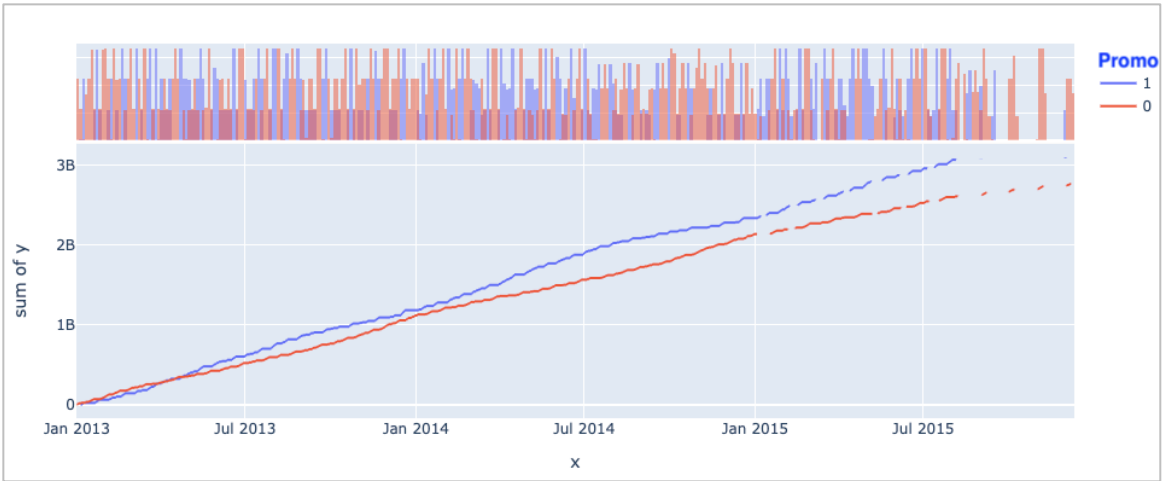

Figure 10. Empirical Cumulative Distribution Plots of Sales by Promo (0: No, 1: Yes)

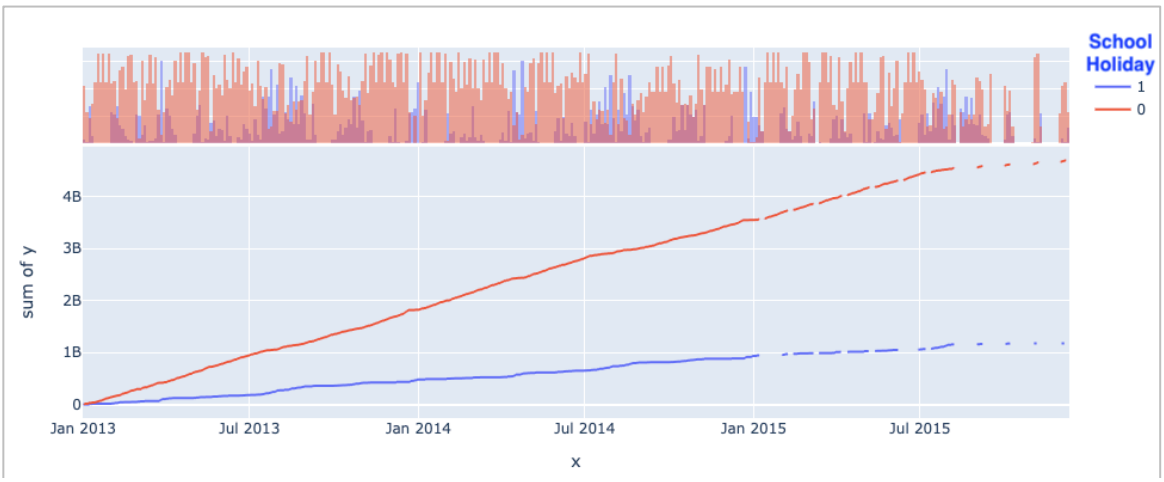

Figure 11. Empirical Cumulative Distribution Plots of Sales by SchoolHoliday (0: No, 1:Yes)

With the same approach, each feature of *Figure 12* was scrutinized and visualized for the store dataset.

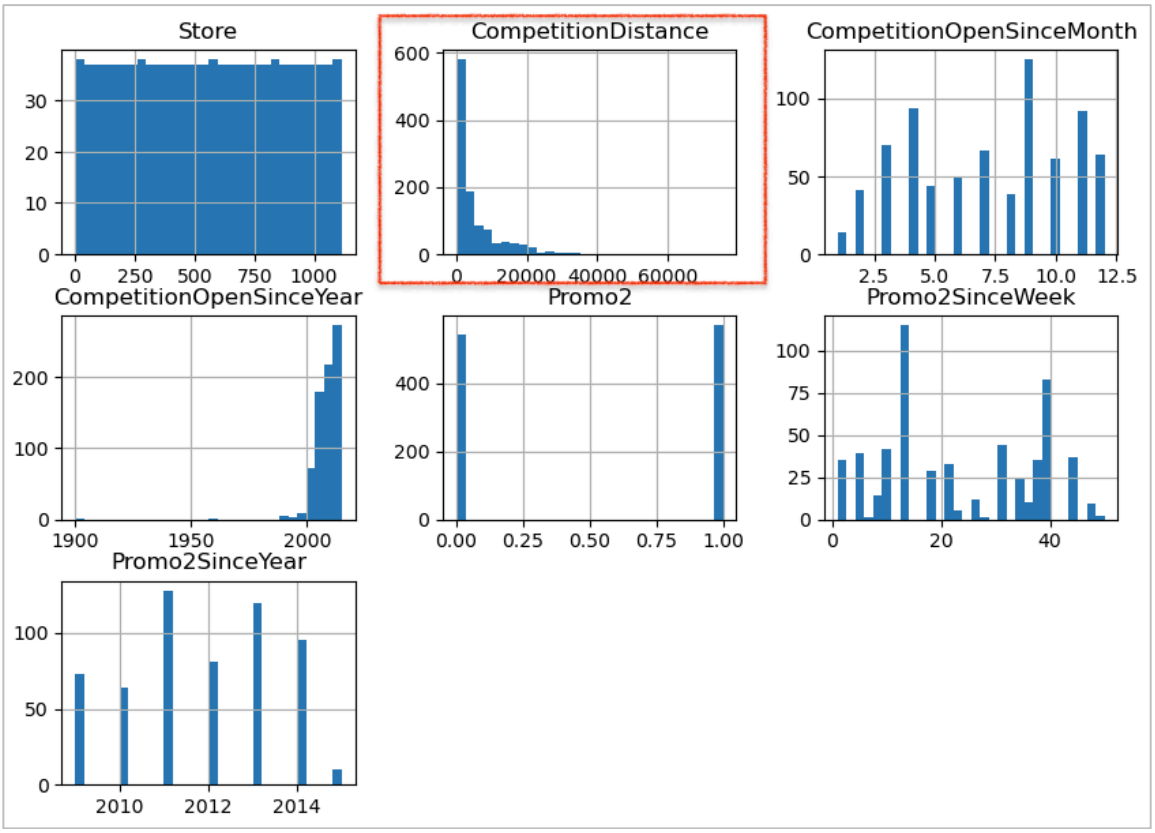

Figure 12. Histogram plots of numerical variables of 'store' dataset

The distinguishable feature in *Figure 12* is "CompetitionDistance", whereby lesser distance indicates closer proximity to city centres and large distance indicates suburb locations. Most stores have competitors within a 20,000 meters distance, but there are some outliers far from the mean, which was identified in section 2.2 of the statistical summary. A boxplot in *Figure 13* illustrates how far outliers are away from the centre of the distribution and where most of 'CompetitionDistance' values are concentrated.

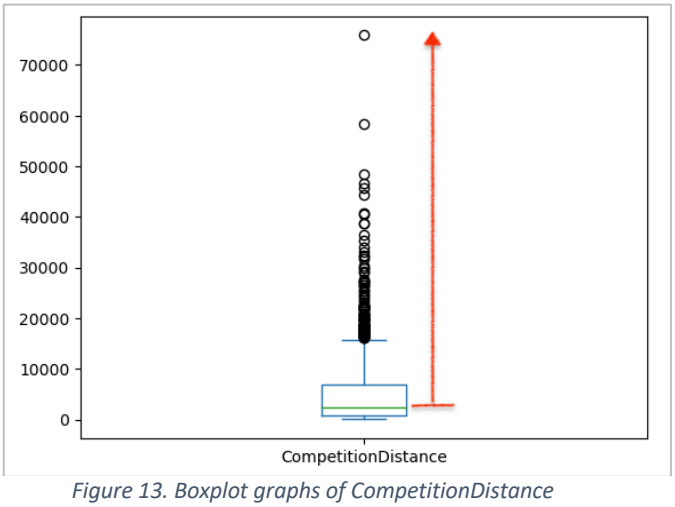

Figure 13. Boxplot graphs of CompetitionDistance

As mentioned, ‘CompetitionDistance’ includes three missing values and outliers, which will be replaced with the median as it shows the left-skewed distribution.

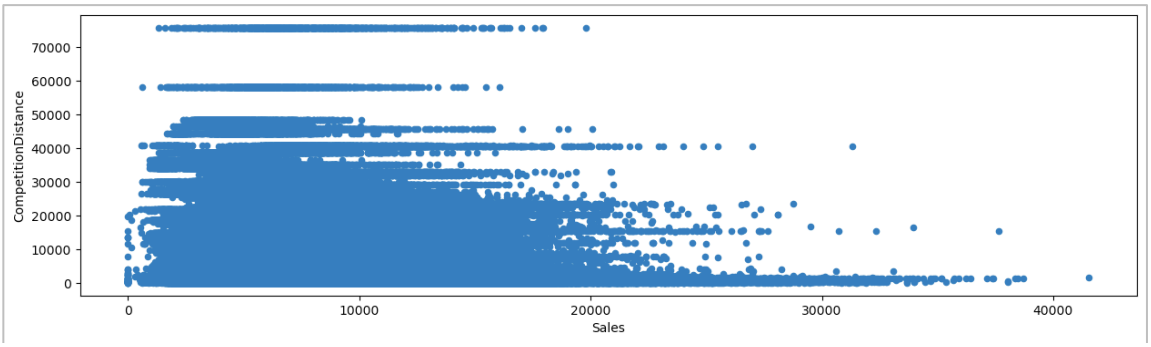

Figure 14. Scatter plot of Sales and CompetitionDistance

The outliers were treated under the following four steps.

Step	Information	Description
1	Most competitors are located within a 5-mile radius	Equivalent to 8,047 meters, and most competitors are located within 10,000 meters of each other (Source: Figure 15)
2	Interquartile range is an important measure to detect outliers	75th percentile of distance is less than 7,000 meters, and 95th percentile is less than 20,400 meters
3	Approximately 53 outliers detected	40 with a ‘CompetitionOpenDate’, 13 without ‘CompetitionOpenDate’
4	‘CompetitionDistance’ outliers replacement	Before replacing, missing ‘CompetitionOpenSinceYear’ and ‘CompetitionOpenSinceMonth’ treated first

Table 1. Four steps of treating CompetitionDistance

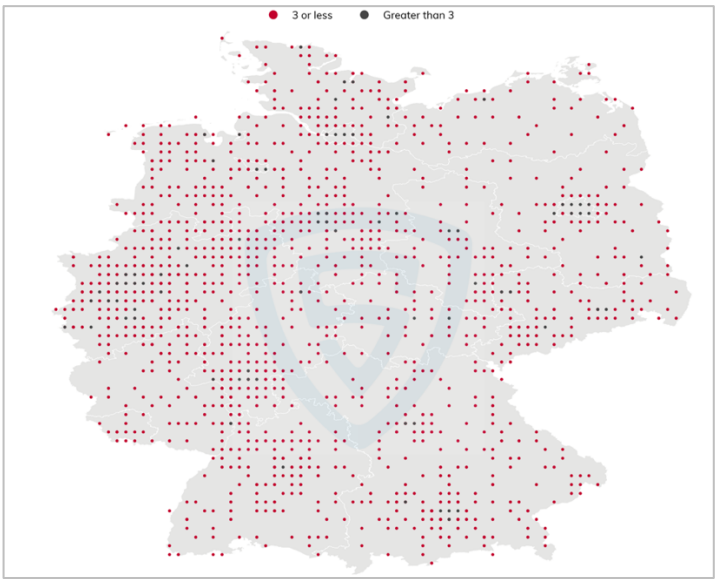

Figure 15. Rossmann locations in Germany in 2023 – Each grid points cover 5 miles radius with at least one location (Reference from ScrapeHero.com)

The Scatter plot and Box plot were used to check the distribution of the missing value in 'CompetitionOpenSinceYear' and 'CompetitionOpenSinceMonth'. Those 354 missing values have been replaced with the mode for year as it is categorical in nature and median for month, respectively. The replacement did not affect the distribution of it. (*Figure 16*)

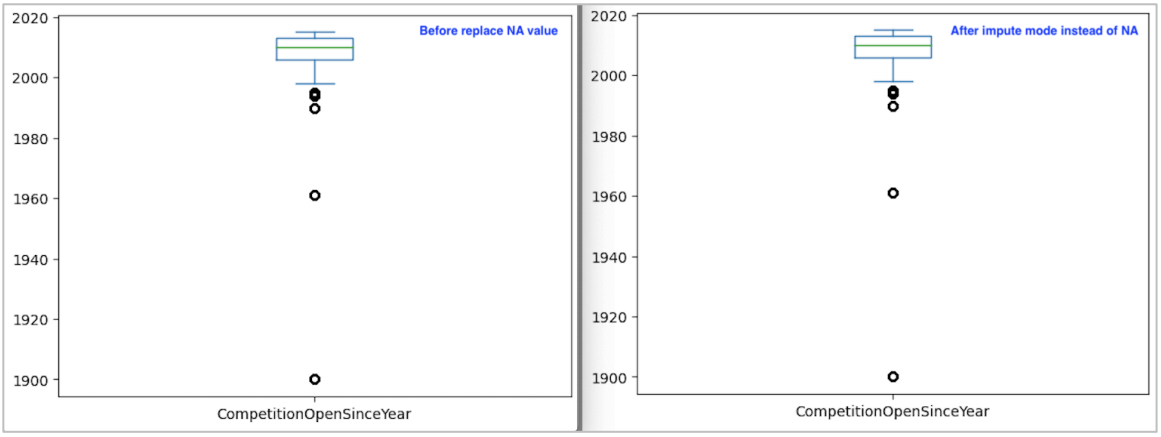

Figure 16. Boxplot before and after missing value imputation

Another issue with this feature is that it includes a couple of outliers, as seen in *Figure 17*. Considering the first Rossmann store opened in 1972, there should only be data after 1972; those smaller than 1972 were also replaced with mode.

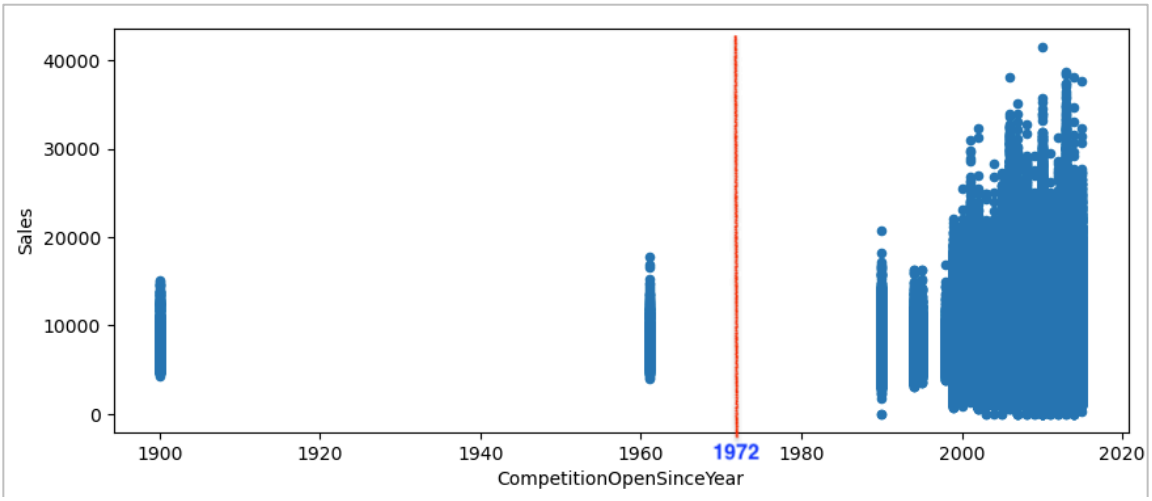

Figure 17. CompetitionSinceOpenYear in Scatter plot (year 1972 is first open year)

After the replacement, the distribution was reformed as per *Figure 18*.

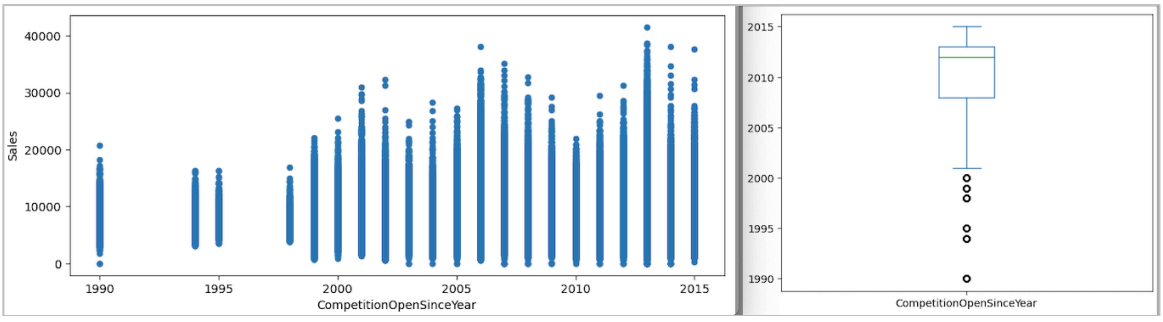

Figure 18. After replacing mode and outliers with mode of CompetitionOpenSinceYear

Following the treatment of CompetitionOpenSinceYear, the competition distance has also been replaced in *Figures 19 and 20*. Compared to *Figures 13* and *14*, the range is within the 95th percentile, but the distribution remains similar to before.

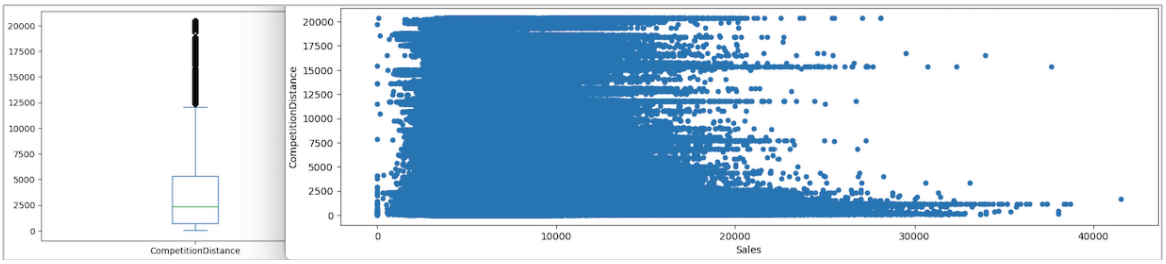

Figure 19. Box plot and scatter plot after the replacement of outliers and missing value of competition distance

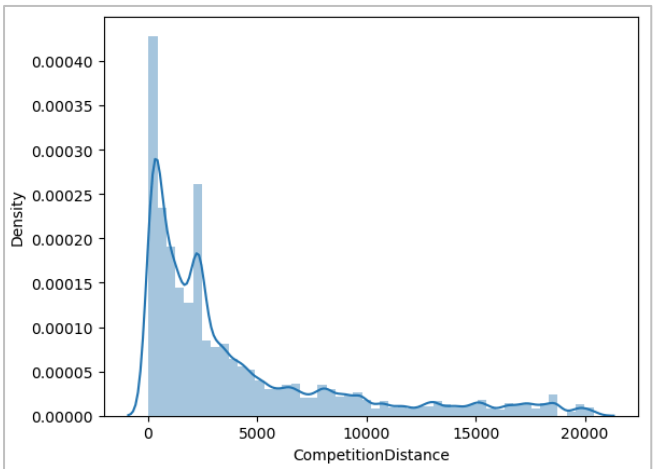

Figure 20. KDE and Histogram of CompetitionDistance feature after missing value and outliers' treatment

With the same approach, as there is no missing value in promo2 features, the missing value in 'Promo2SinceWeek' and 'Promo2SinceYear' also have been replaced with mode. As a result, the overall distribution (*Figure 21*) has been improved compared to *Figures 7* and *12*.

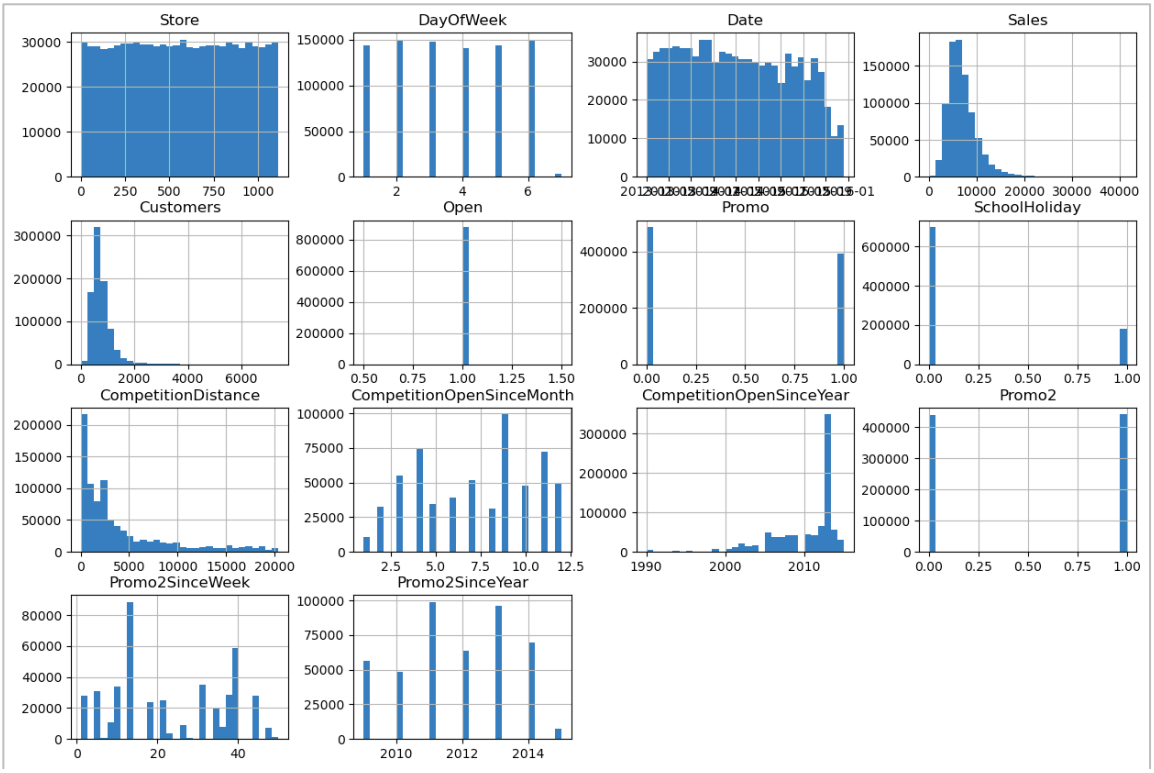

Figure 21. After missing values and outliers' treatment

Figure 22 shows the sales trend for store 1; this can be developed per each different purpose.

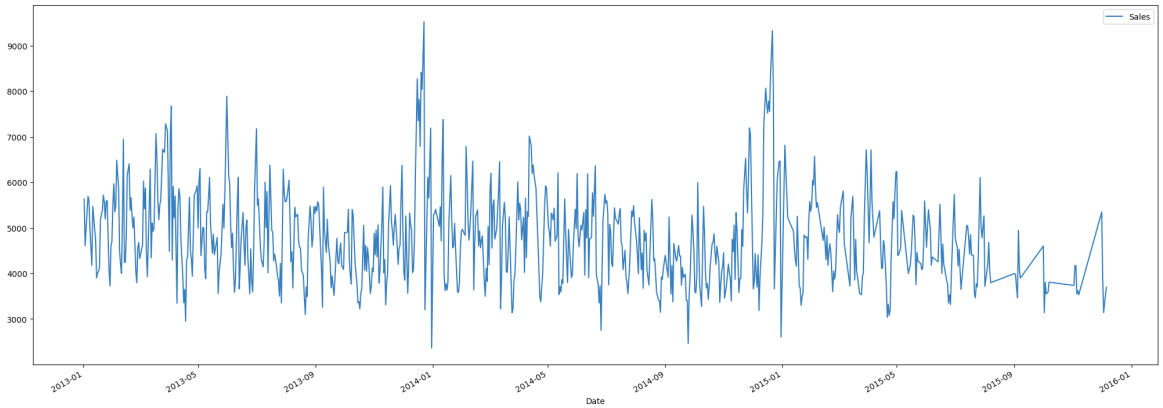

Figure 22. Line graphs of Sales trend

'Promo2Date' and 'CompetitionOpenDate' were added using the existing features to check whether promotion two and competitor existed on the day of sales recorded.

Another set of variables that can be used here is the duration of 'Promo2' and competitor existence, whereby based on the date of the sale and 'CompetitionOpenDate' and 'Promo2Date' which were created above, the duration could be calculated and stored as 'Promo2Open' and 'CompetitionOpen', respectively as numerical variables.

‘CompetitionDist_Cat’ is also added, which divides the ‘CompetitionDistance’ into five ranges. To convert it to a numerical variable, assign the mean of the sales variable in corresponding categories. Target encoding was introduced here to capture the relationship between the categorical variable and the sales (target) variable.
After categorising them, this is again transformed to another numerical variable called ‘Distatnce_num’, which is an average of the sales for each category. As it is related to sales prediction, this will provide a more robust explanation of their correlation. If there is no competition in that period, it is marked as 0.

After the transformation and creation of new variables, the total number of variables which could be used for the prediction turned out to be 35. Only some variables will be used for the model fitting after the correlation check and feature selection step.

```
The Train_Store dataset has 879496 Rows and 35 Variables
```

Figure 23. Number of instances and features after data preprocess

1.4. Correlation Check

Correlation analysis was conducted to identify relationships between the features in the datasets after converting categorical variables to dummy variables (The first column is dropped to avoid multicollinearity). The results reflected moderate to high correlations between certain features, such as sales and customer, and promotional status and sales. *Figure 24* shows that light colours represent a strong positive, and black has a strong negative correlation. (In statistics, less than 0.3 weak, less than 0.5 moderates, and more than 0.5 strong)

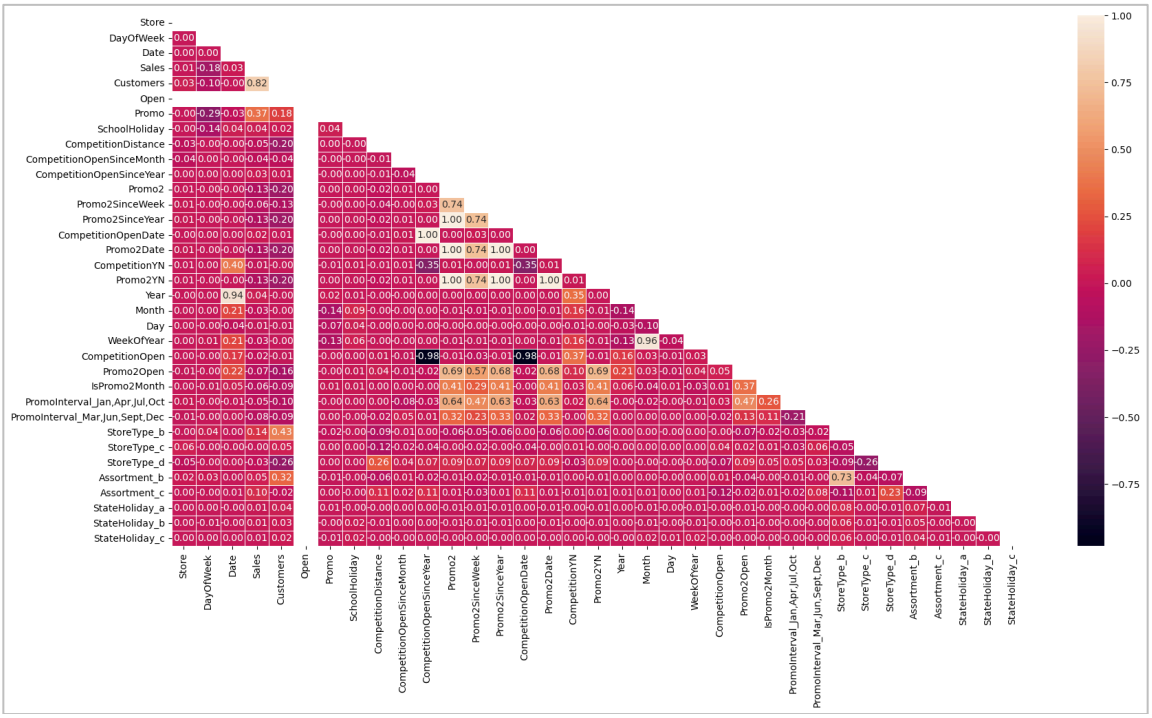

Figure 24. Correlation heatmap of merged dataset

It is important to note that correlation does not imply causation; it is just the strength and direction of the relationship between variables.

In *Figure 25* boxplot, the distribution of each variable against the sales can be presented. These give an idea of sales distribution by each category and their mean and outliers.

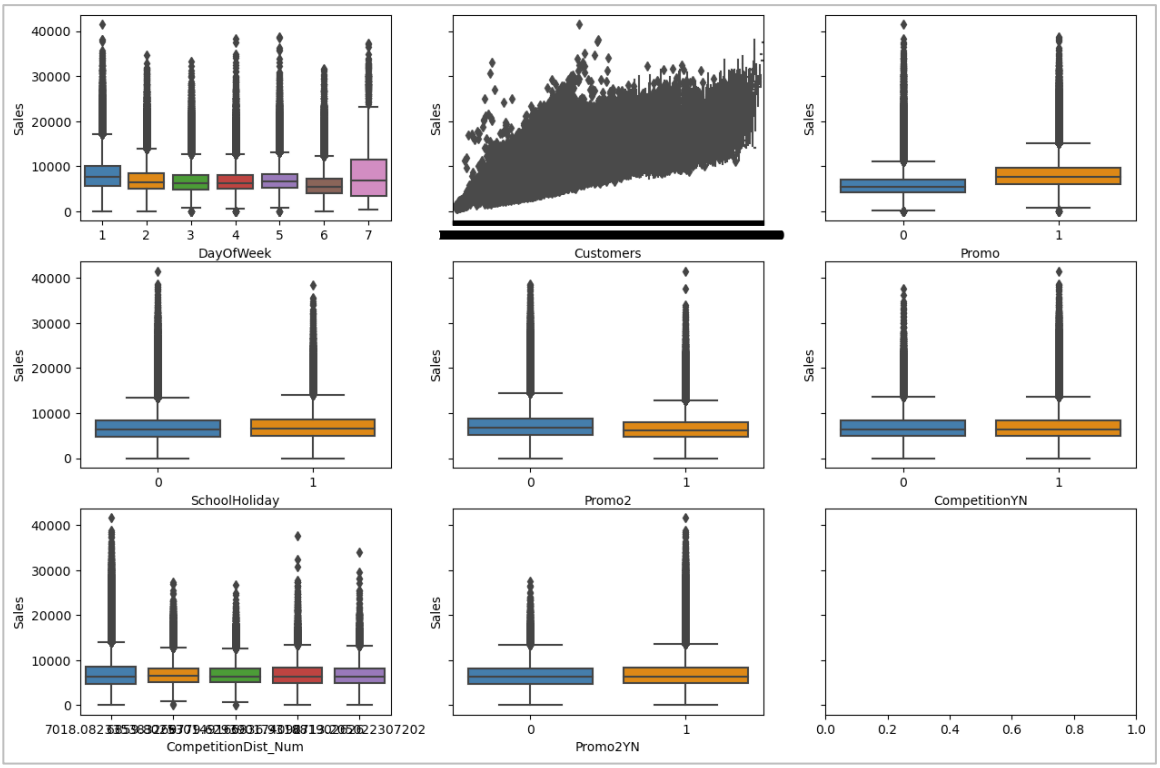

Figure 25. Boxplot (x: categorical variables, y: target variable(sales))

Bar-graphs present the average of sales per category in *Figure 26*. It gives an idea of how much are the average sales of each category.

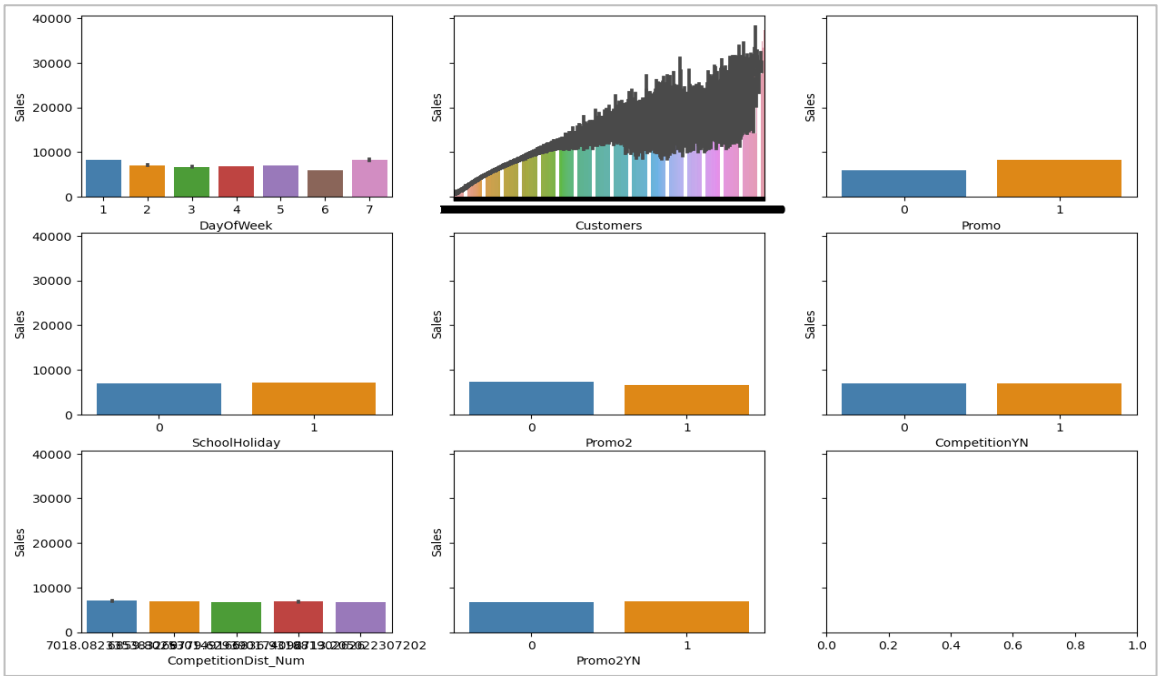

Figure 26. Bar graphs (x: categorical variables, y: target variable(sales))

The heatmap in *Figure 27* shows the correlation among transformed variables. Positive and negative relations are observed, with some variables closely correlated. Light orange indicates a strong positive correlation, and light blue, a strong negative correlation. These observations contributed to feature selection in the next step.

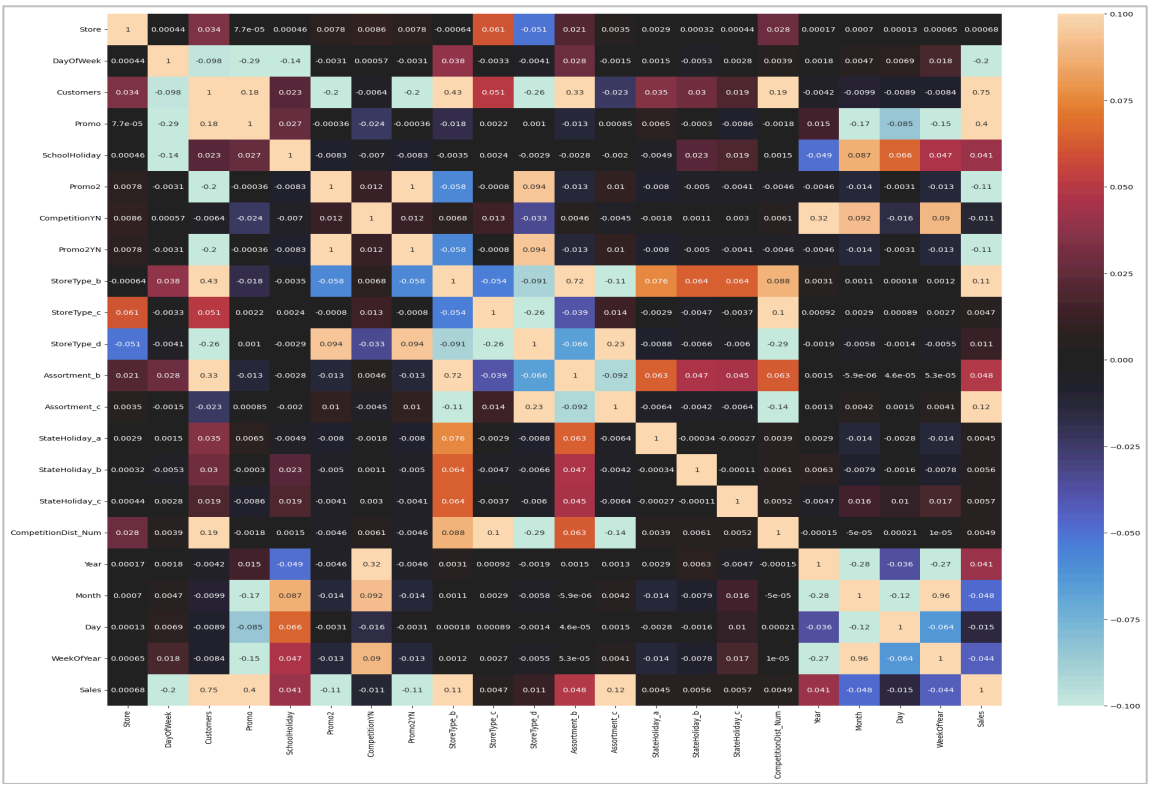

Figure 27. Heatmap with transformed variables

To summarise the pre-processing step, there are new variables which were combined or referred to from the original features in *Table 2*.

Step	Description
1	Train, test, and store datasets were merged to use the same pre-processed data.
2	CompetitionDist_Cat, CompetitionDist_num, CompetitionOpenDate, CompetitionOpen, and CompetitionYN were created using CompetitionDistance, CompetitionOpenSinceMonth, and CompetitionSinceYear. CompetitionDist_num and CompetitionOpen were used in the end as the former has average sales of CompetitionDist_Cat and 0 where CompetitionYN is 0, and the latter indicates the duration of competitor existence.
3	Promo2YN, Promo2Date, and Promo2Open were added using Promo2, Promo2Interval, Promo2SinceWeek, and Promo2SinceYear. Promo2YN reflects the existence of promotion two on the sales date, and Promo2Open indicates the duration of promotion two.
4	Variables with high correlation to each other were omitted, as well as too few correlated variables.
5	Outliers in CompetitionDistance were replaced with median if the competitor existed on the sales date.
6	Month, Day, Year, and WeekOfYear were added based on the date feature in the sales dataset. Month and Year were used as features since they correlate with sales data.

Table 2. Major transformation of Data Preprocess

The Xgboost model's feature importance results were used to determine the most relevant features for prediction. However, there may be alternative feature selection methods, such as statistical tests, correlation analysis, or dimension reduction techniques like principal component analysis (PCA). (This was simulated and attached, as *Figure 34*.)

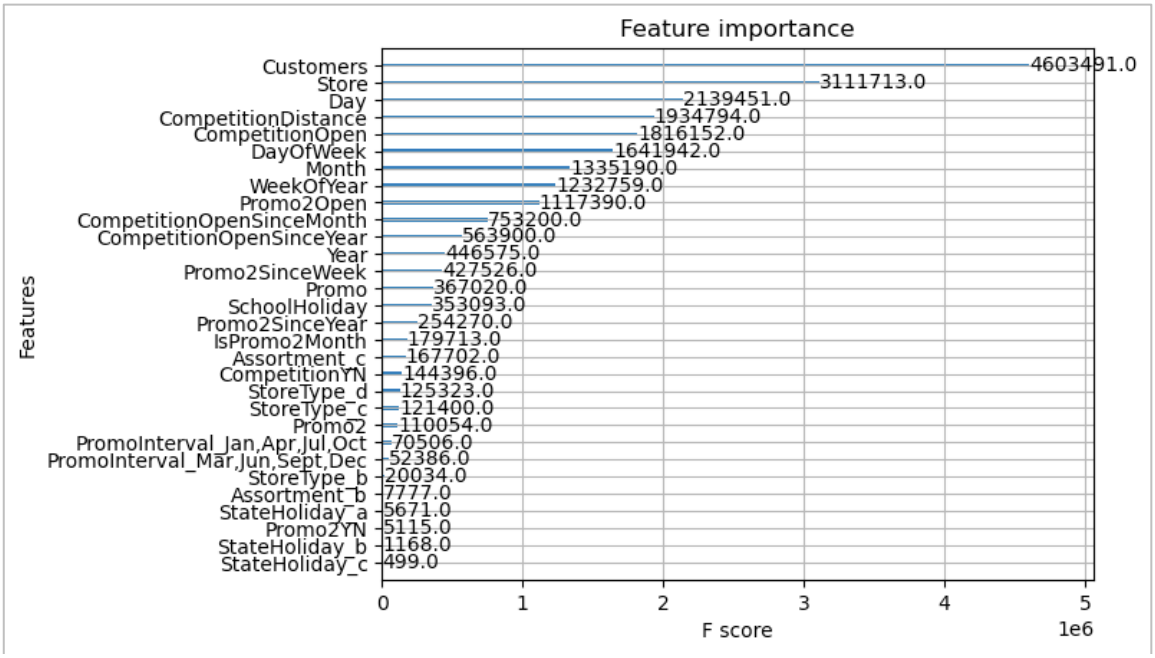

Figure 28. Feature importance plot with first set of variables

As a result, some low-importance features were excluded, such as 'StateHoliday' and 'Promo2YN' (replaceable with other features), and the remaining features, including 'CompetitionDistance', 'CompetitionOpen', 'Promo2Open', 'Month', 'Year', 'Customer', Store, 'DayOfWeek', 'WeekOfYear', 'Promo', 'SchoolHoliday', 'Assortment', and 'StoreType' were used for the final model fitting. (All numerical variables scaled) Various sets of features were used for different models, and the details will be described in section 2.5.

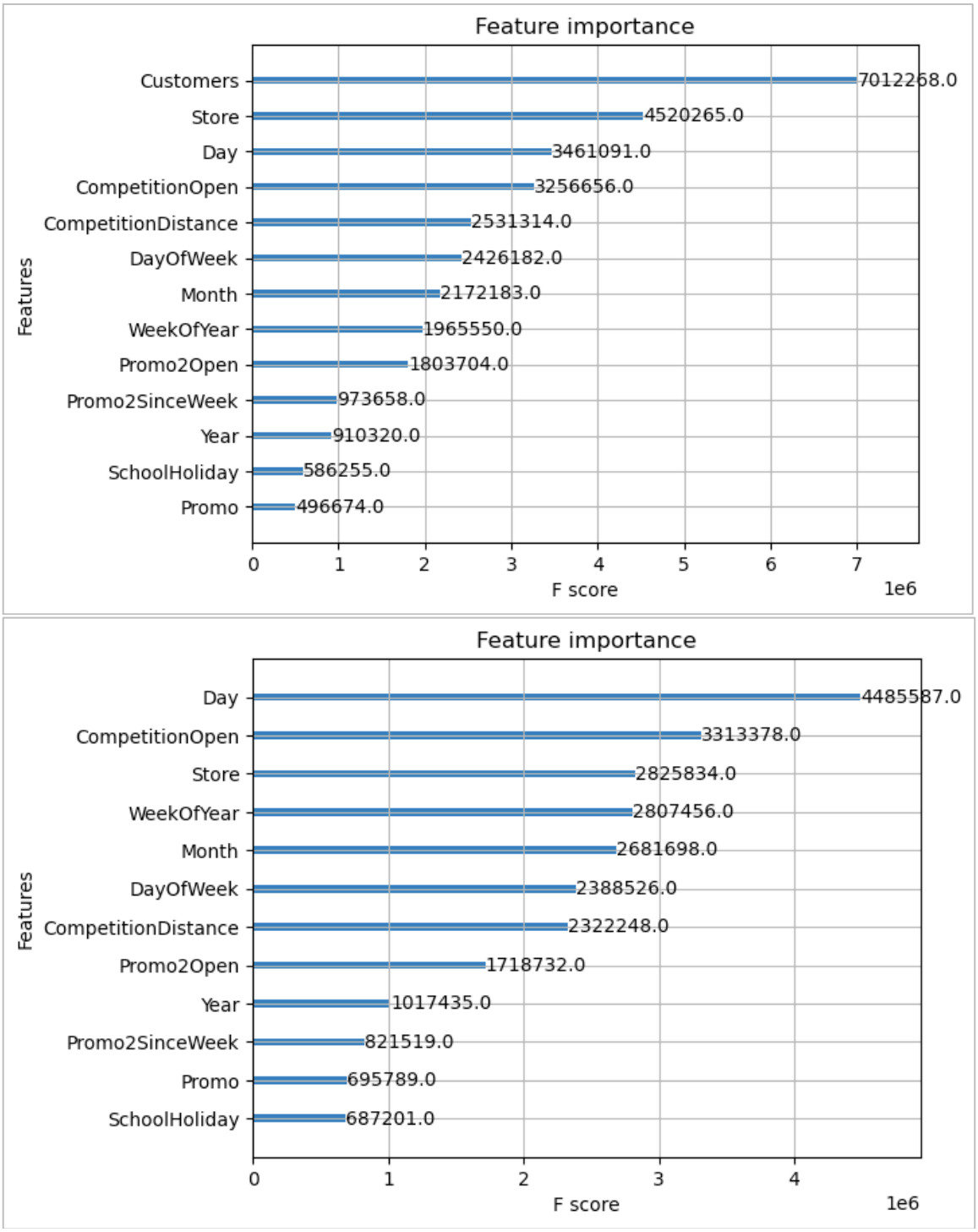

Figure 29. Feature importance with second and third sets of variables
(As Test data doesn't have 'Customers', third set doesn't include 'Customer')

1.5. Model Fitting

In this study, three machine learning models were evaluated for their ability to predict sales data for 1,115 stores. The models included Xgboost (Gradient Boosting Regression), Linear Regression, and MLP regression (Neural Networks). These models were chosen because the target variable had numerical variables, and both categorical and numerical variables existed as input.

The data was divided into two parts, 'traindata' and 'testdata', and further divided into 'X_train_t', 'y_train_t', 'X_test_t', and 'y_test_t', where 'X_test_t' and 'y_test_t' included the sales data of the last six weeks for 1,115 stores, and the remaining was stored in 'X_train_t' and 'y_train_t'..

The Xgboost model showed a decrease in RMSPE and RMSE with increasing iterations, with the final iteration returning relatively low error values. However, this model was influenced by several factors, such as the quality of data, iterations, and hyperparameters.

```
[24]    train-rmse:3.86864      train-rmspe:0.97816     eval-rmse:3.83186       eval-rmspe:0.97723
...
[937]   train-rmse:0.00992      train-rmspe:0.00991     eval-rmse:0.06175       eval-rmspe:0.06076
Training time is 980.296531 s.
```

Figure 30. RMSPE and RMSE over the machine learning iterations

The model's accuracy was higher, with 20 maximum depths. Hyperparameter tuning was also attempted, but the optimal hyperparameter could not be determined due to the many hyperparameters and the high computational cost.

```python
import xgboost as xgb
from sklearn.model_selection import GridSearchCV

# Define the XGBoost regressor
xgb_reg = xgb.XGBRegressor(objective="reg:linear",
                           booster="gbtree",
                           eta=0.03,
                           max_depth=20,
                           subsample=0.9,
                           colsample_bytree=0.7,
                           silent=1,
                           seed=10)

# Define the hyperparameter grid to search over
param_grid = {
    "max_depth": [3, 5, 10, 15, 20],
    "n_estimators": [100, 200, 300, 400, 500],
    "learning_rate": [0.01, 0.03, 0.05, 0.1],
    "min_child_weight": [1, 3, 5, 7],
}

# Create the GridSearchCV object
grid_search = GridSearchCV(xgb_reg, param_grid, cv=5, scoring="neg_mean_squared_error", verbose=1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train_t, y_train_t)

# Print the best hyperparameters and the corresponding mean squared error
print("Best hyperparameters: ", grid_search.best_params_)
print("Best mean squared error: ", grid_search.best_score_)

1197m 10.6s
```

Figure 31. Hyperparameter tuning for Xgboost

The Simple Linear Regression model showed a moderately acceptable accuracy score with an RMSE result of 0.26. The MLP model's score was low initially but improved with optimisation, with an RMSE result of 0.46. The results of the two models differed depending on the features, but this was due to the hyperparameter setting. The computational time of the code to compare the accuracy of multiple models was too long, more than two days, making it inappropriate for this large dataset with high-cost machine-learning models.

```
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.neural_network import MLPRegressor

from sklearn.model_selection import GridSearchCV,RepeatedStratifiedKFold
from sklearn.metrics import explained_variance_score
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

models=[('LR',LinearRegression()),('DT',DecisionTreeRegressor()),('RF',RandomForestRegressor()),('XGB',XGBRegressor())
        ,('SVR',SVR()), ('MLP', MLPRegressor(hidden_layer_sizes=10, random_state=42))]
scores=[]

for name,model in models:
    model.fit(X_train_t3,y_train_t)
    preds=model.predict(X_test_t3)
    score=explained_variance_score(preds,y_test_t)
    mae = mean_absolute_error(y_test_t, preds)
    r2 = r2_score(y_test_t, preds)
    scores.append([name,model,mae,r2,score])
```

917m 59.3s

Figure 32. Multiple Model Fitting

The evaluation measures used in the study included RMSPE for time series forecasting and RMSE for other models. The MAE, R2, and Score for the three models are shown in *Figure 33*.

Name	MAE	R2	Score	Name	MAE	R2	Score
XGB	0.055707	0.971354	0.970473	XGB4	0.048410	0.977009	0.976107
LR	0.189020	0.640288	0.403977	LR4	0.188418	0.638872	0.397390
MLP	0.383880	-0.090554	0.394796	MLP4	0.253143	0.406254	0.394258

Figure 33. Error Evaluation table for three models

In this study, however, cross-validation and hyperparameter tuning were not performed, but they would have played a crucial role in improving the results. The appropriate model would depend on the characteristics of the data, the complexity of the relationship between input and output variables, and the desired accuracy of predictions. Model performance on the training set is not an indicator of its performance on the test set, which is why cross-validation is crucial in choosing the best model. To achieve the best generalization performance, it is important to cross-validate the models and perform hyperparameter tuning.

2. Conclusion

In this report, the focus was on time-series forecasting, and the result was achieved by using Xgboost with feature importance ranking. The success of this approach was due to the effective pre-processing of the data. Data pre-processing included various steps such as missing value imputation, outlier treatment, data transformation, data reduction, and the creation of additional features by linking existing features in a meaningful way. Understanding the data and determining the exact target of the analysis was key to this process. If identifying features was not an issue, in a sales prediction scenario, unsupervised machine learning techniques like PCA could be applied to reduce the dimension of the dataset, as *Figure 34*.

```python
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression

# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)
X_pca_train = pca.fit_transform(X_train)

# Perform KMeans clustering
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(X_pca_train)

# Plot the data with different colors for each cluster
plt.scatter(X_pca_train[:, 0], X_pca_train[:, 1], c=clusters)
plt.show()

# Fit a separate linear regression model for each cluster
regressors = []
for i in np.unique(clusters):
    reg = LinearRegression()
    reg.fit(X_train[clusters==i], y_train[clusters==i])
    regressors.append(reg)

# Use the regression model to make predictions on the test set
X_pca_test = pca.transform(X_test)
clusters_test = kmeans.predict(X_pca_test)
y_pred = np.zeros(y_test.shape)
for i, reg in enumerate(regressors):
    y_pred[clusters_test==i] = reg.predict(X_test[clusters_test==i])
```

```
Mean Squared Error: 0.054981041247207915
```

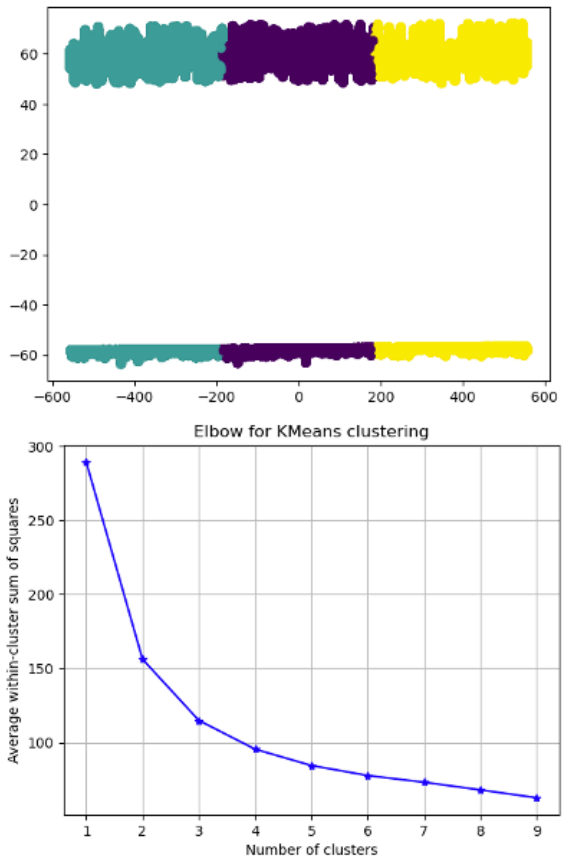

Figure 34. PCA + Clustering (Unsupervised Technique) + XGB (Supervised Technique)

The revised plan in this report differed from the initial plan in terms of feature selection and the model used. The initial plan was limited due to insufficient research on the features and difficulties in connecting different datasets. This led to the use of only a limited number of features and a shallow exploratory data analysis. The comparison between the two plans highlights the importance of thoroughly understanding the structure, data type, and information of the datasets for accurate predictions with the right features.

Data pre-processing was challenging, as it required knowledge of the features and various types of missing values. The large size of the data also made the model fitting and validation process time-consuming. However, by trying out various approaches to variable handling in pre-process step, the results of the model validation process showed that a well-prepared and clean dataset is crucial for accurate machine learning predictions. The use of high-ability models, while computationally expensive, is worth the investment in terms of accuracy.

In conclusion, data pre-processing is an essential step in ensuring reliable and consistent data for machine learning. A thorough understanding of the data and proper machine learning conditions, such as hyperparameter settings, is critical for improving the accuracy of sales forecasting. Further analysis and modelling can be conducted to refine and enhance the predictions. (2493 words)

3. Reference

1. Pandas Document. (2023) https://pandas.pydata.org/docs/

2. Seaborn Document (2022) https://seaborn.pydata.org/api.html

3. xgboost document (2022) https://xgboost.readthedocs.io/en/latest/index.html

4. StatsTest.com (2023) https://www.statstest.com

5. Salvador G. et al. (2015) Data Preprocessing in Data Mining

6. Trevor H. et al. (2001) The elements of Statistical Learning

7. Simon Ro., Mark G. (2011) A first course in Machine Learning