

Rossmann Store Sales Prediction

Forecasting sales using the store, promotion, and competitor data

PHASE 5 : DEPLOYMENT AND PRODUCTIONIZATION

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, we will 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.

```


1 def prepare_updated_dataset(model_features, df):
2     # remove NaNs
3     df.fillna(0, inplace=True)
4     df.loc[df.Open.isnull(), 'Open'] = 1
5     # these features will be used directly
6     model_features.extend(['Store', 'CompetitionDistance', 'Promo', 'SchoolHoliday'])
7
8     # Mapping 'StoreType', 'Assortment' and 'StateHoliday' features as shown below
9     model_features.extend(['StoreType', 'Assortment', 'StateHoliday'])
10    mappings = {'0':0, 'a':1, 'b':2, 'c':3, 'd':4}
11    df.StoreType.replace(mappings, inplace=True)
12    df.Assortment.replace(mappings, inplace=True)
13    df.StateHoliday.replace(mappings, inplace=True)
14
15    #convering Date column using to_datetime function and then extracting relevant features from Date.
16    model_features.extend(['DayOfWeek', 'Month', 'Day', 'Year', 'WeekOfYear'])
17    df['Date'] = pd.to_datetime(df['Date'])
18    df['Year'] = df.Date.dt.year
19    df['Month'] = df.Date.dt.month
20    df['Day'] = df.Date.dt.day
21    df['DayOfWeek'] = df.Date.dt.dayofweek
22    df['WeekOfYear'] = df.Date.dt.weekofyear
23
24    # 'CompetitionOpen' gives the number of months the store has been facing competition
25    model_features.append('CompetitionOpen')
26    df['CompetitionOpen'] = 12 * (df.Year - df.CompetitionOpenSinceYear) + \
27    | | (df.Month - df.CompetitionOpenSinceMonth)
28    # 'PromoOpen' gives the number of months the store has been running its Promo2 promotions
29    model_features.append('PromoOpen')
30    df['PromoOpen'] = 12 * (df.Year - df.Promo2SinceYear) + \
31    | | (df.WeekOfYear - df.Promo2SinceWeek) / 4.0
32    df['PromoOpen'] = df.PromoOpen.apply(lambda x: x if x > 0 else 0)
33    df.loc[df.Promo2SinceYear == 0, 'PromoOpen'] = 0
34
35    # 'IsPromoMonth' informs if the sale on a given date is in the duration of a Promo interval
36    model_features.append('IsPromoMonth')
37    month2str = {1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun', \
38    | | | | 7:'Jul', 8:'Aug', 9:'Sept', 10:'Oct', 11:'Nov', 12:'Dec'}
39    df['monthStr'] = df.Month.map(month2str)
40    df.loc[df.PromoInterval == 0, 'PromoInterval'] = ''
41    df['IsPromoMonth'] = 0
42    for interval in df.PromoInterval.unique():
43    | | if interval != '':
44    | | | | for month in interval.split(','):
45    | | | | | | df.loc[(df.monthStr == month) & (df.PromoInterval == interval), 'IsPromoMonth'] = 1
46
47    return df

```

```

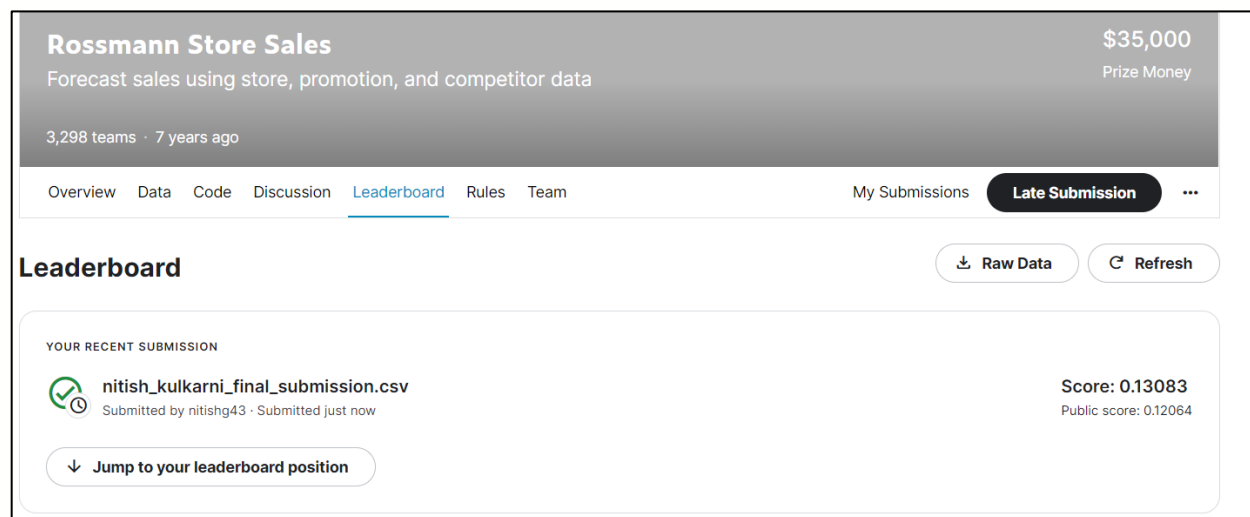
1 # These parameters have been finalised after lot of randomised trials as grid search was taking hours
2 #to train the model on colab
3 xgb_watchlist = [(X_train[model_features], y_train), (X_valid[model_features], y_valid)]
4 xgb_final_model = xgb.XGBRegressor(n_estimators=400, max_depth=10, subsample = 0.9, colsample_bytree = 0.7, 1
5 xgb_final_model.fit(X_train[model_features], y_train,
6 | | | | eval_set=xgb_watchlist,
7 | | | | early_stopping_rounds=20)
8

```

 XGBoost Model Train Score : 0.9679202361107538 , XGBoost Model Test Score : 0.9543885551172726
 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.



The screenshot displays the Kaggle competition interface for 'Rossmann Store Sales'. The header shows the competition title, a prize money of \$35,000, and the number of teams (3,298) and time (7 years ago). The navigation bar includes links for Overview, Data, Code, Discussion, Leaderboard (selected), Rules, and Team. The Leaderboard section shows a recent submission by 'nitish_kulkarni_final_submission.csv' with a score of 0.13083 and a public score of 0.12064. A button labeled 'Jump to your leaderboard position' is visible.

We will be using 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 build the application using Streamlit, we will be using 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.

```

1 import pandas as pd
2 import numpy as np
3 import xgboost as xgb
4
5 # load the model
6 model = xgb.XGBRegressor()
7 model.load_model("model_xgb_final.bin")
8
9 # Prepare dataset and model features to be used
10 def prepare_updated_dataset(model_features, df):
11     # remove NaNs
12     df.fillna(0, inplace=True)
13
14     #make data types consistent
15     df['CompetitionOpenSinceMonth'] = df['CompetitionOpenSinceMonth'].astype(int)
16     df['CompetitionOpenSinceYear'] = df['CompetitionOpenSinceYear'].astype(int)
17     df['Promo2SinceWeek'] = df['Promo2SinceWeek'].astype(int)
18     df['Promo2SinceYear'] = df['Promo2SinceYear'].astype(int)
19     df['PromoInterval'] = df['PromoInterval'].astype(str)
20     df['SchoolHoliday'] = df['SchoolHoliday'].astype(float)
21     df['StateHoliday'] = df['StateHoliday'].astype(str)
22
23     # Use these properties directly
24     model_features.extend(['Store', 'CompetitionDistance', 'Promo', 'SchoolHoliday'])
25
26     # encode these model features
27     model_features.extend(['StoreType', 'Assortment', 'StateHoliday'])
28     mappings = {'0':0, 'a':1, 'b':2, 'c':3, 'd':4}
29     df.StoreType.replace(mappings, inplace=True)
30     df.Assortment.replace(mappings, inplace=True)
31     df.StateHoliday.replace(mappings, inplace=True)
32
33     model_features.extend(['DayOfWeek', 'Month', 'Day', 'Year', 'WeekOfYear'])
34     df['Date'] = pd.to_datetime(df['Date'])
35     df['Year'] = df.Date.dt.year
36     df['Month'] = df.Date.dt.month
37     df['Day'] = df.Date.dt.day
38     df['DayOfWeek'] = df.Date.dt.dayofweek
39     df['WeekOfYear'] = df.Date.dt.weekofyear
40
41
42     model_features.append('CompetitionOpen')
43     df['CompetitionOpen'] = 12 * (df.Year - df.CompetitionOpenSinceYear) + (df.Month - df.CompetitionOpenSinceMonth)
44     # Promo open time in months
45     model_features.append('PromoOpen')
46     df['PromoOpen'] = 12 * (df.Year - df.Promo2SinceYear) + (df.WeekOfYear - df.Promo2SinceWeek) / 4.0
47     df['PromoOpen'] = df.PromoOpen.apply(lambda x: x if x > 0 else 0)
48     df.loc[df.Promo2SinceYear == 0, 'PromoOpen'] = 0
49
50     # Indicate whether sales on a day are in promo interval
51     model_features.append('IsPromoMonth')
52     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'}
53     df['monthStr'] = df.Month.map(month2str)
54     df.loc[df.PromoInterval == 0, 'PromoInterval'] = ''
55     df['IsPromoMonth'] = 0
56     for interval in df.PromoInterval.unique():
57         if interval != '':
58             for month in interval.split(','):
59                 df.loc[(df.monthStr == month) & (df.PromoInterval == interval), 'IsPromoMonth'] = 1
60
61     return df, model_features
62
63 # prediction
64 def predict_sales(df, model_features):
65     sales_p = model.predict(df[model_features])
66     return np.expml(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.

```
1 import streamlit as st
2 import pandas as pd
3 from prepare_and_predict_sales import prepare_updated_dataset, predict_sales
4 import datetime
5 import numpy as np
6 import time
7
8 # set title
9 st.title('Rossmann Store Daily Sales Predictor')
10
11 # set subheader
12 st.header('Forecast daily Sales for next six weeks : ')
13
14 # input values from user
15 # date
16 date = st.date_input(label = 'Enter Date of Interest', value = datetime.date(2015, 8, 1), min_value = datetime.date(2015, 8, 1))
17
18 # store id
19 storeid = st.number_input('Enter Store Id', min_value=1, max_value=1115, value=1)
20
21 # promo
22 promo_inp = st.checkbox('Will you be running a Promo?', value = False)
23 if promo_inp:
24     promo = 1
25 else:
26     promo = 0
27
28 # state holiday feature
29 state_inp = st.selectbox('Will there be a State Holiday?', ('Public holiday', 'Easter holiday', 'Christmas', 'Working Day'), index = 3)
30 mapping = {'Public holiday': 'a', 'Easter holiday': 'b', 'Christmas': 'c', 'Working Day': '0'}
31 state_holiday = mapping[state_inp]
32
33 # school holiday feature
34 school_inp = st.selectbox('Will there be a School Holiday?', ('Yes', 'No'), index = 1)
35 mapping1 = {'Yes': '1', 'No': '0'}
36 school_holiday = mapping1[school_inp]
37
38 pred_button = st.button('Predict')
39 sales_prediction = None
40 if pred_button:
41
42
43     # creating df
44     user_df = pd.DataFrame({'Date': date, 'Store': storeid, 'Promo': promo, 'StateHoliday': state_holiday, 'SchoolHoliday': school_holiday}, index = [0])
45
46     # merging with store constants
47     # store = pd.read_csv('/content/drive/MyDrive/DiplomaProject/store.csv')
48     store = pd.read_csv('store.csv')
49
50     final_df = user_df.merge(store, on = 'Store')
51     model_features = []
52     final_df, model_features = prepare_updated_dataset(model_features, final_df)
53
54     sales_prediction = predict_sales(final_df, model_features)
55     sales_df = pd.DataFrame({'Sales(€)': np.round(sales_prediction, 4)})
56     st.write('The input features are: \n')
57     st.write(final_df)
58     # To hide index on page display, CSS to inject contained in a string
59     hide_table_row_index = """
60         <style>
61         thead tr th:first-child {display:none}
62         tbody th {display:none}
63         </style>
64         """
65
66     # Inject CSS with Markdown
67     st.markdown(hide_table_row_index, unsafe_allow_html=True)
68     final_df = final_df[['Date', 'Store', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance']]
69     st.table(final_df)
70     st.write(f'\n\nThe predicted Sales for Store', storeid, ' on ', date, ' is')
71     st.table(sales_df)
72     st.write(f'\n\nThe predicted Sales for Store {storeid} on {date} is {np.round(sales_prediction, 4)} €')
73
74     # Check if the User wants to see the Business problem that this project is trying to solve
75     project_overview = st.checkbox(
76         label = 'Show me the Business Problem that this Project addresses.',
77         value = False
78     )
79
80     # Display the Business problem that this project is trying to solve
81     if project_overview:
```

```

81 if project_overview:
82
83
84     # progress bar for display aesthetics
85     progress_bar = st.progress(0)
86
87     for percent_complete in range(100):
88         time.sleep(0.0000001)
89         progress_bar.progress( percent_complete + 1 )
90
91     st.markdown( """# **Rossmann Store Sales Prediction**
92     ## Introduction :
93     Predicting sales performance is one of the main challenges faced by every business. Sales forecasting which refers to
94     the process of estimating demand of products over specific set of time in future is important, as the demand for
95     products keeps changing from time to time and it is crucial for business firms to predict customer demands to offer the
96     right products at the right time. Sales forecasting also helps to maintain adequate inventory of products and thus,
97     improving their financial performance.
98
99     ## Problem Statement :
100    Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with
101    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.
102
103    ## Dataset used in building the model:
104    Data is taken from Kaggle from the following link:
105    https://www.kaggle.com/competitions/rossmann-store-sales/data
106    ## Business solution that this project delivers :
107    For each store, the daily sales predictions for the next six weeks."""
108 )

```

- 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

```

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
```

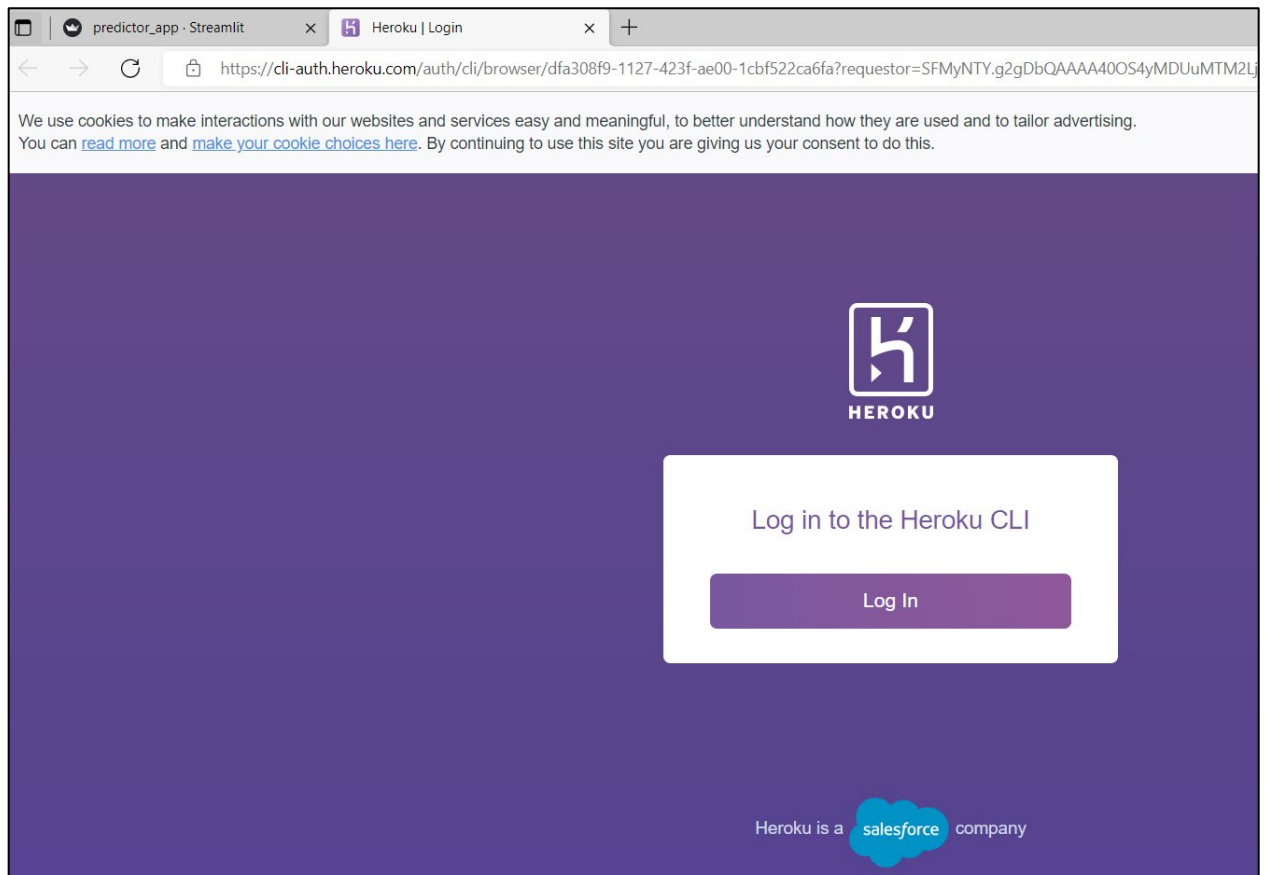
8) Create a Heroku account and then download the Heroku CLI.

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.



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.

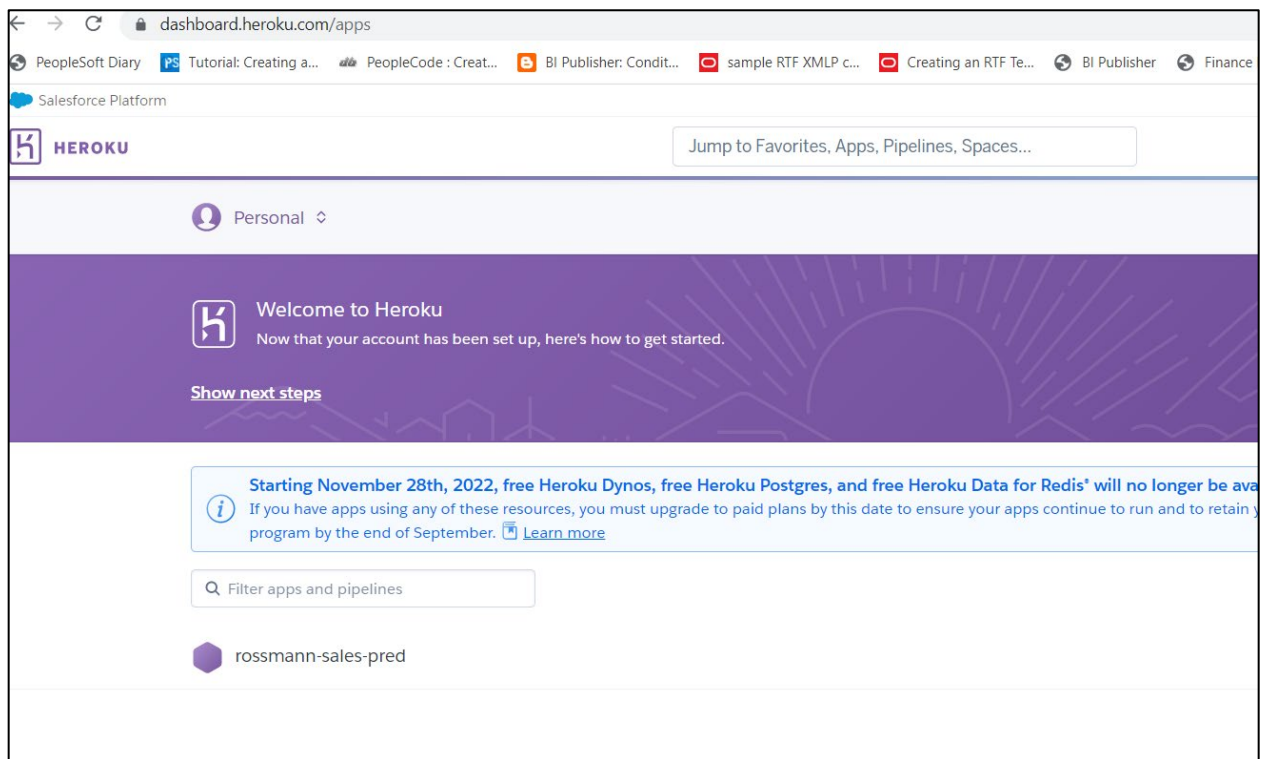
```
git add .
```

```
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.



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

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

A screenshot of the 'Rossmann Store Daily Sales Predictor' web application. The browser address bar shows 'rossmann-sales-pred.herokuapp.com'. The page has a white background with a large black title 'Rossmann Store Daily Sales Predictor'. Below the title, it says 'Forecast daily Sales for next six weeks :'. There are four input fields: 'Enter Date of Interest' with the value '2022/10/04', 'Enter Store Id' with the value '1', 'Will Store be running a Promo?' with an unchecked checkbox, and 'Will there be a State Holiday?' with a dropdown menu showing 'None'. Below these is another dropdown menu for 'Will there be a School Holiday?' showing 'No'. At the bottom, there is a 'Predict' button.

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.