A PROJECT REPORT ON

Forecasting sales using the store, promotion, and competitor data

A project report submitted in fulfillment for the Diploma Degree in AI & ML

Under

Applied Roots with University of Hyderabad

Project submitted by

Nitish Kulkarni

Under the Guidance of

Mentor: Aniket Vishnu

University of Hyderabad

ACKNOWLEDGEMENT

I would like to express deepest gratitude to my Mentor: Aniket Vishnu, for his valuable suggestions, noble guidance, support, and encouragement throughout this project.

Abstract:

Sales forecasting refers to the process of predicting the sales performance of a company over a specific period in the future. Rossmann, which operates over 3,000 drug stores in 7 European countries has currently tasked its managers to predict their stores daily sales for up to six weeks in advance. The goal of this project is to enable managers to forecast these sales. In this project, we have applied Machine learning techniques to predict the Rossmann store daily sales.

Contents

1. Problem definition

1.1 Problem Background

Sales forecasting refers to the process of predicting the sales performance of a company over a specific period in the future.

Sales forecasting of products over specific set of time in future is important, as the demand for products keeps changing from time to time and it is crucial for business firms to predict customer demands to offer the right products at the right time.

1.2 Importance of Sales forecasting

1. Sales forecasting helps businesses to make better informed business decisions and aid in overall business planning and risk management.

2. Sales forecasting allows these businesses to efficiently allocate resources for future growth and manage their cash flow.

3. Sales forecasts also helps to achieve the sales goals by identifying if there are any early warning signals in their sales forecast and change their strategies wherever necessary.

4. Sales forecasting helps businesses to estimate their costs and revenue and maintain adequate inventory of products and thus, improving the financial performance.

Predicting sales performance is one of the main challenges faced by every business as there are several factors that influence the forecast. Some of major factors can be listed as following:

1. **Competition in the market** - There will be several competitors in the market that are involved in selling the same products/services with their own set of promotional strategies, campaigns etc. This can alter the market share of the existing firms significantly and must be considered while forecasting sales.

2. **Demographics of consumers** such as their income, education, age, gender etc. heavily influence the sales of any products/services and must be considered while forecasting sales.

3. **Economic factors** like inflation, recession, national income, disposable personal income, etc. have significant impact on sales of any products and this must be thorough considered while forecasting sales.

4. **Sales strategies** like running promotional or advertising campaign potentially can influence sales and the impact of running these campaigns should be thoroughly considered while forecasting sales.

1.3 Problem Statement

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied

2. Literature survey and Data Acquisition

After extensive literature survey, forecasting can be classified as being broadly based on Qualitative and Quantitative techniques.

2.1 Qualitative/Intuitive Forecasting Techniques

These methods depend on gut feeling of sales representatives and sales managers about the forecast of sales, and these methods heavily rely on the experience and smartness of the sales managers and representatives.

These methods are usually not reliable as the forecast can vary unrealistically from one manager to another.

These methods are usually relied upon when we do not have enough past data and when the environment is too unpredictable.

Some of the commonly used Qualitative forecasting methods can be summarized as follows:

Sales force composite Method

The Sales Force Composite Method involves the sales agents forecasting the sales for their respective territories, which is then consolidated at regional levels, and then the aggregate of all these factors is consolidated to develop an overall company sales forecast.

The sales force composite method is the bottom-up approach wherein the sales agents give their opinion on sales trend to the top management.

Jury Method

The Jury Method also known as an Executive Opinion Method is a sales forecasting method that involves the executives from different departments coming together and forecasting sales for the given period based on their experience and specialization

The jury method is the Top-down approach as it is based on the judgments of the top executives coming together and giving their opinions on sales trend.

User/Customer Survey Method

This method of forecasting involves the direct survey of the consumers and based on customers' expectations of their needs and requirements , sales is forecast.

Delphi method

This method involves gathering the opinions of the panel of experts on the problem being encountered through questionnaires usually sent through mail.

Here, a change agent aggregates all the anonymous opinions received through the questionnaires and based on the initial responses, the second questionnaire is prepared and is again sent to the same group of experts. This process is continued until the responses are narrowed and consensus on sales is reached.

2.2 Quantitative Forecasting Techniques

Quantitative forecasting methods are data-driven objective processes that rely on sales data collected from the past to provide forecasts.

Some of the commonly used Quantitative forecasting methods can be

summarized as follows:

Straight-Line/Historical Growth Method

This is a simple forecast method that involves calculating future sales by factoring any simple growth factor into the equation. Straight-line forecasting also called the historical growth rate gives a rough idea of where sales will be based on past growth rate.(For example – If we consider monthly sales growth as 10% and last month's sales was 100$,then forecast would be 110$).

Run Rate Method

This is another simple method which considers the averages of past sales data and forecast the sales (For example – If the sales in last 10 months was 1000$, then the forecast for next 2 months would be 200$)

Moving Average Forecast Method

In this method, the sales are forecast by taking the rolling averages of the last several values of the time series. When using this method to forecast sales, the period used in predicting sales moves forward depending on the timeframe that we are dealing with. The sales data from several consecutive past periods is combined to provide a reasonable sales forecast over the future period.(For example - If the average sales in last 3 periods is 100 $, then the next period will be in that range)

Weighted Moving Average Method

This sales forecast method takes into account that the more recent data is probably more accurate than older data and to account for that, while forecasting sales ,a numerical weight is assigned to the more recent data to increase their impact on the projection.(For example – if data from past 50 months is used while forecasting sales, more weight would be placed on recent months sales data than for older data to account for change in market conditions over time)

Time Series Analysis Method

In this sales forecasting method, the data of several past years are chronologically ordered and then analyzed to break data into seasonal variation patterns, Cyclical Patterns, trends, and growth components and then use this to project sales.

Regression Method

This method involves studying the functional relation of many factors that influence sales with sales which is their dependent variable. Then, the relationship between the dependent variable (sales forecast) and the causal variables is used in regression to forecast the sales.

Regression equation could be equation demonstrated as :

Y = (w1*x1)+(w2*x2)+….+(wn*xn)+b

Here ,

Y represents the sales,
x1,x2 ….xn represent the causal factors
w1,w2….wn are the constants that show the extent to which these factors influence the sales.

Advanced Machine Learning Methods

Due to recent advances in computing technologies and access to vast amounts of data, advanced machine learning techniques can be used to analyze the past sales data and predict future sales. The objective of these

ML forecasting methods is to improve the accuracy of sales forecasts while minimizing a loss function.

The most significant difference between these machine learning methods and older traditional methods like linear regression lies in the manner of error minimization. While most traditional methods utilize explainable linear processes, most advanced machine learning methods use nonlinear techniques to minimize the loss functions.

2.3 Dataset

Data is taken from Kaggle from the following link:

https://www.kaggle.com/competitions/rossmann-store-sales/data

Dataset properties

Dataset is approximately of size 40 MB and consists of the following four files:

train.csv - historical data including Sales

test.csv - historical data excluding Sales

sample_submission.csv - a sample submission file in the correct format

store.csv - supplemental information about the stores

File	**Variables**	**Number of variables**
train.csv	store, day of week, date, sales, customers, open, promo, state holiday, school holiday	9
test.csv	id, store, dayofweek, date, open, promo, state holiday, school holiday	8
store.csv	store, storetype, assortment, competition distance, competition open since month, promo2, promo2since week, promo2since year, promo interval	10

3. Exploratory Data Analysis

Exploratory Data Analysis is a very important step in a data science project cycle as it helps to understand the input data and identify patterns and correlations in data. EDA is also helpful while handling missing data, encoding features, and to engineer new features if necessary and to understand the predictions of the machine learning model.

train.csv file contains a total of 1017209 observations.

There are 9 columns in total and sales is the target variable.

Store	a unique Id for each store
DayOfWeek	indicates the day of the week
Date	data of the sales
Sales	sales/day, this is the target variable
Customers	number of customers on a given day
Open	Boolean to say whether a store is open or closed (0 = closed, 1 = open)
Promo	describes if store is running a promo on that day or not
StateHoliday	indicate which state holiday (a = public holiday, b = Easter holiday, c = Christmas, 0 = None)
SchoolHoliday	indicates if the (Store, Date) was affected by the closure of public schools

Initial insights :

=> Sales amount per day = 5773 Euros

=> Minimum sales per day = 0

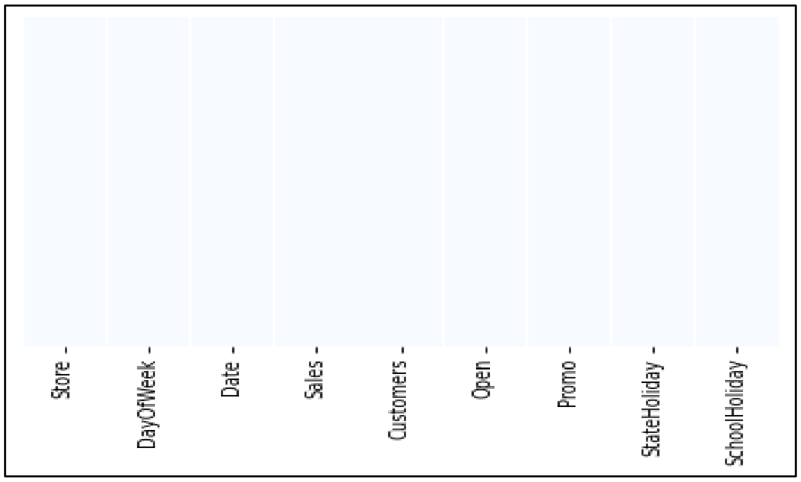
=> Maximum sales per day = 41551

=> Average number of customers = 633

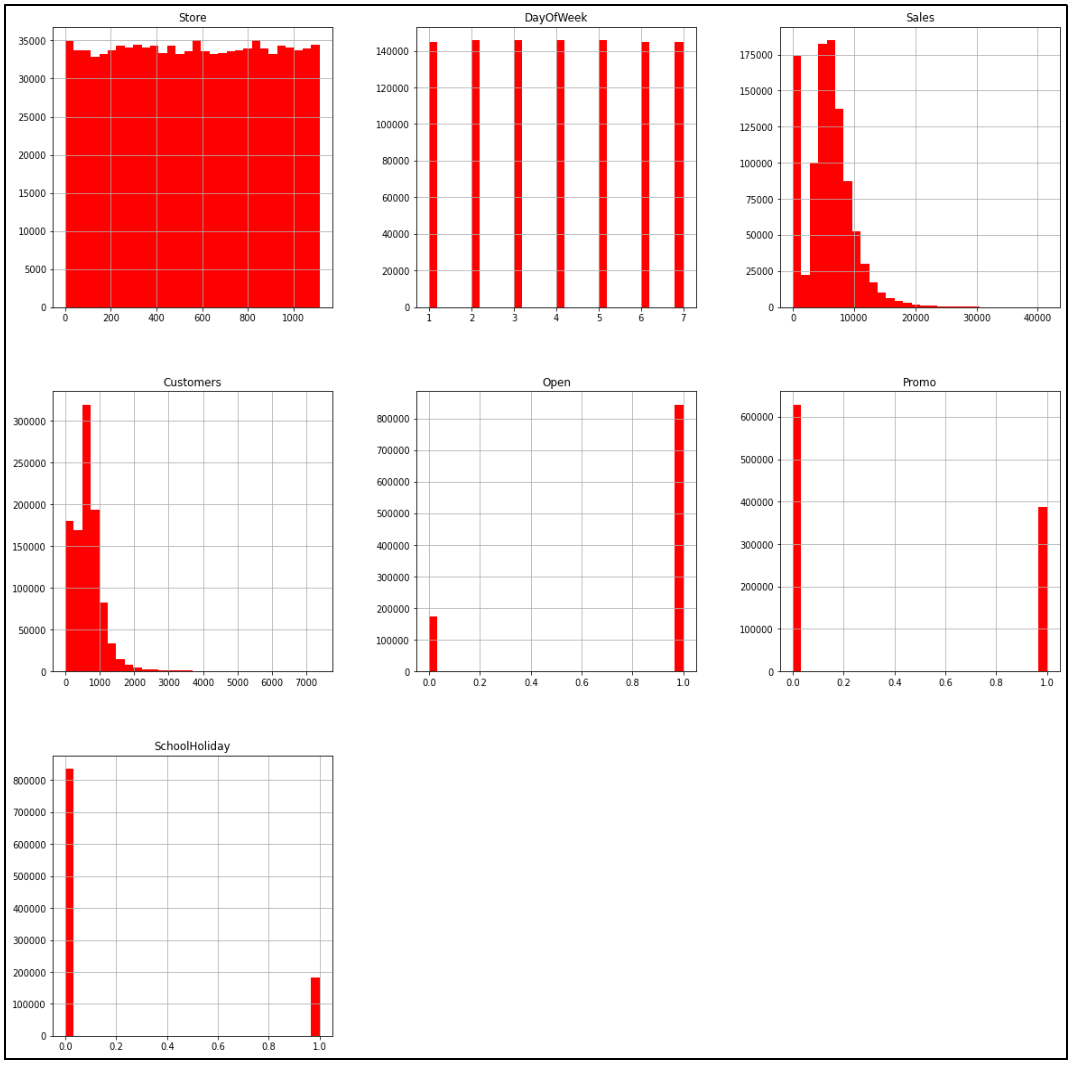
=> Minimum number of customers = 0

=> Maximum number of customers = 7388

The file contains no missing data.



Also, a histogram of the dataset reveals the following insights:



=>Data is equally distributed among all stores

=>Data is equally distributed across various Days of the week

=>Stores are open approximately 80% of the time

=>Promo1 was running approximately 40% of the time

=>Average sales are around 5000-6000 Euros

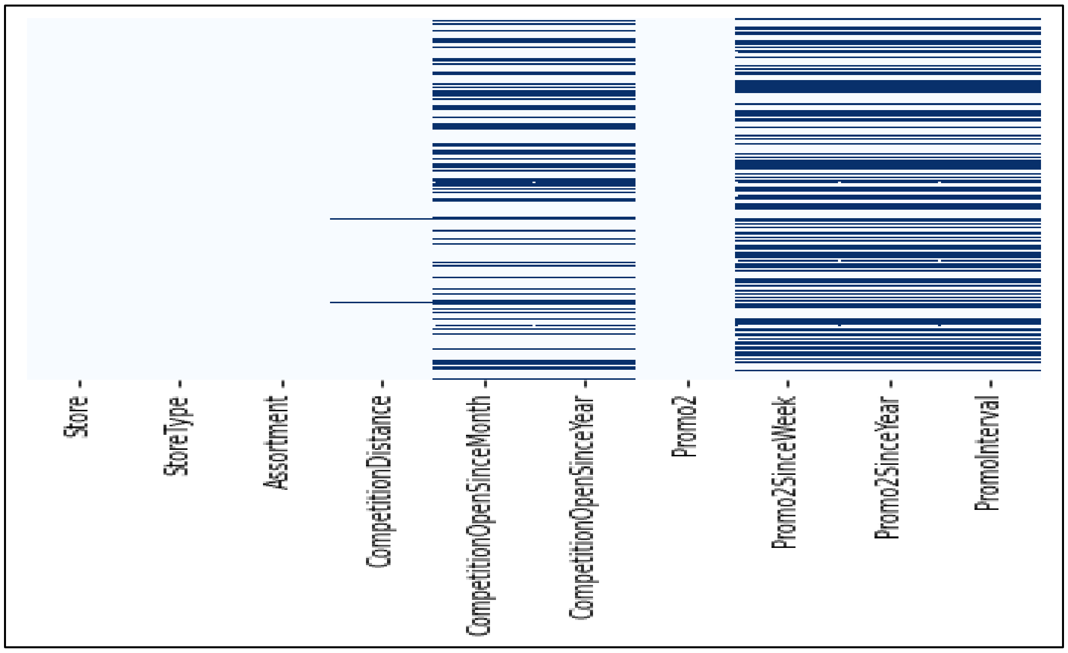
=>School holidays are around 18% of the time

Store.csv file only includes information about the unique 1115 stores that are part of this project and does not contain information about transaction sales per day.

There are 10 features as shown below :

Store	a unique Id for each store
StoreType	categorical variable to indicate type of store (a, b, c, d)
Assortment	describes an assortment level: a = basic, b = extra, c = extended
CompetitionDistance	distance to closest competitor store in meters
CompetitionOpenSinceMonth	provides an estimate of the Month since when competition was open
CompetitionOpenSinceYear	provides an estimate of the Year since when competition was open
Promo2	Promo2 is a continuing and consecutive promotion for some stores (0 = store is not participating, 1 = store is participating)
Promo2SinceYear	Year since when the store started participating in Promo2
Promo2SinceWeek	Week since when the store started participating in Promo2
PromoInterval	describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

This file contains missing data.

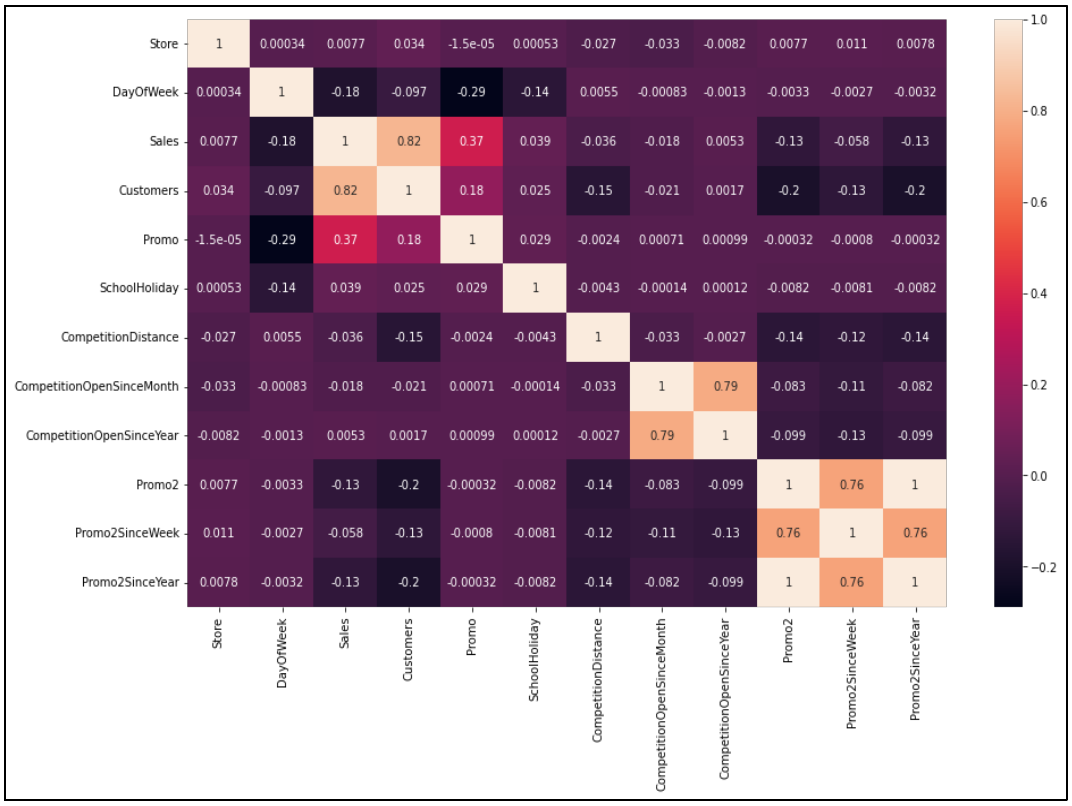


=>'CompetitionDistance' feature has 3 rows missing which are filled with the average values of the 'CompetitionDistance' column.

=>'CompetitionOpenSinceMonth' and 'CompetitionOpenSinceYear' features have 354 rows missing and these are filled with zeros.

=>If 'promo2' feature is zero, values in features 'promo2SinceWeek', 'Promo2SinceYear', and 'PromoInterval' are set to zero.

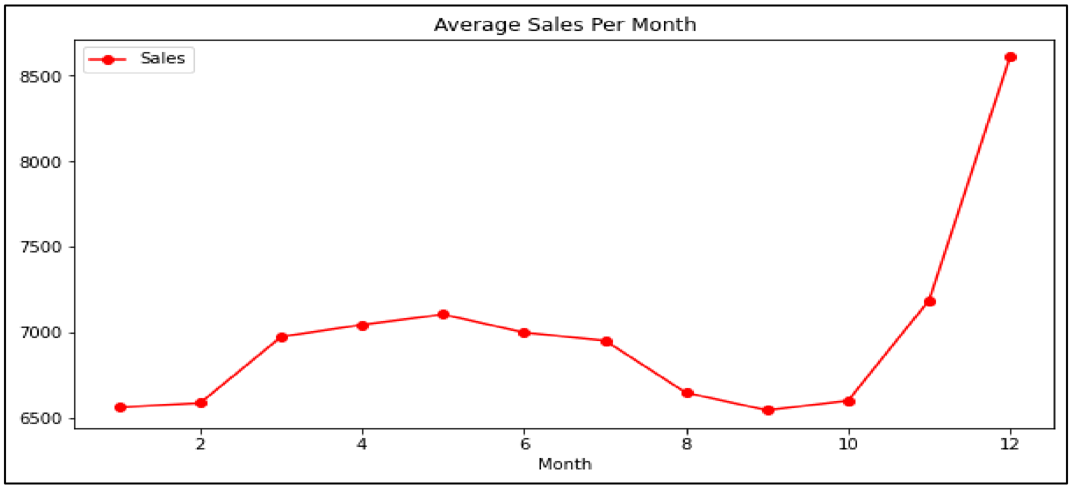
The train.csv and store.csv files are merged based on 'store' feature to gain more insights into the data.



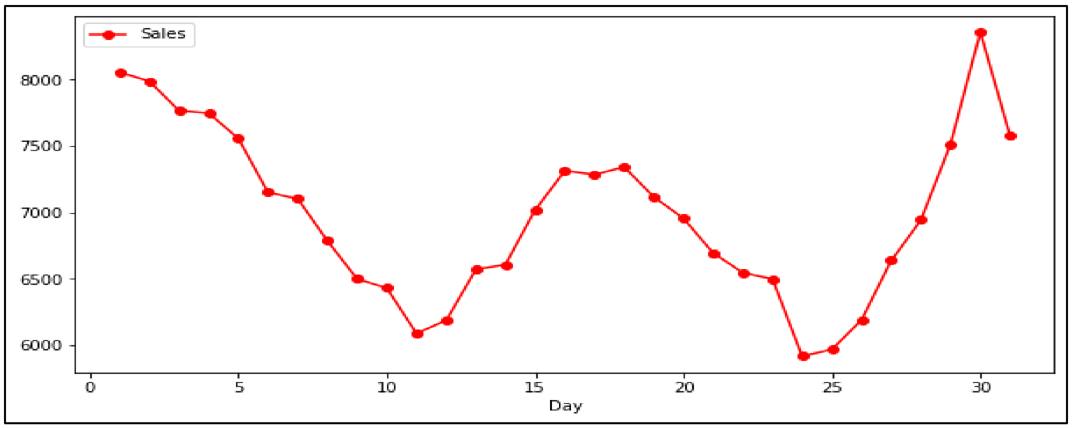
The correlation map suggests a strong positive correlation exists between the Sales and Customers of a store and also a positive correlation between the stores that had a Promo equal to 1(stores running promotions) and number of Customers.

From the 'Date' feature, new features 'Year', 'Month' and 'Day' are extracted as this helps to understand the relation of sales with respect to any given Month and Day.

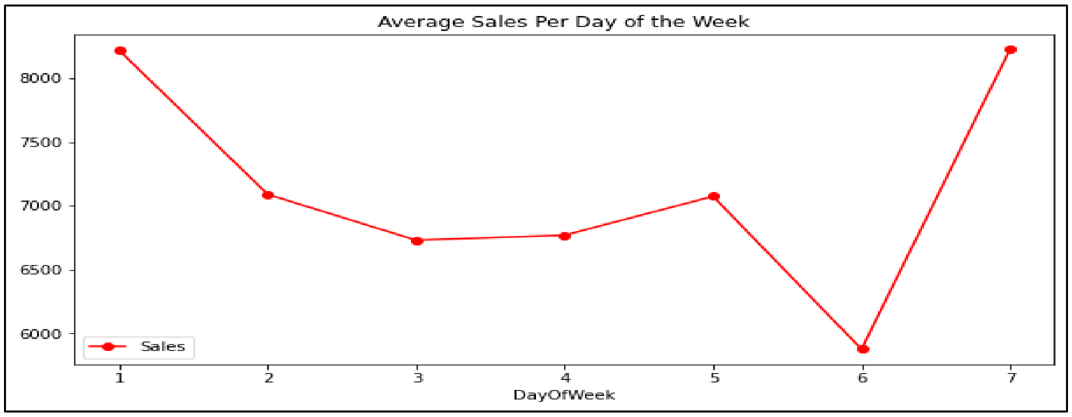
A look at the average sales and number of customers per month reveals that sales peak around Christmas timeframe.



A look at the sales per day of the month shows that minimum number of customers are generally around the 24th of the month, and that most sales are around 30th and 1st of the month.



Also, a look at the sales per day of the week reveals that minimum number of customers are generally around Saturday, and that most sales are on Sunday and Monday.

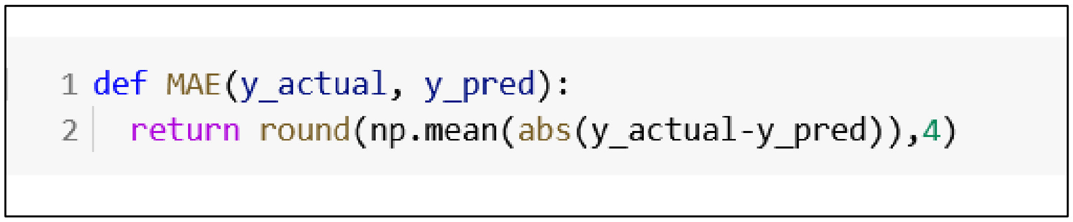

Average Sales Per Day of the Week
8000
7500
7000
6500
6000
Sales
1
2
3
4
5
6
7
DayOfWeek

4. Modeling and Error Analysis

Defining the Error functions before proceeding with modeling and error analysis.

Mean Absolute Error (MAE)

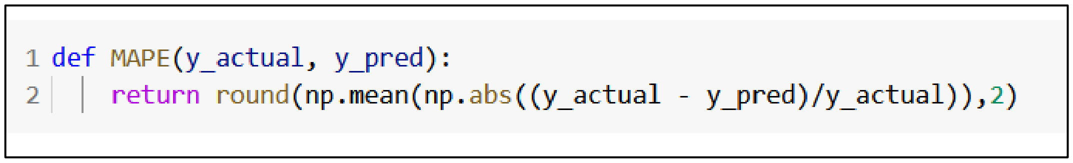
The MAE is the absolute difference between the data and the model's predictions. As we just use the absolute value of the residual, the MAE does not indicate underperformance or overperformance of the model. A small MAE suggests the model is great at prediction, while a large MAE suggests that the model has trouble in certain areas. A MAE of 0 means that the model is a perfect predictor of the outputs.



```python
def MAE(y_actual, y_pred):
    return round(np.mean(abs(y_actual-y_pred)),4)
```

Mean Absolute Percentage Error (MAPE)

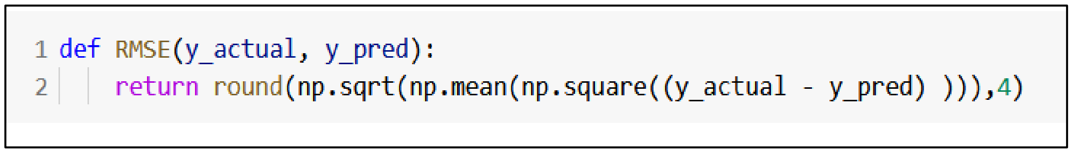
MAPE Shows how far the prediction is from the actual value, on average, as a percentage



```python
def MAPE(y_actual, y_pred):
    return round(np.mean(np.abs((y_actual - y_pred)/y_actual)),2)
```

Root Mean Squared Error (RMSE)

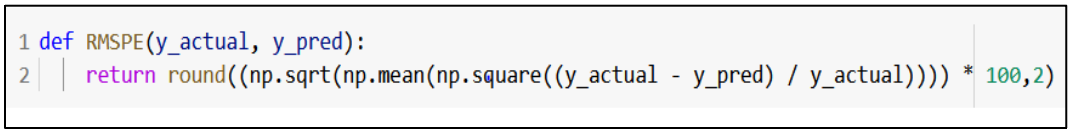
RMSE is the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error, thus larger errors have a disproportionately large effect on RMSE and also, RMSE is sensitive to outliers. RMSE is always non-negative, and a value of 0 indicates a perfect fit to the data.



```python
def RMSE(y_actual, y_pred):
    return round(np.sqrt(np.mean(np.square((y_actual - y_pred) ))),4)
```

Root Mean Squared Percentage Error (RMSPE)

RMSPE gives an idea of the magnitude of the error in relation to the actual values.



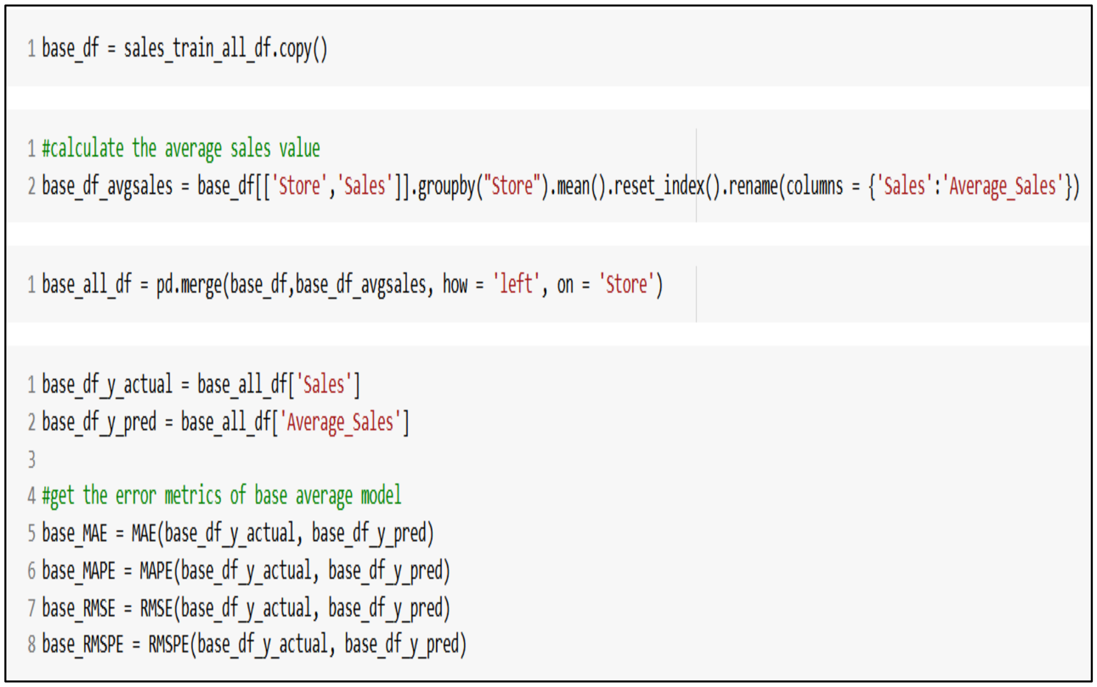
```python
def RMSPE(y_actual, y_pred):
    return round((np.sqrt(np.mean(np.square((y_actual - y_pred) / y_actual)))) * 100,2)
```

4.1 Average Baseline Model

First , we created a Baseline model that just calculates the average sales per store

Here, 'StoreType' and 'Assortment' categorical features are handled by getting dummy variables before feeding data to machine learning algorithms.

‘stateHoliday’ feature is mapped in such a way that only ‘0’ would mean a Non-holiday and rest of values (‘a’,’b’,’c’) would refer to holidays.



```python
base_df = sales_train_all_df.copy()
```

```python
#calculate the average sales value
base_df_avgsales = base_df[['Store','Sales']].groupby("Store").mean().reset_index().rename(columns = {'Sales':'Average_Sales'})
```

```python
base_all_df = pd.merge(base_df,base_df_avgsales, how = 'left', on = 'Store')
```

```python
base_df_y_actual = base_all_df['Sales']
base_df_y_pred = base_all_df['Average_Sales']

#get the error metrics of base average model
base_MAE = MAE(base_df_y_actual, base_df_y_pred)
base_MAPE = MAPE(base_df_y_actual, base_df_y_pred)
base_RMSE = RMSE(base_df_y_actual, base_df_y_pred)
base_RMSPE = RMSPE(base_df_y_actual, base_df_y_pred)
```

Applying Average baseline model to the data yielded the performance summarized below :

Base Model MAE	1456.855
Base Model MAPE	23.62
Base Model RMSE	1957.9119
Base Model RMSPE	39.37

4.2 Linear Regression (Ordinary Least Squares)

Linear Regression is a machine Learning model to find the linear relationship of the data/independent variables with the output/dependent/target variable. The objective of linear regression is to find a line that best fits the data by minimizing the loss function. The loss function is the function that computes the distance between the predicted output of the algorithm and the expected output. While the loss function is a value which is calculated at every instance, the cost function is calculated as an average of the loss function.

Regression equation could be demonstrated as :

Y = (w1*x1)+(w2*x2)+….+(wn*xn)+b

Here ,

Y represents the dependent/target variable,
x1,x2 ….xn represent the independent variables
w1,w2….wn are the constants that show the extent to which these factors influence the target variable.

Applying Linear Regression model to the data yielded the performance summarized below :

Train Score	0.8262
Test Score	0.8287
Train MAE	938.1422
Test MAE	934.7298
Train MAPE	14.29
Test MAPE	14.26
Train RMSE	1292.5497
Test RMSE	1287.4622
Training RMSPE	19.15
Testing RMSPE	19.23

4.3 Ridge Regression (Linear least squares with l2 regularization)

Ridge Regression Model is equivalent to a Linear Regression model plus the L2 regularization, i.e., adding penalty equivalent to square of the magnitude of coefficients. Here, model cost function is altered by adding a penalty that is equivalent to square of the magnitude of the coefficients.

Here, the cost function is regularized in such a way that when the coefficients take large values, the loss function is penalized.
Applying Ridge Regression model to the data yielded the performance summarized below :

Train Score	0.8259
Test Score	0.8285
Train MAE	939.6218
Test MAE	936.2103
Train MAPE	14.32
Test MAPE	14.29
Train RMSE	1293.5579
Test RMSE	1288.4075
Training RMSPE	19.18
Testing RMSPE	19.24

4.4 Lasso Regression (Linear Model trained with L1 regularization)

Lasso Regression Model is equivalent to a Linear Regression model plus the L1 regularization, i.e., adding penalty equivalent to magnitude of coefficients. Here, model cost function is altered by adding a penalty that is equivalent to the magnitude of the coefficients.

Applying Lasso Regression model to the data yielded the performance summarized below :

Train Score	0.8258
Test Score	0.8284
Train MAE	940.0902
Test MAE	936.6873

Train MAPE	14.32
Test MAPE	14.29
Train RMSE	1293.9468
Test RMSE	1288.7855
Training RMSPE	19.18
Testing RMSPE	19.25

4.5 Decision Tree

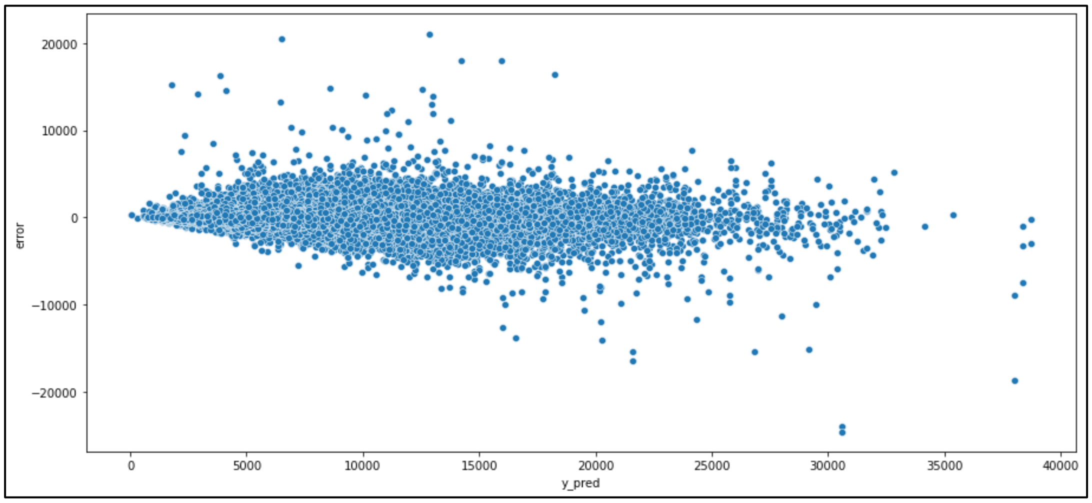
Decision Tree algorithms use a flowchart like tree structure for regression/classification tasks resulting from a series of feature-based splits. A decision tree algorithm starts with a root node, and then passing through a series of internal nodes known as Decision nodes ends with a decision made by leaf node. Decision Tree algorithms can be thought of as inverted tree structures where root is at the top which is then split into multiple nodes through a series of If-else statements.
Here, 'StoreType' and 'Assortment' categorical features are label encoded before feeding data to tree-based machine learning algorithms.

Applying Decision Tree model to the data yielded the performance summarized below :

Train Score	0.9999
Test Score	0.9477
Train MAE	0.0084
Test MAE	463.8825
Train MAPE	0.0
Test MAPE	0.07
Train RMSE	2.149
Test RMSE	711.1437
Training RMSPE	0.05

Testing RMSPE	9.61

Error Distribution of Decision Tree Model



From the above plot, we can observe that there are few stores with higher error. But majority of the points appear to be in a small horizontal area around zero which indicates that variance in the error is small and centered around zero.

5. Advanced Modeling and feature engineering

Ensemble Models

Ensemble models refer to models where either multiple machine learning models or different training data sets are combined for predictions. The predictions of these base models are then aggregated to obtain a final prediction for the unseen data.

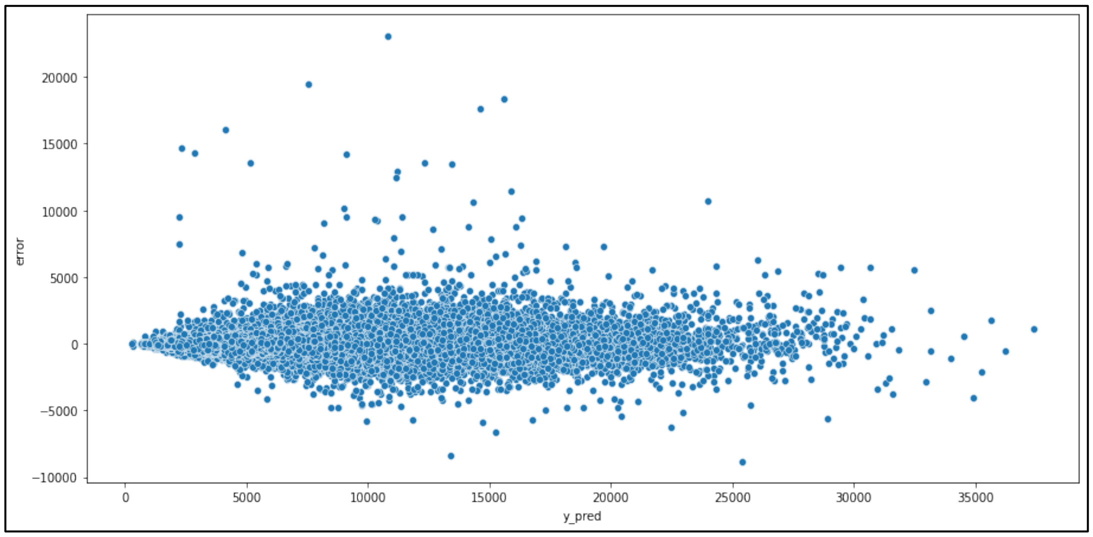
5.1 Random Forest

Random Forest is an ensemble ML model that uses multiple decision trees through a technique called Bootstrapping and Aggregation known as bagging. Random Forest combines multiple decision trees and arrives at final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models, and rows and features are randomly sampled from the dataset forming sample datasets for every model.

Applying Random Forest model to the data yielded the performance summarized below :

Train Score	0.9999
Test Score	0.9481
Train MAE	122.3835
Test MAE	329.9142
Train MAPE	0.02
Test MAPE	0.05
Train RMSE	186.7553
Test RMSE	499.4189
Training RMSPE	2.50
Testing RMSPE	6.64

Error Distribution of Random Forest Model



From the above plot, we can observe that the error distribution is much narrower in Random Forest model than in Decision Tree model and there are few stores with higher error. The variance in the error is smaller than Decision Tree model and centered around zero.

5.2 XGBoost

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems. Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancel out and better one sums up to form final good predictions.

Applying Random Forest model to the data yielded the performance summarized below:

Train Score	0.8848
Test Score	0.8854
Train MAE	771.3302
Test MAE	771.8178
Train MAPE	0.12
Test MAPE	0.12
Train RMSE	1051.5082
Test RMSE	1052.8874
Training RMSPE	16.60
Testing RMSPE	16.31

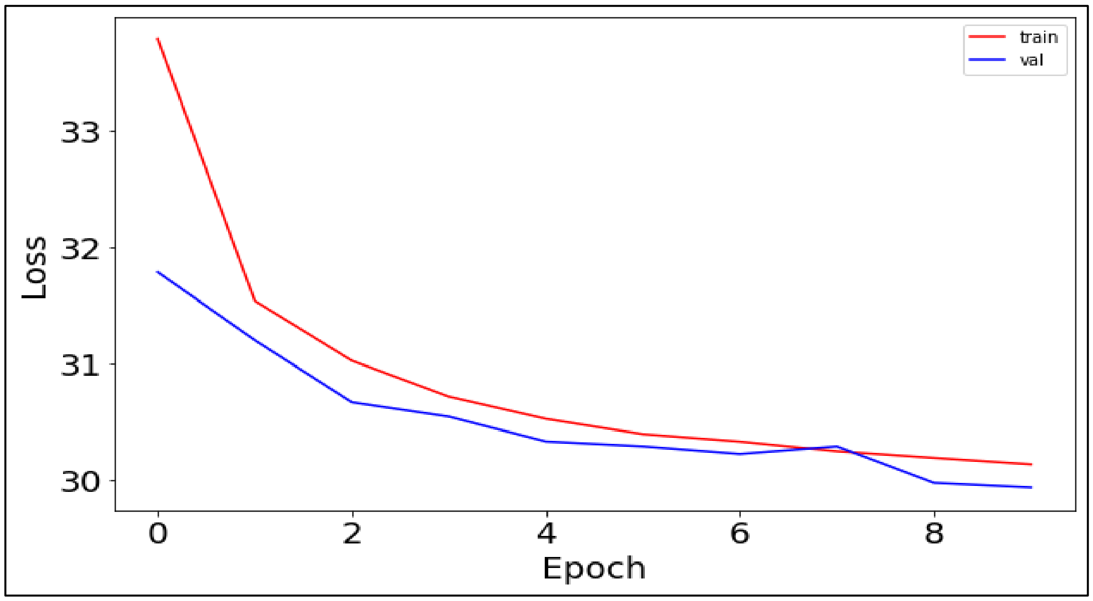
5.3 Deep Learning MLP Model

MLP is a neural network for regression/classification tasks. The data flows in a single direction, that is forward, from the input layers to hidden layer(s) and then to output layer. Backpropagation is a technique where the multi-layer perceptron receives feedback on the error in its results and the MLP adjusts its weights accordingly to make more accurate predictions in the future.
A simple MLP model has a single hidden layer of nodes with an output layer used to make a prediction.
Here, we need to emphasize on the shape of the input dimension which is equal to the number of features in x train data.

```
_________________________________________________________________
 Layer (type)              Output Shape           Param #
=================================================================
 dense (Dense)               (None, 50)             900

 dense_1 (Dense)              (None, 1)              51

=================================================================
Total params: 951
Trainable params: 951
```

```
Non-trainable params: 0
_________________________________________________________________
```

Plot of Loss Function

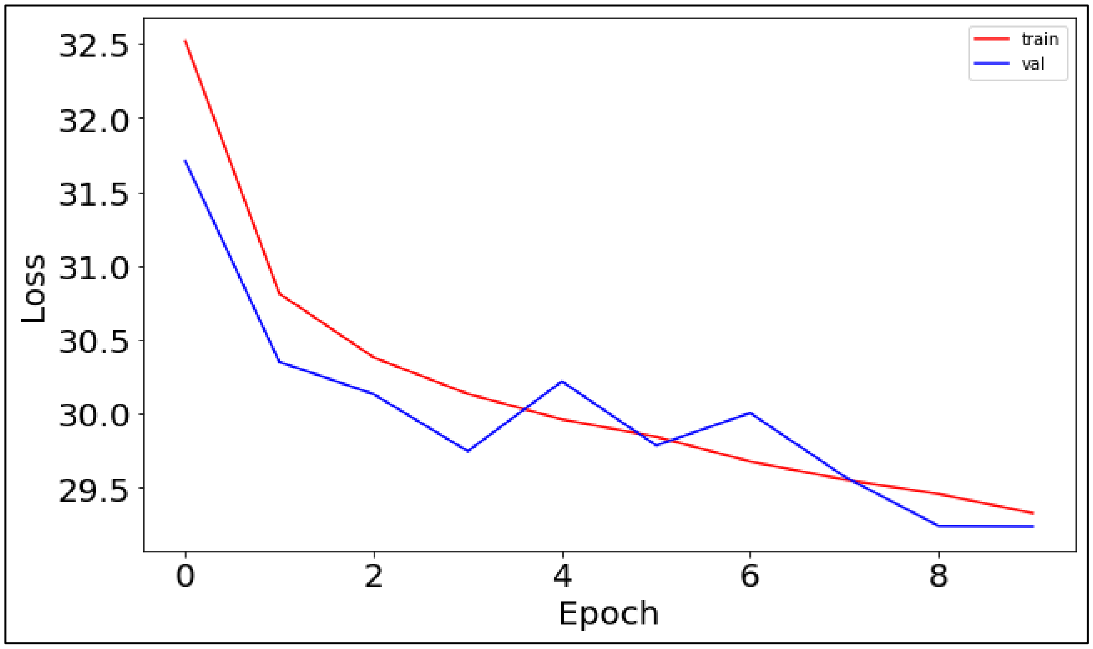


This model gives a score of 78.35% with MAE of 896.51.

Now, let us increase the number of neurons and see if it changes the model performance:

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_2 (Dense)              (None, 350)             6300

dense_3 (Dense)              (None, 1)             351

=================================================================
Total params: 6,651
Trainable params: 6,651
Non-trainable params: 0
_________________________________________________________________
```

Plot of Loss Function

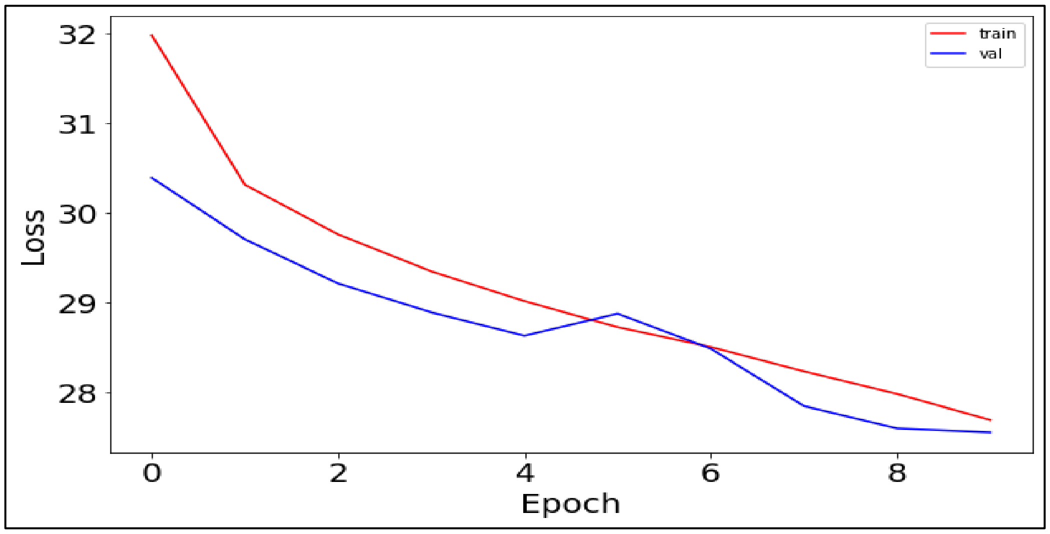


This model gives a score of 80.9% with MAE of 853.54.

Now, let us create a MLP model with additional set of layers and keeping the number of neurons as 300 as that gave a better model performance earlier :

```
________________________________________________________________
 Layer (type)              Output Shape            Param #
================================================================
 dense_4 (Dense)              (None, 350)             6300

 dense_5 (Dense)              (None, 350)             122850

 dense_6 (Dense)              (None, 350)             122850

 dense_7 (Dense)              (None, 350)             122850

 dense_8 (Dense)              (None, 1)             351

================================================================
Total params: 375,201
Trainable params: 375,201
Non-trainable params: 0
________________________________________________________________
```

Plot of Loss Function



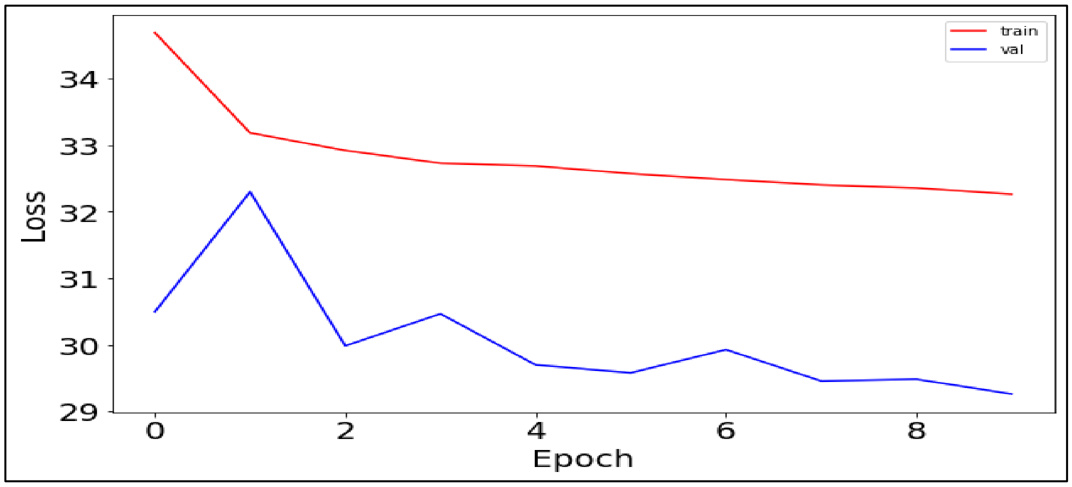
This model gives a score of 85.46% with MAE of 757.39.

When the networks grow deeper, there is a chance of too much of learning from the training data and overfit to it. Dropout is a simple and powerful way to prevent overfitting. Some neurons will be randomly selected and dropped from the network in each layer. Dropout layer needs to be added after activation layer.

```
_________________________________________________________________
Layer (type)              Output Shape           Param #
=================================================================
dense_9 (Dense)             (None, 350)            6300

dropout (Dropout)          (None, 350)            0

dense_10 (Dense)            (None, 350)            122850

dropout_1 (Dropout)         (None, 350)            0

dense_11 (Dense)            (None, 350)            122850

dropout_2 (Dropout)         (None, 350)            0

dense_12 (Dense)            (None, 350)            122850

dropout_3 (Dropout)         (None, 350)            0
```

```
 dense_13 (Dense)          (None, 1)              351

=================================================================
Total params: 375,201
Trainable params: 375,201
Non-trainable params: 0
```

Plot of Loss Function



This model gives a score of 82.93% with MAE of 856.74.

=> A basic MLP neural network with just a single hidden layer consisting of 50 neurons had a score of 78.35% with MAE of approximately 896.

=> When we increased the number of neurons to 350 in this basic MLP network, the score increased to 80.9% and MAE reduced to approximately 853.

=> When we increased the number of hidden layers to 4 with each hidden layer consisting of 350 neurons in the MLP network, the score increased to 85.46% and MAE reduced to approximately 757.

=> When we introduced Dropout layer to this above MLP network to reduce overfitting, there was no noticeable improvement in model performance.

5.4 XGBoost revisited with updated Feature Engineering

In our current project case study until now, as we observe that the best Model performance has a test RMSPE of 6.63%, there is a wide scope to improve the performance of the models developed until now.

Before deployment and Productionization of the model, let us revisit the input data pre-processing used in this project before employing the ML models and see if that results in any improvement in model performance, as we see that feature engineering has played a significant role in our analysis until now.

Even in this input dataset pre-processing function, let us consider only those entries where sales are greater than zero and only entries for open stores.

The input 'Date' feature has been split into day, month, and year similar to what was done in earlier analysis, and these are passed as inputs to our model. First Date column is converted using to_datetime function of pandas and later, the mentioned other features are extracted from Date.

Also, to extract more meaningful information from store.csv training file and to make the features related to Promotion and Competition much more meaningful in our dataset, new features 'CompetitionOpen' has been added which gives the number of months the store has been facing competition due to a competitor being open near the store, feature 'PromoOpen' has been added which gives the number of months the store has been running its Promo2 promotion strategy and a feature 'IsPromoMonth' has been added to find out if the sale on a given date is in the duration of a Promo interval.

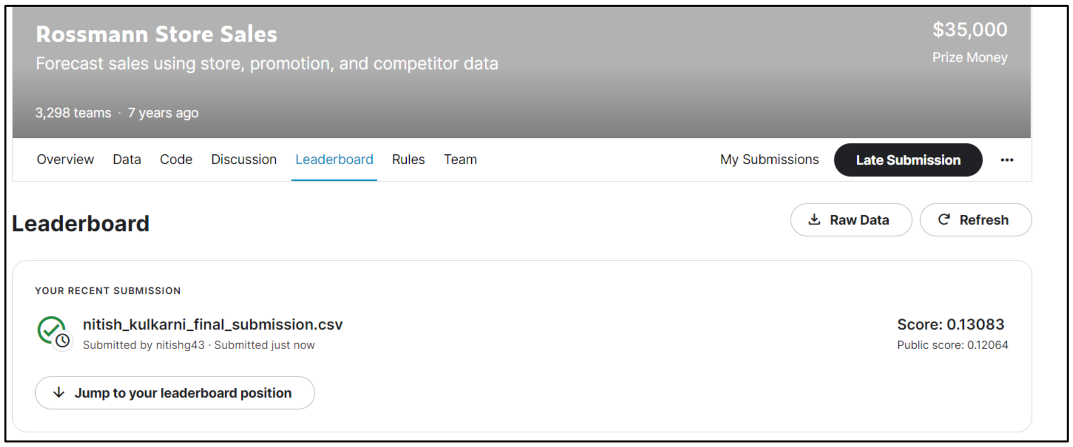
The mappings for features 'StoreType' and 'StateHoliday' will be the same as done earlier.

Applying XGBoost Model to this updated data yielded the performance summarized below :

Train Score	0.9679
Test Score	0.9544
Train MAE	0.0559
Test MAE	0.0669
Train MAPE	0.01
Test MAPE	0.01
Train RMSE	0.0762
Test RMSE	0.0907
Training RMSPE	0.9
Testing RMSPE	1.06

We can see that the updated preprocessing has resulted in significant changes in model performance and improvement in test RMSPE of the best model from 6.63% to 1.06%. Hence, we will be using this model to get the submission csv file to upload in Kaggle, as well as to download the model to be used for deployment.

This submission gives us the final Kaggle submission score of 0.12064.



Leaderboard

YOUR RECENT SUBMISSION

nitish_kulkarni_final_submission.csv
Submitted by nitishg43 · Submitted just now

Score: 0.13083
Public score: 0.12064

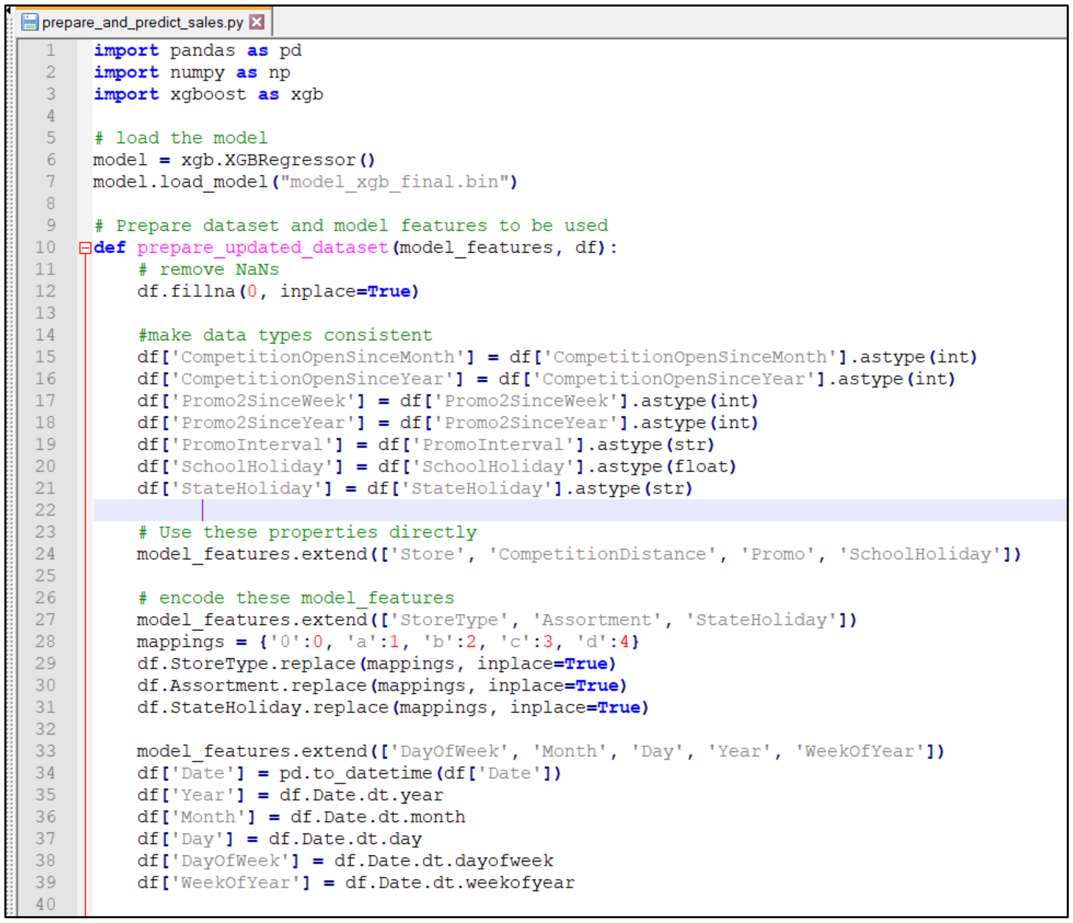
6. Deployment & Productionization

We have used Streamlit and Heroku to deploy our model . Streamlit is an open-source and easy-to-learn Python framework that can be used to create interactive websites for Machine Learning projects. It is an ideal tool for show casing Machine learning projects without the need to know the intricacies of web development, as building an app in Streamlit only requires knowledge of Python.

After we built the application using Streamlit, we have deployed it on Heroku to share it with others for demonstration and usage purposes. Heroku is a platform-as-a-service (PaaS) that can be used to deploy and manage applications built in different programming languages, including Python on cloud.

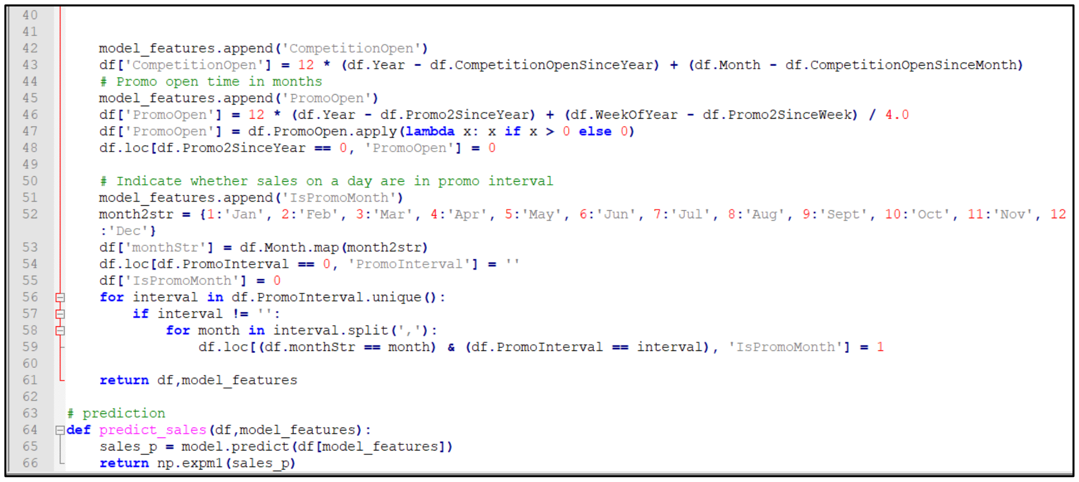
We have followed the steps shown below to generate our application using Streamlit app and later to deploy the application on Heroku :

1) Create the python file **prepare_and_predict_sales.py** containing the function **prepare_updated_dataset** to prepare the input dataset and the function **predict_sales** to predict the sales output from the input dataset.



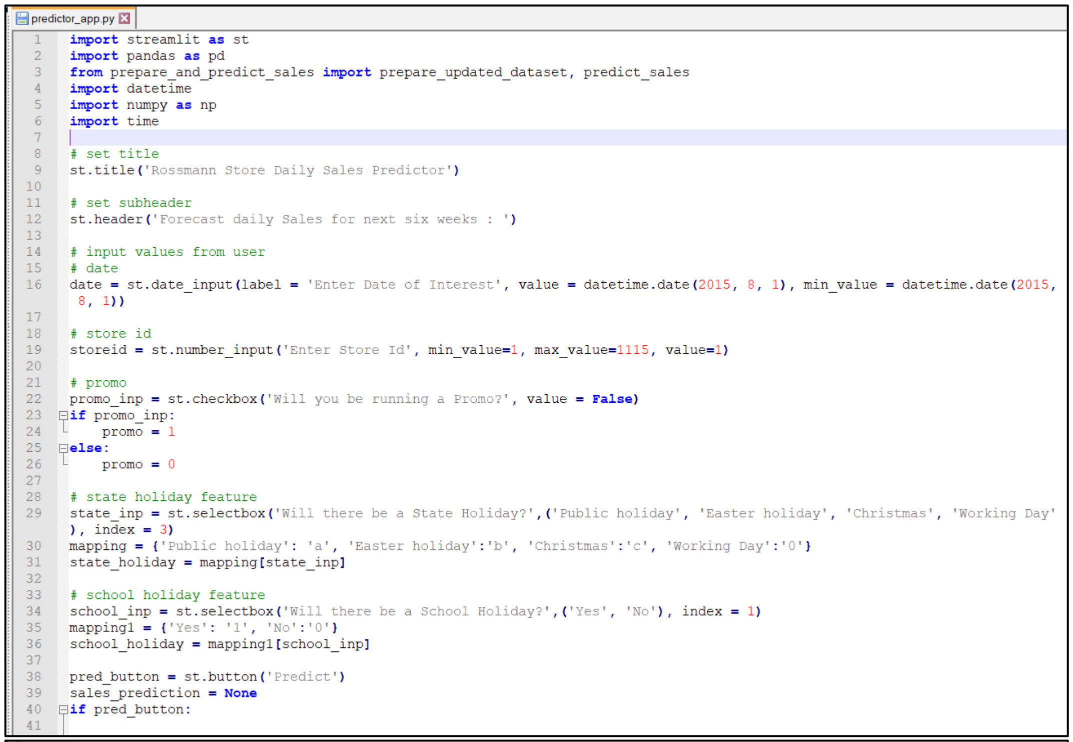
prepare_and_predict_sales.py

```python
import pandas as pd
import numpy as np
import xgboost as xgb

# load the model
model = xgb.XGBRegressor()
model.load_model("model_xgb_final.bin")

# Prepare dataset and model features to be used
def prepare_updated_dataset(model_features, df):
    # remove NaNs
    df.fillna(0, inplace=True)

    #make data types consistent
    df['CompetitionOpenSinceMonth'] = df['CompetitionOpenSinceMonth'].astype(int)
    df['CompetitionOpenSinceYear'] = df['CompetitionOpenSinceYear'].astype(int)
    df['Promo2SinceWeek'] = df['Promo2SinceWeek'].astype(int)
    df['Promo2SinceYear'] = df['Promo2SinceYear'].astype(int)
    df['PromoInterval'] = df['PromoInterval'].astype(str)
    df['SchoolHoliday'] = df['SchoolHoliday'].astype(float)
    df['StateHoliday'] = df['StateHoliday'].astype(str)

    # Use these properties directly
    model_features.extend(['Store', 'CompetitionDistance', 'Promo', 'SchoolHoliday'])

    # encode these model_features
    model_features.extend(['StoreType', 'Assortment', 'StateHoliday'])
    mappings = {'0':0, 'a':1, 'b':2, 'c':3, 'd':4}
    df.StoreType.replace(mappings, inplace=True)
    df.Assortment.replace(mappings, inplace=True)
    df.StateHoliday.replace(mappings, inplace=True)

    model_features.extend(['DayOfWeek', 'Month', 'Day', 'Year', 'WeekOfYear'])
    df['Date'] = pd.to_datetime(df['Date'])
    df['Year'] = df.Date.dt.year
    df['Month'] = df.Date.dt.month
    df['Day'] = df.Date.dt.day
    df['DayOfWeek'] = df.Date.dt.dayofweek
    df['WeekOfYear'] = df.Date.dt.weekofyear

```

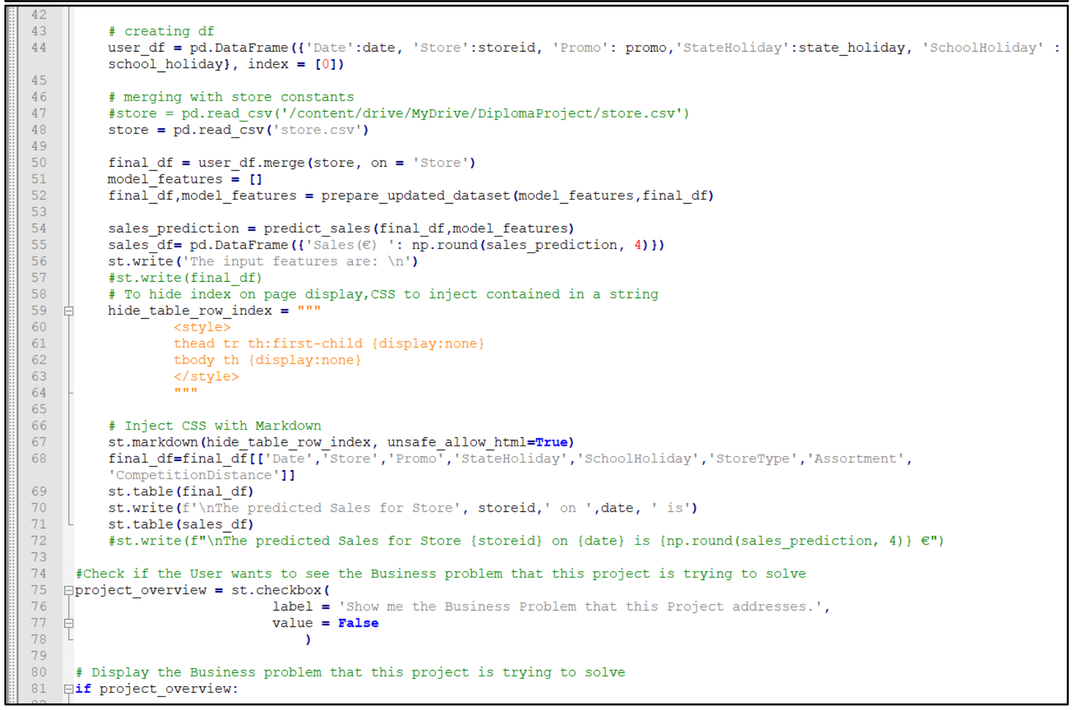


```python


    model_features.append('CompetitionOpen')
    df['CompetitionOpen'] = 12 * (df.Year - df.CompetitionOpenSinceYear) + (df.Month - df.CompetitionOpenSinceMonth)
    # Promo open time in months
    model_features.append('PromoOpen')
    df['PromoOpen'] = 12 * (df.Year - df.Promo2SinceYear) + (df.WeekOfYear - df.Promo2SinceWeek) / 4.0
    df['PromoOpen'] = df.PromoOpen.apply(lambda x: x if x > 0 else 0)
    df.loc[df.Promo2SinceYear == 0, 'PromoOpen'] = 0

    # Indicate whether sales on a day are in promo interval
    model_features.append('IsPromoMonth')
    month2str = {1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun', 7:'Jul', 8:'Aug', 9:'Sept', 10:'Oct', 11:'Nov', 12
    :'Dec'}
    df['monthStr'] = df.Month.map(month2str)
    df.loc[df.PromoInterval == 0, 'PromoInterval'] = ''
    df['IsPromoMonth'] = 0
    for interval in df.PromoInterval.unique():
        if interval != '':
            for month in interval.split(','):
                df.loc[(df.monthStr == month) & (df.PromoInterval == interval), 'IsPromoMonth'] = 1

    return df,model_features

# prediction
def predict_sales(df,model_features):
    sales_p = model.predict(df[model_features])
    return np.expm1(sales_p)
```

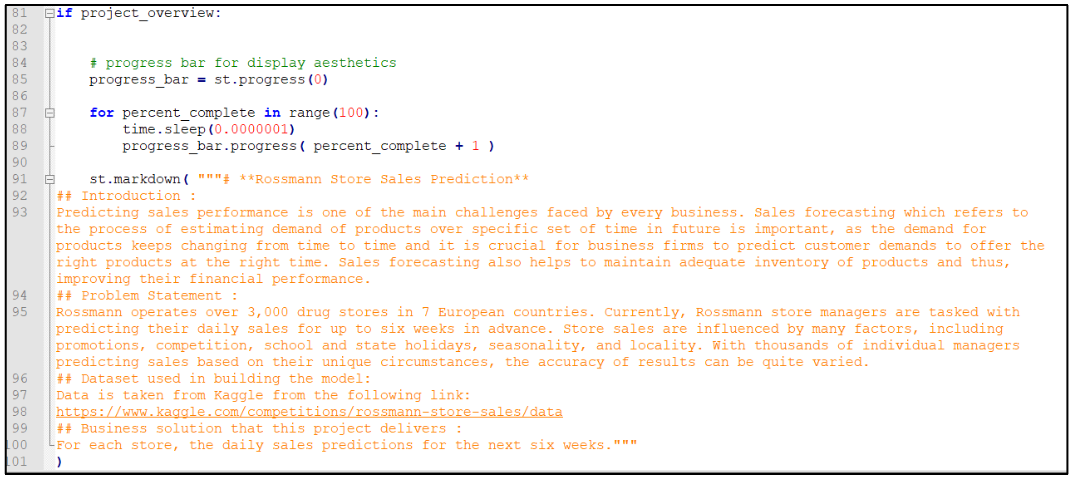
2) Create the python file **predictor_app.py** that imports above functions from **prepare_and_predict_sales.py** file and this file contains python code to use Streamlit framework to generate an application that enables users to enter the input values and get the predicted sales output.



```python
import streamlit as st
import pandas as pd
from prepare_and_predict_sales import prepare_updated_dataset, predict_sales
import datetime
import numpy as np
import time

# set title
st.title('Rossmann Store Daily Sales Predictor')

# set subheader
st.header('Forecast daily Sales for next six weeks : ')

# input values from user
# date
date = st.date_input(label = 'Enter Date of Interest', value = datetime.date(2015, 8, 1), min_value = datetime.date(2015, 8, 1))

# store id
storeid = st.number_input('Enter Store Id', min_value=1, max_value=1115, value=1)

# promo
promo_inp = st.checkbox('Will you be running a Promo?', value = False)
if promo_inp:
    promo = 1
else:
    promo = 0

# state holiday feature
state_inp = st.selectbox('Will there be a State Holiday?',('Public holiday', 'Easter holiday', 'Christmas', 'Working Day'), index = 3)
mapping = {'Public holiday': 'a', 'Easter holiday':'b', 'Christmas':'c', 'Working Day':'0'}
state_holiday = mapping[state_inp]

# school holiday feature
school_inp = st.selectbox('Will there be a School Holiday?',('Yes', 'No'), index = 1)
mapping1 = {'Yes': '1', 'No':'0'}
school_holiday = mapping1[school_inp]

pred_button = st.button('Predict')
sales_prediction = None
if pred_button:

```

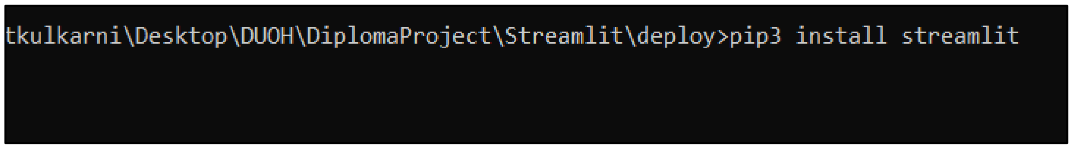


```python

    # creating df
    user_df = pd.DataFrame({'Date':date, 'Store':storeid, 'Promo': promo,'StateHoliday':state_holiday, 'SchoolHoliday' : school_holiday}, index = [0])

    # merging with store constants
    #store = pd.read_csv('/content/drive/MyDrive/DiplomaProject/store.csv')
    store = pd.read_csv('store.csv')

    final_df = user_df.merge(store, on = 'Store')
    model_features = []
    final_df,model_features = prepare_updated_dataset(model_features,final_df)

    sales_prediction = predict_sales(final_df,model_features)
    sales_df= pd.DataFrame({'Sales(€) ': np.round(sales_prediction, 4)})
    st.write('The input features are: \n')
    #st.write(final_df)
    # To hide index on page display,CSS to inject contained in a string
    hide_table_row_index = """
            <style>
            thead tr th:first-child {display:none}
            tbody th {display:none}
            </style>
            """

    # Inject CSS with Markdown
    st.markdown(hide_table_row_index, unsafe_allow_html=True)
    final_df=final_df[['Date','Store','Promo','StateHoliday','SchoolHoliday','StoreType','Assortment','CompetitionDistance']]
    st.table(final_df)
    st.write(f'\nThe predicted Sales for Store', storeid,' on ',date, ' is')
    st.table(sales_df)
    #st.write(f"\nThe predicted Sales for Store {storeid} on {date} is {np.round(sales_prediction, 4)} €")

#Check if the User wants to see the Business problem that this project is trying to solve
project_overview = st.checkbox(
                        label = 'Show me the Business Problem that this Project addresses.',
                        value = False
                            )

# Display the Business problem that this project is trying to solve
if project_overview:
```



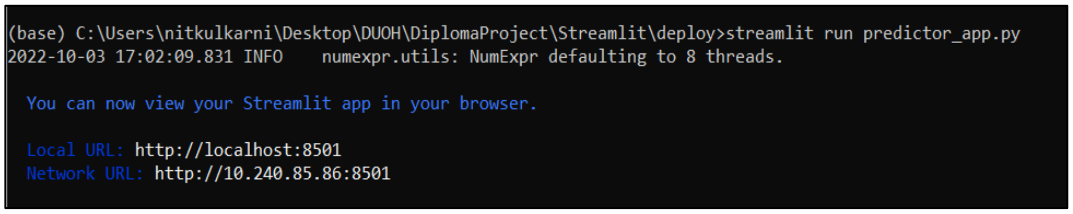
```
if project_overview:


    # progress bar for display aesthetics
    progress_bar = st.progress(0)

    for percent_complete in range(100):
        time.sleep(0.0000001)
        progress_bar.progress( percent_complete + 1 )

    st.markdown( """# **Rossmann Store Sales Prediction**
## Introduction :
Predicting sales performance is one of the main challenges faced by every business. Sales forecasting which refers to the process of estimating demand of products over specific set of time in future is important, as the demand for products keeps changing from time to time and it is crucial for business firms to predict customer demands to offer the right products at the right time. Sales forecasting also helps to maintain adequate inventory of products and thus, improving their financial performance.
## Problem Statement :
Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.
## Dataset used in building the model:
Data is taken from Kaggle from the following link:
https://www.kaggle.com/competitions/rossmann-store-sales/data
## Business solution that this project delivers :
For each store, the daily sales predictions for the next six weeks."""
)
```

3) Install Streamlit application by running the command “pip3 install streamlit”.

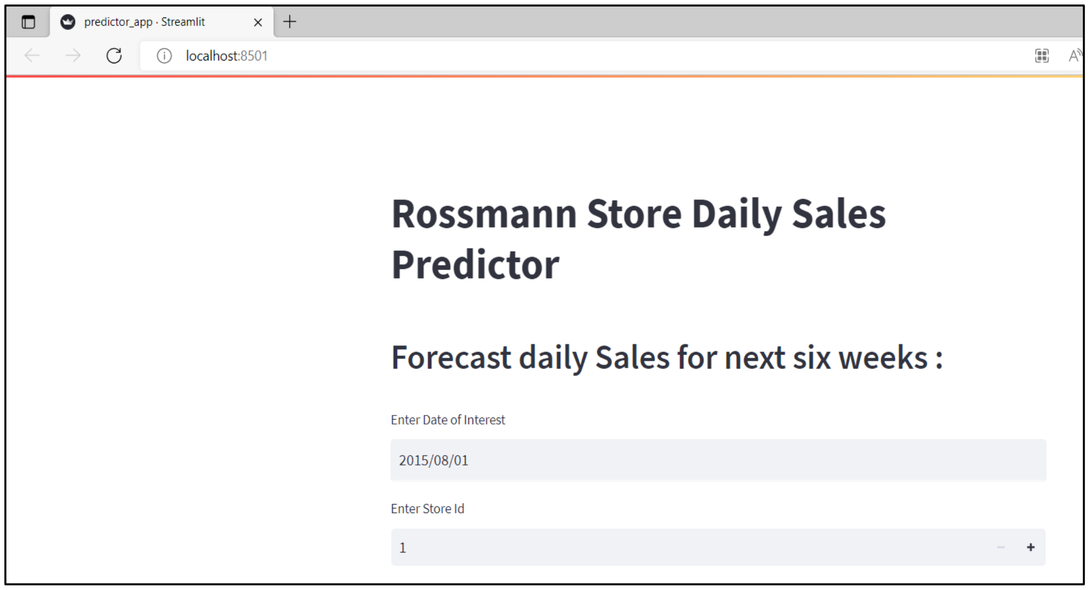


```
tkulkarni\Desktop\DUOH\DiplomaProject\Streamlit\deploy>pip3 install streamlit
```

4) Run the predictor_app.py locally to verify if the application is working by running the command “streamlit run predictor_app.py’.



```
(base) C:\Users\nitkulkarni\Desktop\DUOH\DiplomaProject\Streamlit\deploy>streamlit run predictor_app.py
2022-10-03 17:02:09.831 INFO     numexpr.utils: NumExpr defaulting to 8 threads.

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.240.85.86:8501
```


predictor_app · Streamlit

localhost:8501

Rossmann Store Daily Sales Predictor

Forecast daily Sales for next six weeks :

Enter Date of Interest

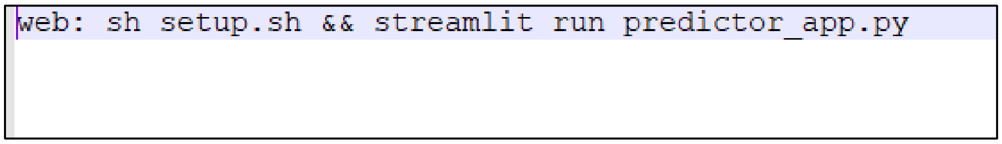
2015/08/01

Enter Store Id

1

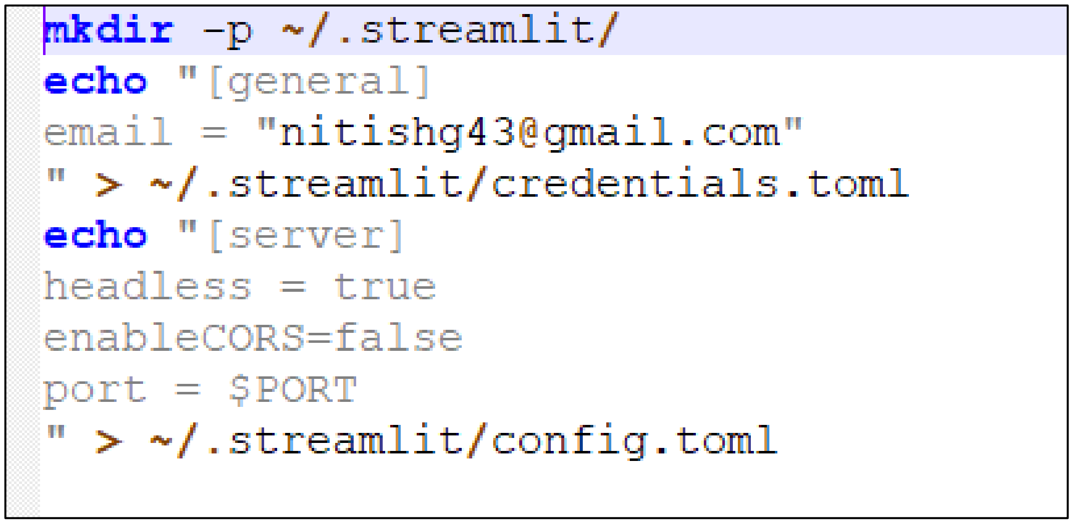
5) After we have verified that the application is working, we will create the **requirements.txt** file that will contain the packages required to run the application on Heroku. This file can be created first by creating a local virtual environment, then installing all packages that we need to run the model, and after verifying that the Streamlit application is working, run the command “pip freeze > requirements.txt”.

6) Create **Procfile** which contains the following text to inform that this is a web application.



```
web: sh setup.sh && streamlit run predictor_app.py
```

7) Create file **setup.sh** containing the following text:

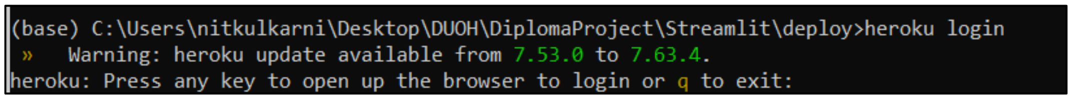


```
mkdir -p ~/.streamlit/
echo "[general]
email = "nitishg43@gmail.com"
" > ~/.streamlit/credentials.toml
echo "[server]
headless = true
enableCORS=false
port = $PORT
" > ~/.streamlit/config.toml
```

8) Create a Heroku account and then download the Heroku CLI.

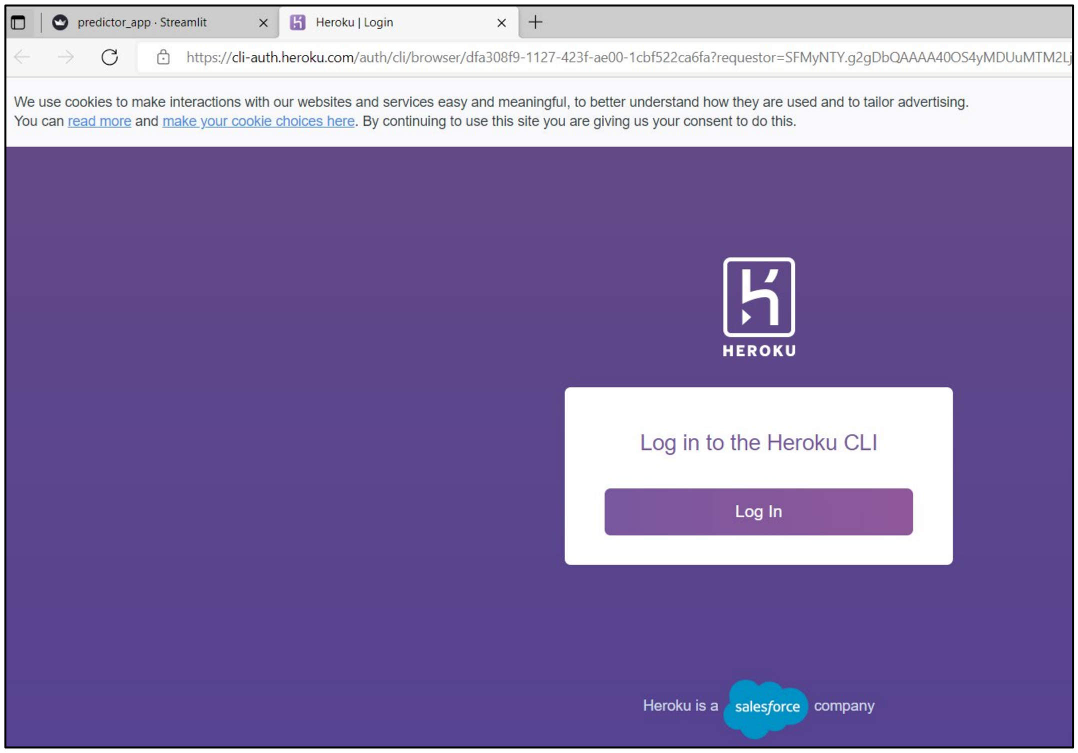
9) Place all the above-mentioned files in a folder, create a git repository by running the “git init” command in the directory.

10) Login into Heroku in the CLI using the command “heroku login”.



```
(base) C:\Users\nitkulkarni\Desktop\DUOH\DiplomaProject\Streamlit\deploy>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.63.4.
heroku: Press any key to open up the browser to login or q to exit:
```

Press any key and we will be redirected to the login page. Here in this page, log in to your account.


predictor_app · Streamlit
Heroku | Login
https://cli-auth.heroku.com/auth/cli/browser/dfa308f9-1127-423f-ae00-1cbf522ca6fa?requestor=SFMyNTY.g2gDbQAAAA40OS4yMDUuMTM2Lj

We use cookies to make interactions with our websites and services easy and meaningful, to better understand how they are used and to tailor advertising. You can read more and make your cookie choices here. By continuing to use this site you are giving us your consent to do this.

HEROKU
Log in to the Heroku CLI
Log In
Heroku is a salesforce company

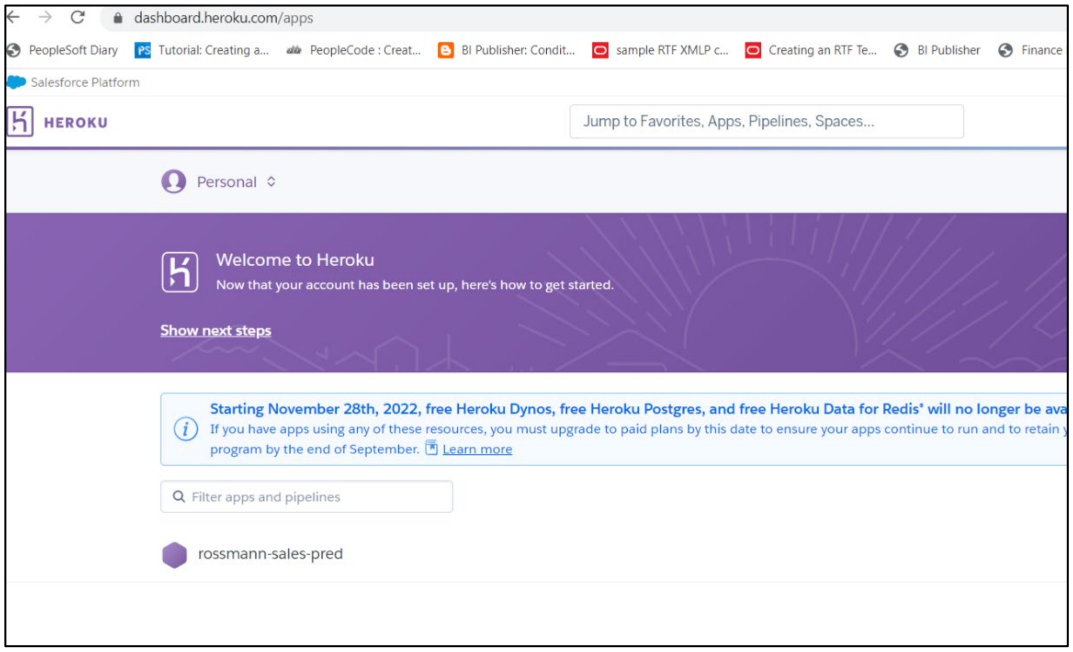
11) We will then create our application “**rossmann-sales-pred**” using the command “heroku create rossmann-sales-pred” which creates a Heroku instance.

12) Then , we will run the following commands to push the code into the newly created Heroku instance.

```
git add .

git commit -am " make it better"

git push heroku master
```

While running “git push heroku master” command, system automatically detects that our application is a Python application and installs all the dependencies from **requirements.txt** file.

Once the command has run, login to https://dashboard.heroku.com/ , select the application “**rossmann-sales-pred**” and launch the application and verify if is working as expected.


dashboard.heroku.com/apps

PeopleSoft Diary | Tutorial: Creating a... | PeopleCode : Creat... | BI Publisher: Condit... | sample RTF XMLP c... | Creating an RTF Te... | BI Publisher | Finance

Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal

Welcome to Heroku

Now that your account has been set up, here's how to get started.

Show next steps

Starting November 28th, 2022, free Heroku Dynos, free Heroku Postgres, and free Heroku Data for Redis® will no longer be ava

If you have apps using any of these resources, you must upgrade to paid plans by this date to ensure your apps continue to run and to retain y

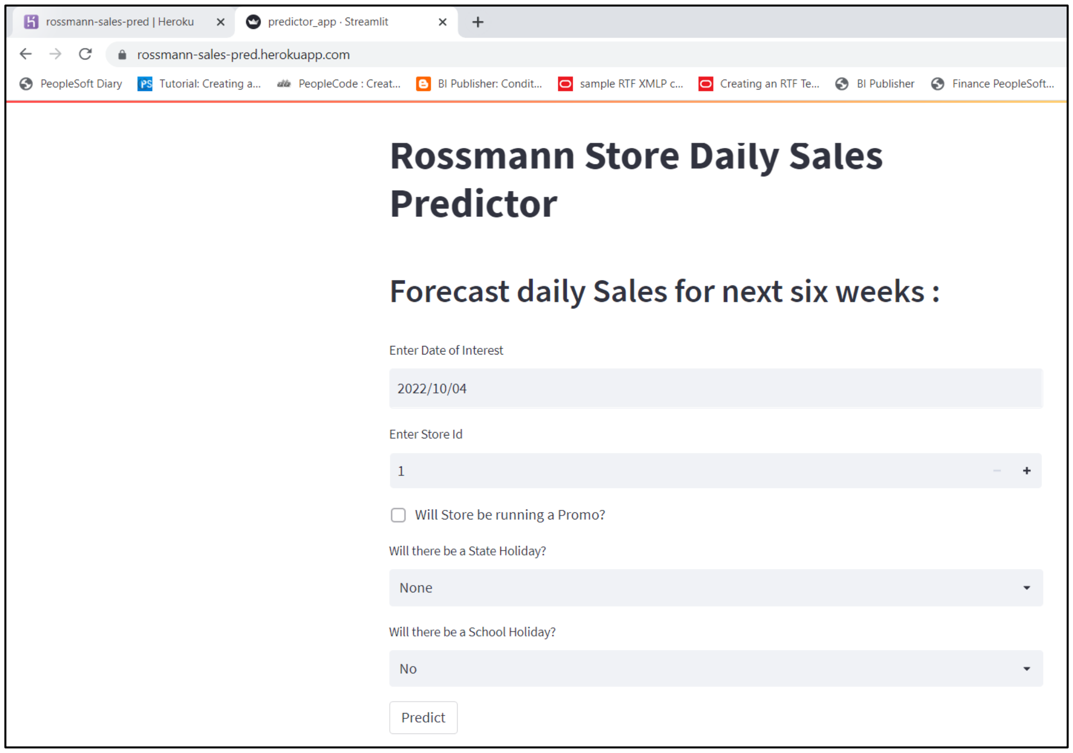
program by the end of September. Learn more

Filter apps and pipelines

rossmann-sales-pred

This is the URL of the web application of deployed model .

https://rossmann-sales-pred.herokuapp.com/


rossmann-sales-pred | Heroku | predictor_app · Streamlit

rossmann-sales-pred.herokuapp.com

PeopleSoft Diary | Tutorial: Creating a... | PeopleCode : Creat... | BI Publisher: Condit... | sample RTF XMLP c... | Creating an RTF Te... | BI Publisher | Finance PeopleSoft...

Rossmann Store Daily Sales Predictor

Forecast daily Sales for next six weeks :

Enter Date of Interest: 2022/10/04

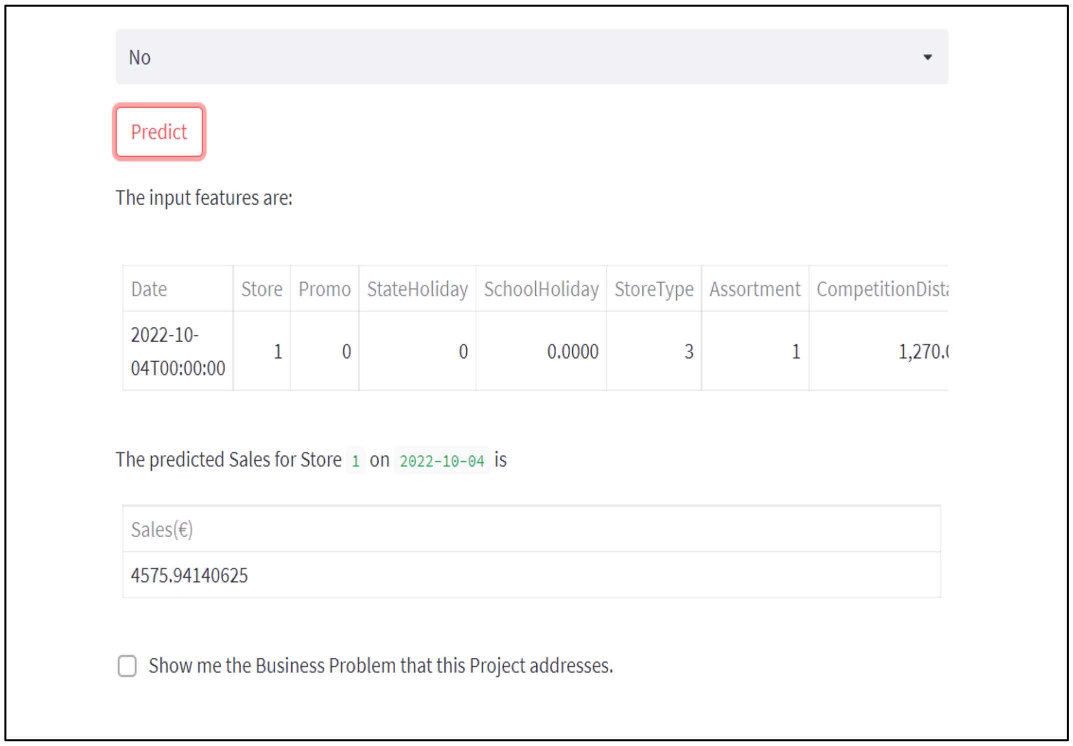
Enter Store Id: 1

☐ Will Store be running a Promo?

Will there be a State Holiday?: None

Will there be a School Holiday?: No

Predict


No

Predict

The input features are:

Date	Store	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDist
2022-10-04T00:00:00	1	0	0	0.0000	3	1	1,270.0

The predicted Sales for Store `1` on `2022-10-04` is

Sales(€)
4575.94140625

☐ Show me the Business Problem that this Project addresses.

7. References

1. https://www.kaggle.com/competitions/rossmann-store-sales/overview
2. https://www.kaggle.com/competitions/rossmann-store-sales/data
3. https://www.kaggle.com/competitions/rossmann-store-sales/overview/evaluation
4. https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/022.pdf
5. http://cs229.stanford.edu/proj2015/192_report.pdf
6. https://businessjargons.com/
7. https://stackoverflow.com/
8. https://www.kaggle.com/competitions/rossmann-store-sales/discussion/17026#96290
9. ttps://www.analyticsvidhya.com/blog/2021/06/deploy-your-ml-dl-streamlit-application-on-heroku/
10. https://towardsdatascience.com/quickly-build-and-deploy-an-application-with-streamlit-988ca08c7e83
11. https://gilberttanner.com/blog/deploying-your-streamlit-dashboard-with-heroku/