Project Description :

In this problem we collecting data about who comes to your website and what they do when they get there

To Increase the website traffic we need to do:

- Social media Marketing
- Email marketing
- Guest Blogging
- Online Advertising
- Paid Advertising
- Optimize website Performance

Project Objective :

• The main objectives for website traffic analysis are essential for understanding and improving the website's performance, user experience, and overall online presence.

• These objectives guide the efforts and help to make data-driven decisions.

• The primary objective for many websites is to attract more visitors. This may include increasing organic search traffic, referral traffic, or social media traffic

Innovation:

• Voice Search Optimization: As voice assistants like Siri and Alexa become more common, optimizing in website for voice search and it can be a game-changer. Voice search often results in more conversational queries, so the content need to be change accordingly.

• Video Content: Video continues to grow in popularity. Consider creating engaging video content that not only lives on the website but also on platforms like YouTube. Video can help boost SEO and drive traffic from video-sharing platforms.

• Progressive Web Apps (PWAs): PWAs provide a more app-like experience on the web, offering faster loading times and offline functionality. Implementing a PWA can improve user engagement and retention.

• Personalization: Use of AI and machine learning algorithms to personalize content for each visitor based on their preferences and behavior. This can lead to higher engagement and conversions.

• Chatbots and AI Assistants: Implementing chatbots powered by AI to offer real-time assistance to website visitors. They can help answer questions, guide users, and collect valuable data for future improvements.

• User-Generated Content: Encourage users to create and share content related to website. This can include reviews, testimonials, and user-generated blog posts. User-generated content not only attracts traffic but also builds trust.

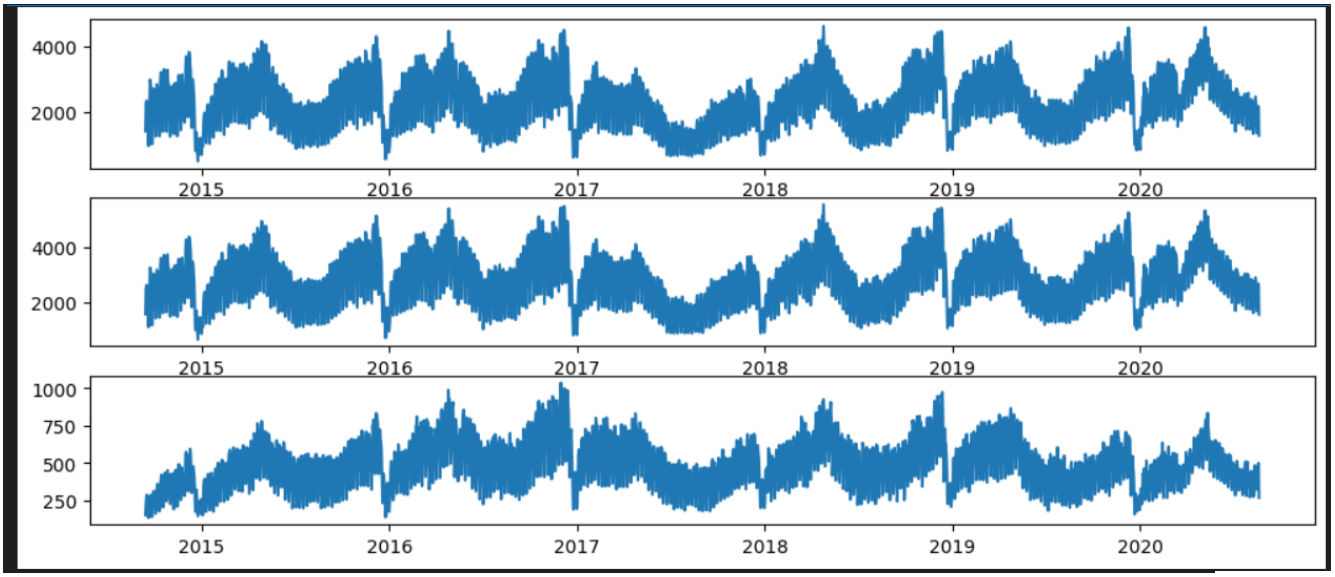
• Interactive Content: Create interactive content like quizzes, calculators, and surveys that engage users and encourage sharing. Interactive content can be highly shareable on social media, bringing in more traffic.

Project Devolopment :

```
import pandas as pd

whole_dataset.index = pd.to_datetime(whole_dataset.index)
whole_dataset
whole_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2167 entries, 2014-09-14 to 2020-08-19
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Row                2167 non-null   int64 
 1   Day                2167 non-null   object
 2   Day.Of.Week        2167 non-null   int64 
 3   Page.Loads         2167 non-null   int64 
 4   Unique.Visits      2167 non-null   int64 
 5   First.Time.Visits  2167 non-null   int64 
 6   Returning.Visits   2167 non-null   int64 
dtypes: int64(6), object(1)
memory usage: 135.4+ KB
```

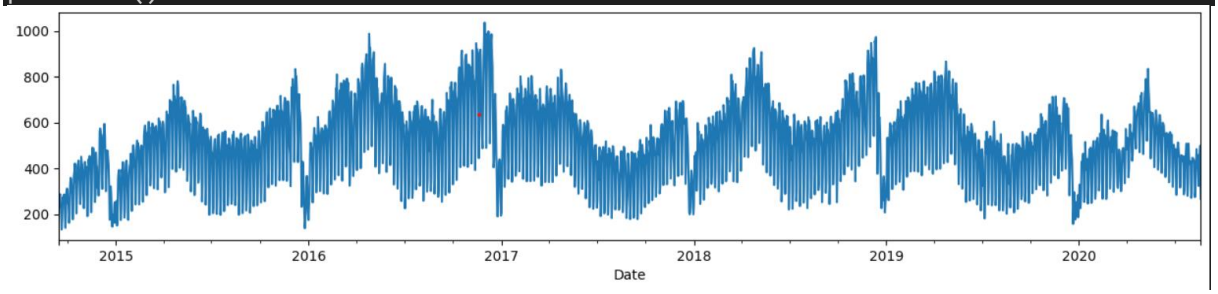
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, figsize=(12, 5))

axs[0].plot(whole_dataset['First.Time.Visits'])
axs[1].plot(whole_dataset['Unique.Visits'])
axs[2].plot(whole_dataset['Returning.Visits'])
plt.show()
```




```
target_column = whole_dataset['Returning.Visits']
target_column
```

```
Date
2014-09-14    152
2014-09-15    231
2014-09-16    278
2014-09-17    287
2014-09-18    236
             ...
2020-08-15    323
2020-08-16    351
2020-08-17    457
2020-08-18    499
2020-08-19    267
Name: Returning.Visits, Length: 2167, dtype: int64
```

```
target_column.plot(figsize=(15, 3))
plt.show()
```




```
len(target_column)
```

```
2167
```

```
test_dataset =
timeseries_dataset_from_array(target_column[TEST_DATA_BOUNDARY_INDEX -
WINDOW_SIZE:],
```

```
                                                target_column[TEST_DATA_BOUND
ARY_INDEX:],
                                                sequence_length=WINDOW_SIZE
                                            )
len(test_dataset), len(list(test_dataset.unbatch()))
```

```
(2, 217)
```

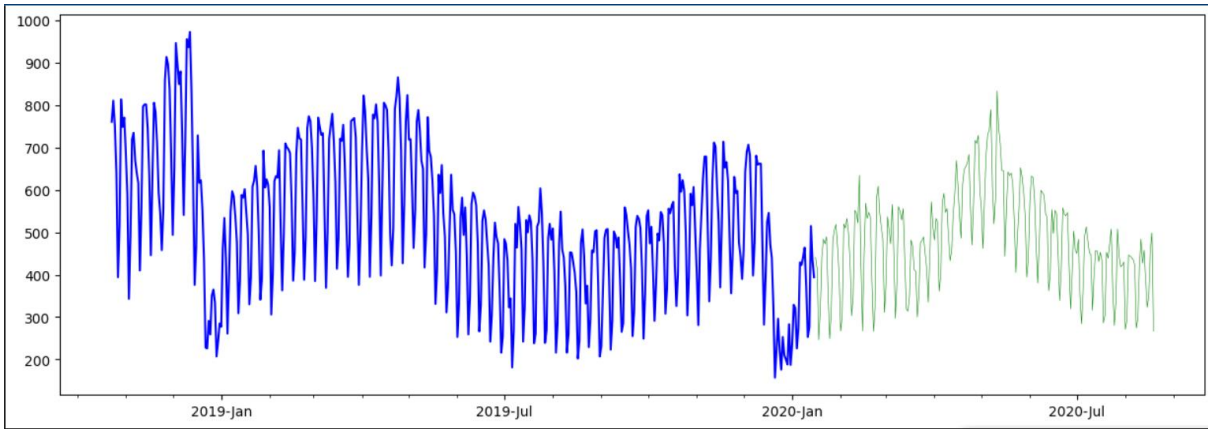
```
target_column[TEST_DATA_BOUNDARY_INDEX-10:TEST_DATA_BOUNDARY_INDEX+10].values,
list(test_dataset)[0][0][0].numpy(), list(test_dataset)[0][1][0].numpy()
```

```
(array([429, 423, 442, 464, 372, 253, 277, 515, 434, 394, 441, 413, 246,
        314, 443, 484, 473, 490, 353, 249]),
 array([515, 434, 394]),
 441)
```

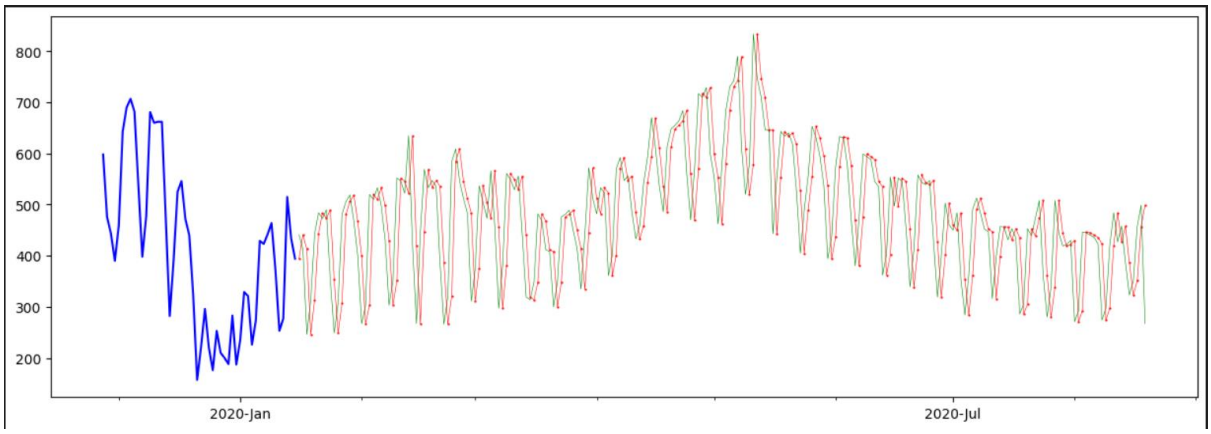
```
list(test_dataset)[-1][0][-1].numpy(), list(test_dataset)[-1][1][-1].numpy()
```

```
(array([351, 457, 499]), 267)
```

```
import numpy as np
import matplotlib.dates as mdates

def plot_time_series(predictions = None, start_index=1500):
    timesteps = pd.to_datetime(target_column.index)

    fig,ax = plt.subplots(1,figsize=(15,5))
    ax.xaxis.set_major_locator(mdates.MonthLocator(bymonth=(1, 7)))
    ax.xaxis.set_minor_locator(mdates.MonthLocator())
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%b'))
    plt.plot(timesteps[start_index:TEST_DATA_BOUNDARY_INDEX],
target_column[start_index:TEST_DATA_BOUNDARY_INDEX],
            color='blue')
    # Plot test dataset
    plt.plot(timesteps[TEST_DATA_BOUNDARY_INDEX:],
target_column[TEST_DATA_BOUNDARY_INDEX:],
             color='green', linewidth=0.4)

    if predictions is not None:
        pred_timesteps = timesteps[TEST_DATA_BOUNDARY_INDEX:]
        plt.plot(pred_timesteps, predictions, linewidth=0.4, color='red')
        plt.scatter(pred_timesteps, predictions, s=0.4, color='red')


plot_time_series()
```




```
plot_time_series(baseline_predictions.ravel(), start_index=1900)
```




```
y_true = target_column[TEST_DATA_BOUNDARY_INDEX : ]

len(y_true), y_true
```

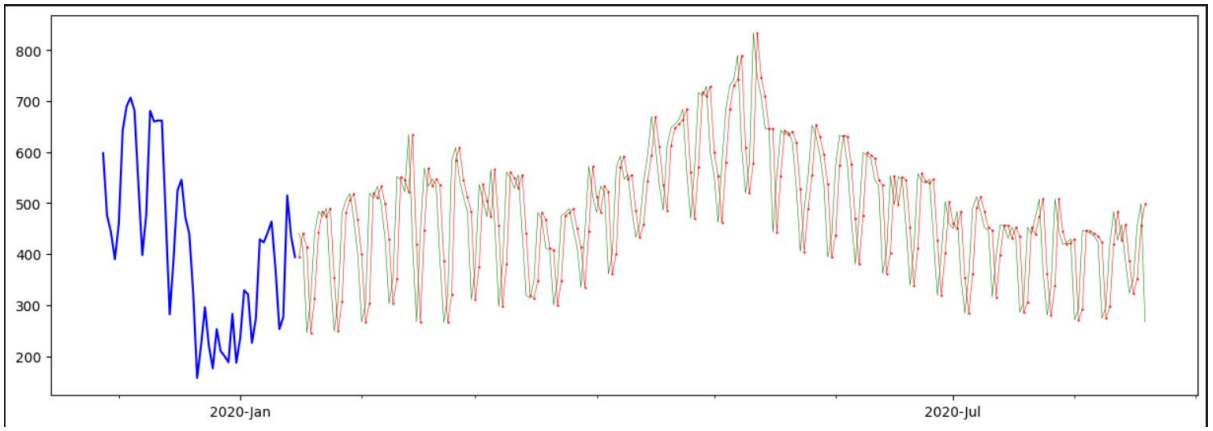
```
(217,
 Date
 2020-01-16    441
 2020-01-17    413
 2020-01-18    246
 2020-01-19    314
 2020-01-20    443
               ...
 2020-08-15    323
 2020-08-16    351
 2020-08-17    457
 2020-08-18    499
 2020-08-19    267
 Name: Returning.Visits, Length: 217, dtype: int64)
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error

def evaluate_predictions(y_true, y_preds):
    mae = mean_absolute_error(y_true, y_preds)
    mse = mean_squared_error(y_true, y_preds)
    rmse = np.sqrt(mse)
    mape = mean_absolute_percentage_error(y_true, y_preds)

    return {
        'mae': mae,
        'mse': mse,
        "rmse": rmse,
        "mape": mape
    }

evaluate_predictions(y_true, baseline_predictions)
```

```
{'mae': 72.19815668202764,
 'mse': 8508.622119815669,
 'rmse': 92.24219273096054,
 'mape': 0.16713927858326993}
```

```
MODEL_METRICS = pd.DataFrame(columns=['mae', 'mse', 'rmse', 'mape'])

def evaluate_model(model):
    predictions = model.predict(test_dataset, verbose=0)
    metrics = evaluate_predictions(y_true, predictions)

    MODEL_METRICS.loc[model.name] = metrics
    plot_time_series(predictions.ravel(), start_index=1900)
    return metrics
evaluate_model(baseline_model)
```

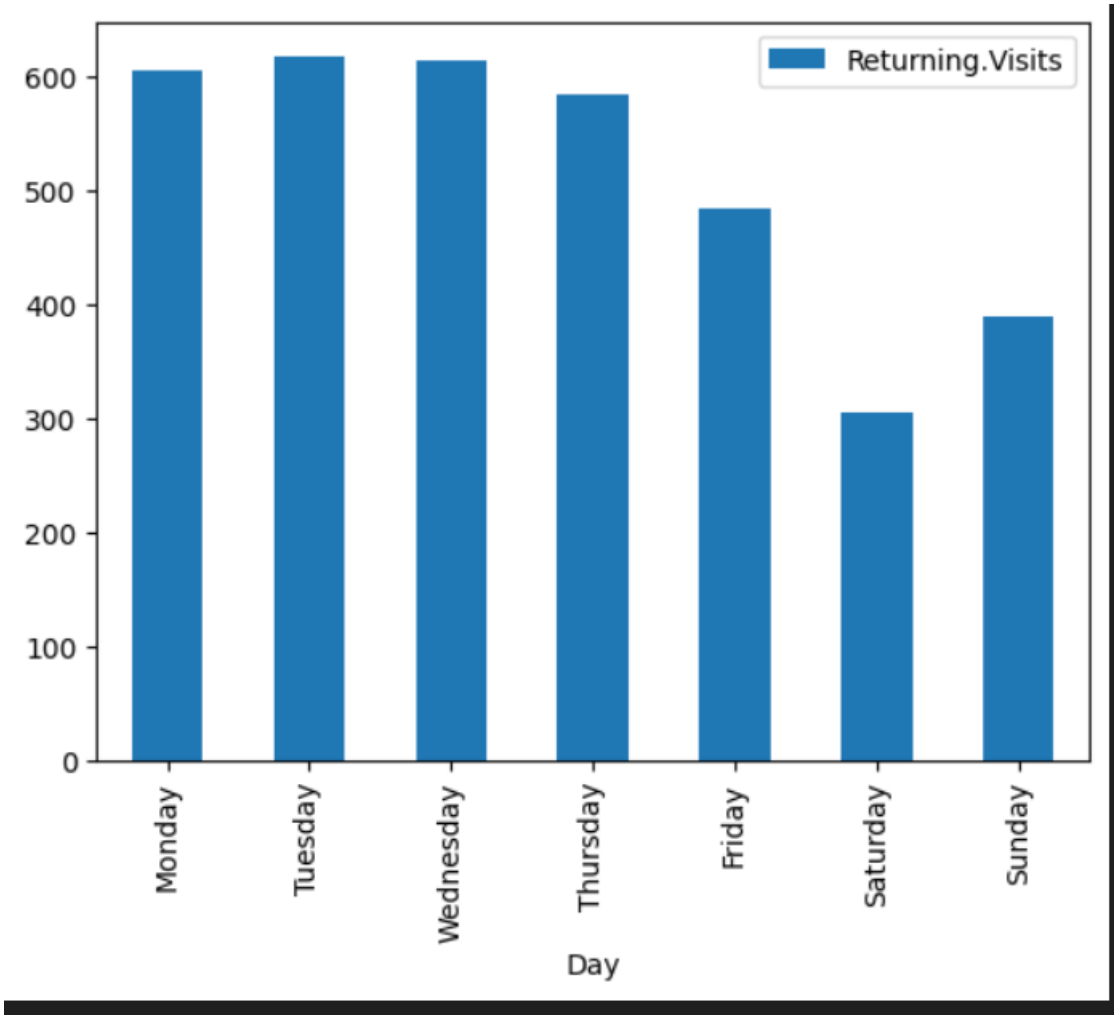
```
{'mae': 72.19815668202764,
 'mse': 8508.622119815669,
 'rmse': 92.24219273096054,
 'mape': 0.16713927858326993}
```



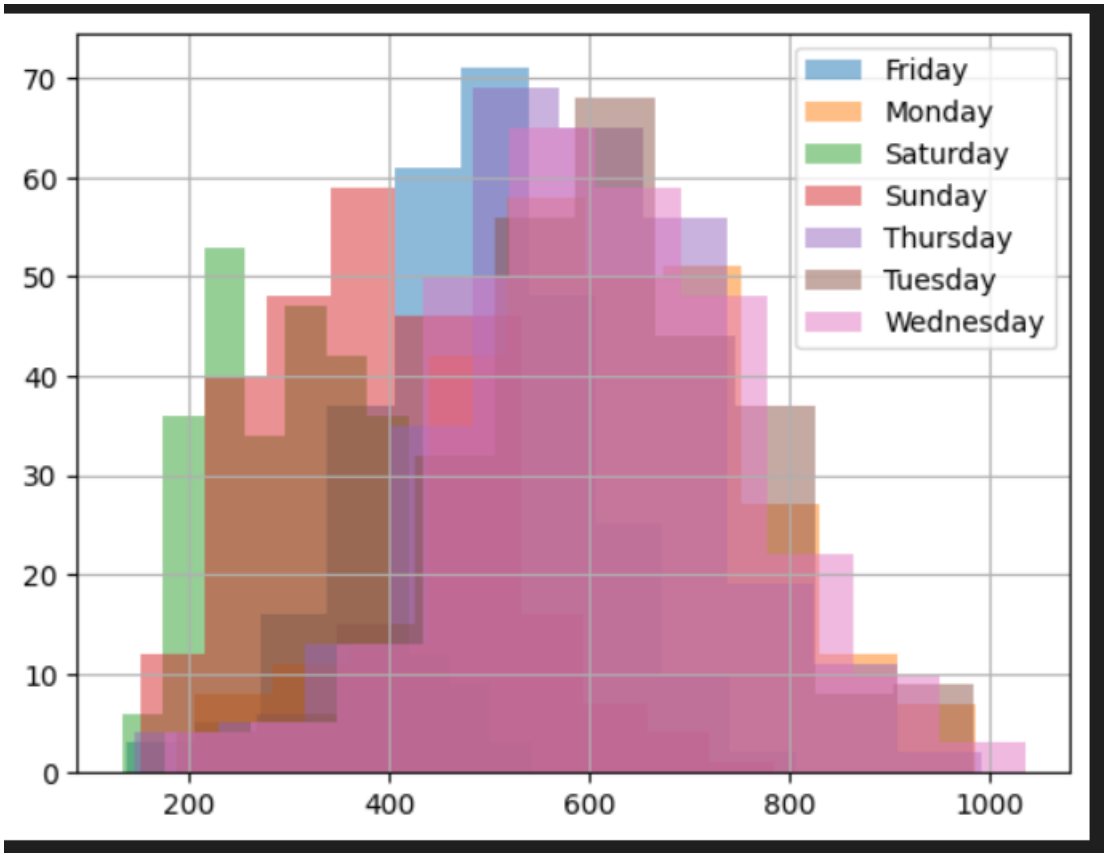

```
unbatched_train_dataset = whole_dataset[:TEST_DATA_BOUNDARY_INDEX + 1].copy()
unbatched_train_dataset
dataset_by_day = unbatched_train_dataset.groupby(by=['Day'])
dataset_by_day['Returning.Visits'].mean()
```

```
Day
Friday       484.697842
Monday       606.512545
Saturday     306.071942
Sunday       390.573477
Thursday     584.627240
Tuesday      617.888889
Wednesday    614.369176
Name: Returning.Visits, dtype: float64
```

```
DAYS_OF_WEEK = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
pd.DataFrame(dataset_by_day['Returning.Visits'].mean()).loc[DAYS_OF_WEEK].plot
(kind='bar')
```




```
dataset_by_day['Returning.Visits'].hist(legend=True, alpha=0.5)
plt.show()
```

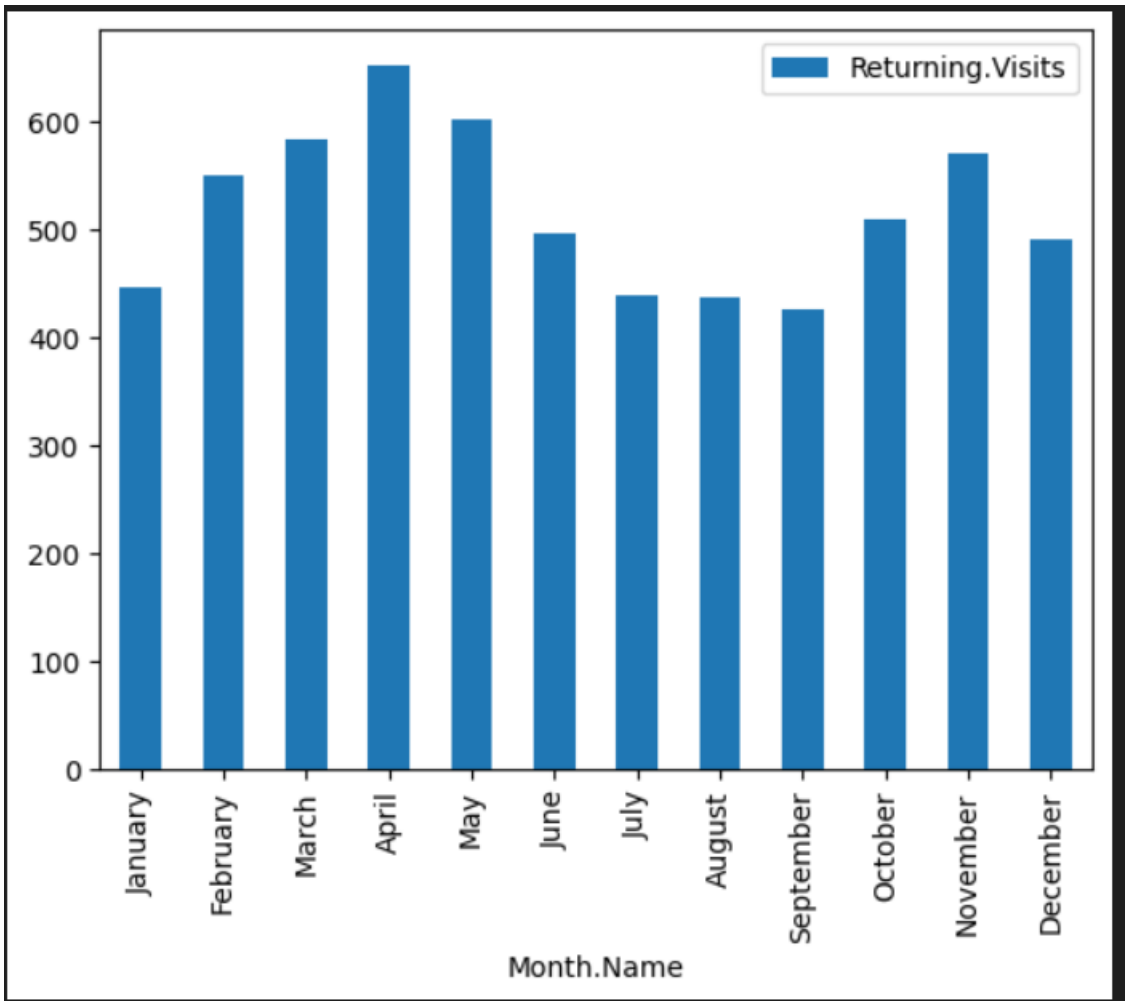



```
import calendar

train_dataset_with_months = unbatched_train_dataset.copy()
train_dataset_with_months['Month.Name'] =
pd.Series(train_dataset_with_months.index,
                                                    index=train_dataset_with_m
onths.index)\
                                            .apply(lambda x:
calendar.month_name[x.month])
train_dataset_with_months
MONTH_NAMES = list(calendar.month_name)[1:]
dataset_group_by_month = train_dataset_with_months.groupby(by='Month.Name')
dataset_group_by_month['Returning.Visits'].mean().loc[MONTH_NAMES]
```

```
Month.Name
January      445.976608
February     549.354610
March        583.470968
April        651.740000
May          601.135484
June         496.180000
July         438.509677
August       437.522581
September    426.173653
October      509.209677
November     569.716667
December     490.274194
```

```
Name: Returning.Visits, dtype: float64
```

```
pd.DataFrame(dataset_group_by_month['Returning.Visits'].mean()).loc[MONTH_NAMES].plot(kind='bar')
plt.show()
```




```
train_dataset2 = dataset2['Returning.Visits']
train_dataset2
```

```
Date
2014-09-17    287
2014-09-18    236
2014-09-19    241
2014-09-20    133
2014-09-21    175
              ...
2020-01-12    277
2020-01-13    515
2020-01-14    434
2020-01-15    394
2020-01-16    441
```

```
Name: Returning.Visits, Length: 1948, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
X_cat_encoder = OrdinalEncoder(categories = [DAYS_OF_WEEK, MONTH_NAMES])
X_cat_encoded = X_cat_encoder.fit_transform(dataset2_cat_features)
X_cat_encoded, X_cat_encoder.categories_
```

```
(array([[2., 8.],
        [3., 8.],
        [4., 8.],
        ...,
        [1., 0.],
        [2., 0.],
        [3., 0.]]),
 [array(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
         'Sunday'], dtype=object),
  array(['January', 'February', 'March', 'April', 'May', 'June', 'July',
         'August', 'September', 'October', 'November', 'December'],
        dtype=object)])
```

```
X_test_cat_input = test_dataset2[['Day', 'Month.Name']]
X_test_cat_input = X_cat_encoder.transform(X_test_cat_input)
X_test_cat_input.shape, X_test_cat_input[:5]
```

```
((217, 2),
 array([[3., 0.],
        [4., 0.],
        [5., 0.],
        [6., 0.],
        [0., 0.]]))
```

```
model_3_preds = model_3.predict([X_test_rv_history_input, X_test_cat_input])
model_3_preds[:15]
```

```
7/7 [==============================] - 0s 2ms/step
array([[302.65192],
       [400.0867 ],
       [343.56638],
       [247.99692],
       [432.88104],
       [465.76254],
       [441.44153],
       [399.31952],
       [418.44266],
       [275.72876],
       [302.09192],
       [408.75903],
       [484.41925],
       [458.57355],
       [451.74045]], dtype=float32)
```