**Website traffic analysis**

**Phase 3:** Development part 1

**Topic:**

Building the website traffic analysis using IBM Cognos for visualization define the objectives of the analysis and load website traffic data from the source shared.



**Introduction:**

* Web traffic analytics refers to collecting data about who comes to your website and what they do when they get there. That data is crucial to building effective [sales and marketing](https://www.leadfeeder.com/blog/sales-and-marketing-alignment/) strategies.
* Website traffic analysis using machine learning is a powerful approach to gain insights, make predictions, and optimize a website's performance. This technique leverages advanced algorithms to process and analyze vast amounts of web traffic data, providing valuable information that can inform decision-making and drive improvements in various aspects of a website's functionality and user experience.
* Loading and pre-processing a dataset for website traffic analysis involves several steps, including data collection, data cleaning, data transformation, and data exploration.

**Data set:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Row | Day | Day.Of.Week | Date | Page.Loads | Unique.Visits | First.Time.Visits | Returning.Visits |
| 1 | Sunday | 1 | 9/14/2014 | 2,146 | 1,582 | 1,430 | 152 |
| 2 | Monday | 2 | 9/15/2014 | 3,621 | 2,528 | 2,297 | 231 |
| 3 | Tuesday | 3 | 9/16/2014 | 3,698 | 2,630 | 2,352 | 278 |
| 4 | Wednesday | 4 | 9/17/2014 | 3,667 | 2,614 | 2,327 | 287 |
| 5 | Thursday | 5 | 9/18/2014 | 3,316 | 2,366 | 2,130 | 236 |
| 6 | Friday | 6 | 9/19/2014 | 2,815 | 1,863 | 1,622 | 241 |
| 7 | Saturday | 7 | 9/20/2014 | 1,658 | 1,118 | 985 | 133 |
| 8 | Sunday | 1 | 9/21/2014 | 2,288 | 1,656 | 1,481 | 175 |
| 9 | Monday | 2 | 9/22/2014 | 3,638 | 2,586 | 2,312 | 274 |

**Necessary steps:**

Import the libraries:

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

**program:**

**in [1]:**

import pandas as pd

FILE\_LOCATION = ('/kaggle/input/daily-website-visitors/daily-website-visitors.csv')

whole\_dataset =pd.read\_csv(FILE\_LOCATION,index\_col='Date',thousands=',')

whole\_dataset.index = pd.to\_datetime(whole\_dataset.index)

whole\_dataset

**out [1]:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Row | Day | Day.Of.week | Date | Page.Loads | Unique.Visits | First.Time.Visits | Returning.Visits |
| 1 | Sunday | 1 | 9/14/2014 | 2,146 | 1,582 | 1,430 | 152 |
| 2 | Monday | 2 | 9/15/2014 | 3,621 | 2,528 | 2,297 | 231 |
| 3 | Tuesday | 3 | 9/16/2014 | 3,698 | 2,630 | 2,352 | 278 |
| 4 | Wednesday | 4 | 9/17/2014 | 3,667 | 2,614 | 2,327 | 287 |
| 5 | Thursday | 5 | 9/18/2014 | 3,316 | 2,366 | 2,130 | 236 |
| 6 | Friday | 6 | 9/19/2014 | 2,815 | 1,863 | 1,622 | 241 |
| 7 | Saturday | 7 | 9/20/2014 | 1,658 | 1,118 | 985 | 133 |
| 8 | Sunday | 1 | 9/21/2014 | 2,288 | 1,656 | 1,481 | 175 |
| 9 | Monday | 2 | 9/22/2014 | 3,638 | 2,586 | 2,312 | 274 |

**In [2]:**

import matplotlib.pyplot as plt

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

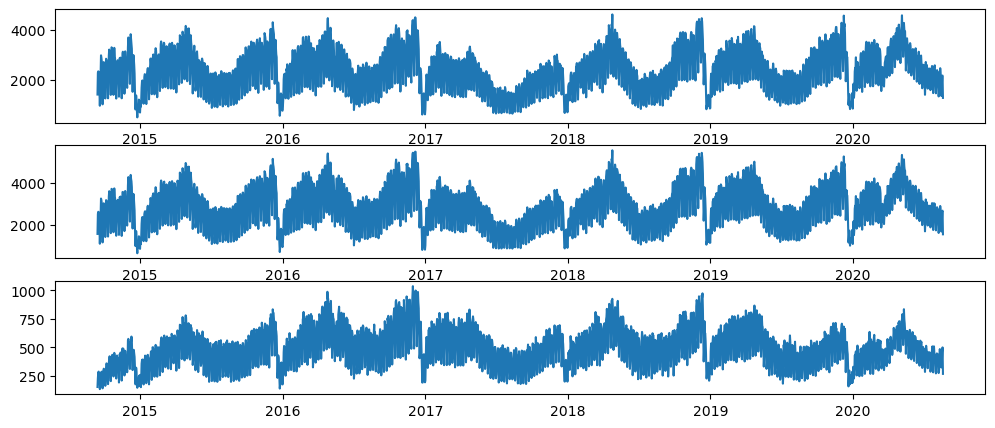
axs[0].plot(whole\_dataset['First.Time.Visits'])

axs[1].plot(whole\_dataset['Unique.Visits'])

axs[2].plot(whole\_dataset['Returning.Visits'])

plt.show()

**out [2]:**



**Pre-processing data:**

* Machine learning for website traffic analysis begins with the collection and pre-processing of data. Data sources may include web server logs, analytics tools, or user interactions. The data is cleaned, transformed, and made ready for analysis.

**In [3]:**

target\_column = whole\_dataset['Returning.Visits']

target\_column

**out [3]:**

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

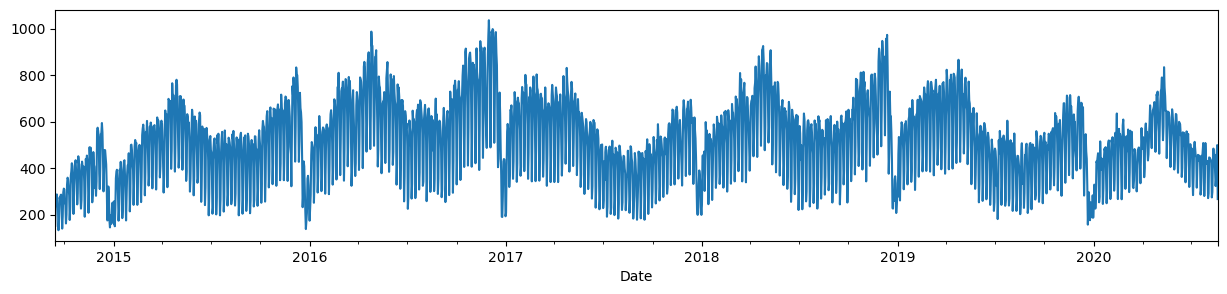
Name: Returning.Visits, Length: 2167, dtype: int64

**In [4]:**

target\_column.plot(figsize=(15, 3))

plt.show()

**out [4]:**



**In [5]:**

TEST\_DATA\_PERCENTAGE = 0.1

TEST\_DATA\_BOUNDARY\_INDEX = int((1 - TEST\_DATA\_PERCENTAGE) \* len(target\_column))

print(f"Train data:**\t**Returning Visits [:**{**TEST\_DATA\_BOUNDARY\_INDEX**}**] (**{**TEST\_DATA\_BOUNDARY\_INDEX + 1**}**)")

print(f"Test data:**\t**Returning Visits [**{**TEST\_DATA\_BOUNDARY\_INDEX**}**:] (**{**len(target\_column) - TEST\_DATA\_BOUNDARY\_INDEX**}**)")

print(f"**\n**Last target on train data: **{**target\_column[TEST\_DATA\_BOUNDARY\_INDEX]**}**")

**out [5]:**

Train data: Returning Visits [:1950] (1951)

Test data: Returning Visits [1950:] (217)

**In [6]:**

target\_column[TEST\_DATA\_BOUNDARY\_INDEX10:TEST\_DATA\_BOUNDARY\_INDEX+10].

values, (list(train\_dataset)[-1][0][-1].numpy(), list(train\_dataset)[-1][1][-1].numpy())

**out [6]:**

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

(array([277, 515, 434]), 394))

Plot the train and test datasets

**In [7]:**

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'))

*# Plot train dataset*

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:

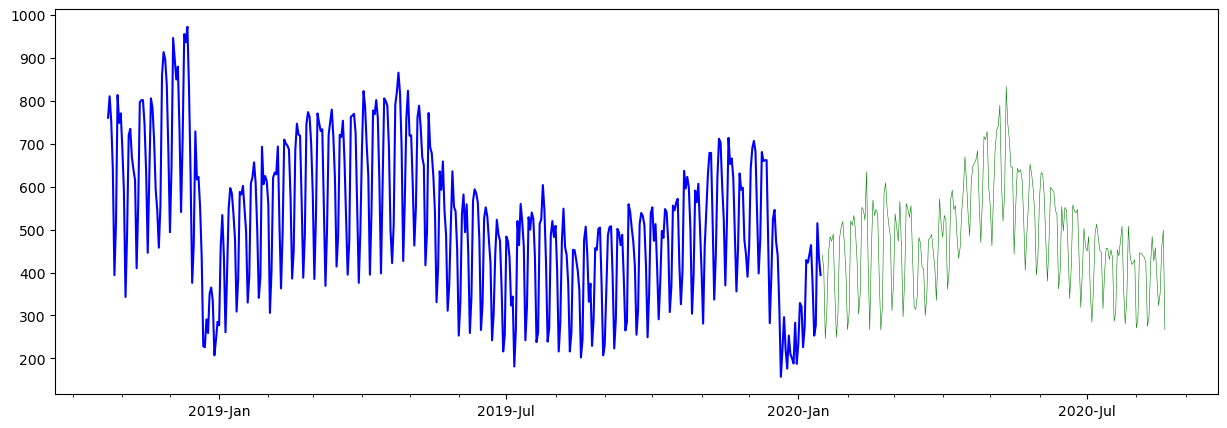
pred\_timesteps = timesteps[TEST\_DATA\_BOUNDARY\_INDEX:]

plt.plot(pred\_timesteps, predictions, linewidth=0.4, color='red')

plt.scatter(pred\_timesteps, predictions, s=0.4, color='red')

plot\_time\_series()

**out [7]:**



**In [8]:**

import tensorflow as tf

from tensorflow.keras.layers import Layer

from tensorflow.keras import Model

class **NaiveForecastLayer**(Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

def call(self, inputs):

result = inputs[:, -1]

return result[:, tf.newaxis]

baseline\_model = NaiveForecastLayer()

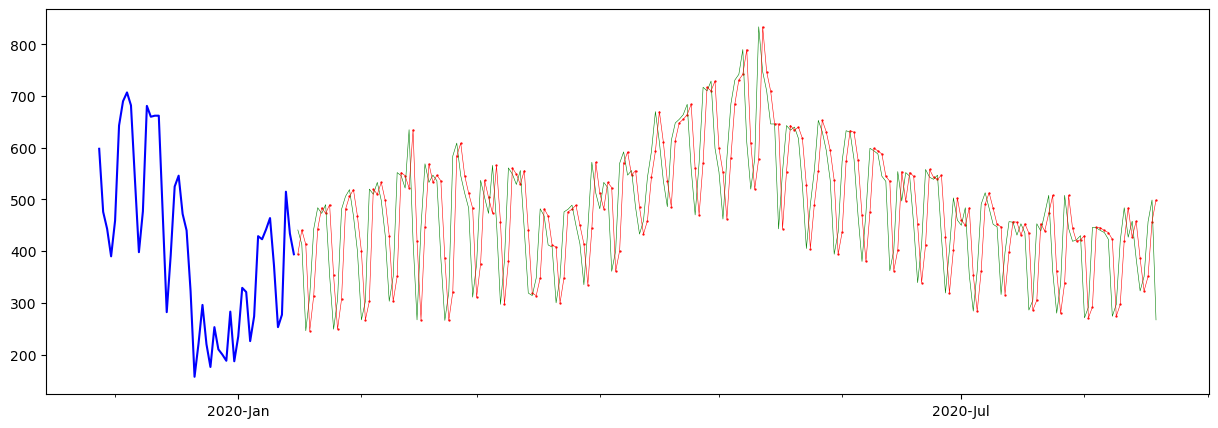
baseline\_model.\_name = 'model\_0'

baseline\_model.compile(metrics=[tf.keras.metrics.MeanAbsoluteError()])

baseline\_predictions = baseline\_model.predict(test\_dataset)

plot\_time\_series(baseline\_predictions.ravel(), start\_index=1900)

**out [8]:**



**In [9]:**

y\_true = target\_column[TEST\_DATA\_BOUNDARY\_INDEX : ]

len(y\_true), y\_true

**out [9]:**

(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)

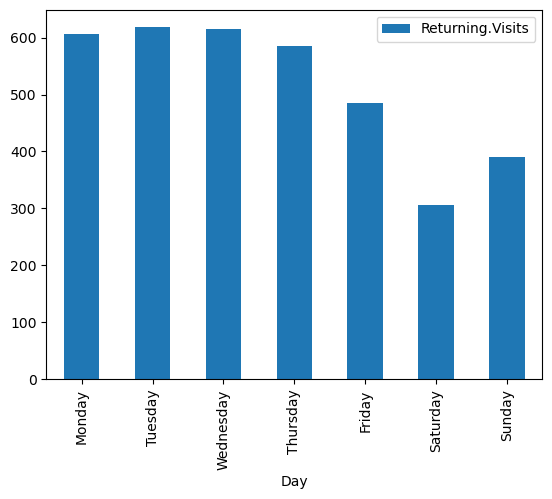
**In [10]:**

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')

**out [10]:**

<Axes: xlabel='Day'>

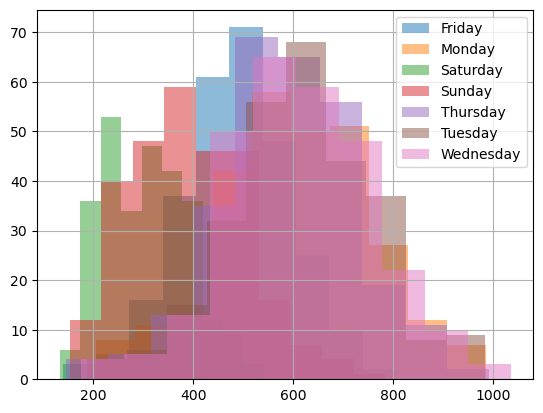


**In [11]:**

dataset\_by\_day['Returning.Visits'].hist(legend=True, alpha=0.5)

plt.show()

**out [11]:**

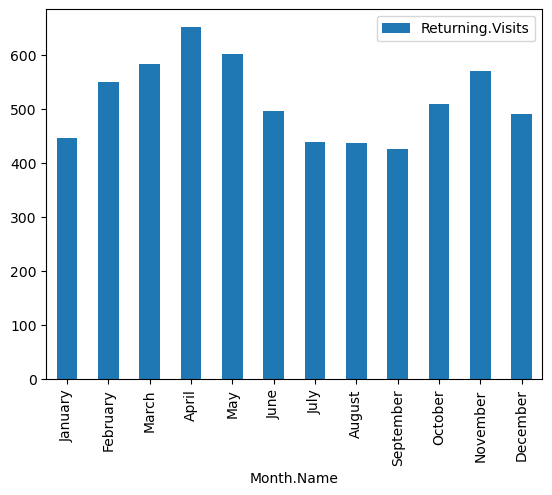


**In [12]:**

pd.DataFrame(dataset\_group\_by\_month['Returning.Visits'].mean()).loc[MONTH\_NAMES].plot(kind='bar')

plt.show()

**out [12]:**



**In [13]:**

from tensorflow.data import Dataset

model3\_history = model\_3.fit(x=[dataset2\_rv\_history\_features, X\_cat\_encoded], y=train\_dataset2, epochs=5)

pd.DataFrame(model3\_history.history).plot()

**out [13]:**

Epoch 1/5

61/61 [==============================] - 3s 7ms/step - loss: 232.3113

Epoch 2/5

61/61 [==============================] - 0s 7ms/step - loss: 105.8665

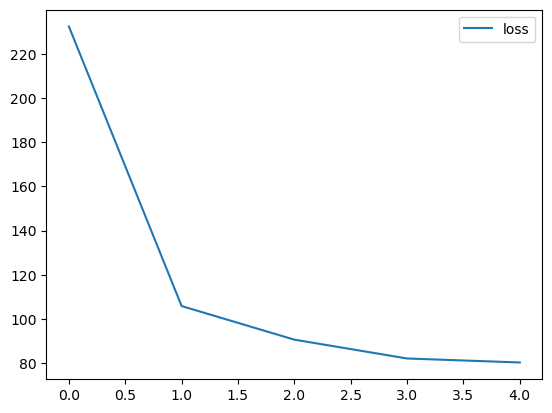
Epoch 3/5

61/61 [==============================] - 0s 7ms/step - loss: 90.6746

Epoch 4/5

61/61 [==============================] - 0s 7ms/step - loss: 82.1568

Epoch 5/561/61 [==============================]- 0s 7ms/step - loss: 80.3541



<Axes: >

**In [14]:**

y\_dataset = test\_dataset2['Returning.Visits']

y\_dataset

**out [14]:**

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

**In [15]:**

def evaluate\_model\_predictions(y\_true, predictions, model\_name):

metrics = evaluate\_predictions(y\_true, predictions)

MODEL\_METRICS.loc[model\_name] = metrics

plot\_time\_series(predictions.ravel(), start\_index=1900)

return metrics

evaluate\_model\_predictions(y\_dataset, model\_3\_preds, 'model\_3 (multi-input)')

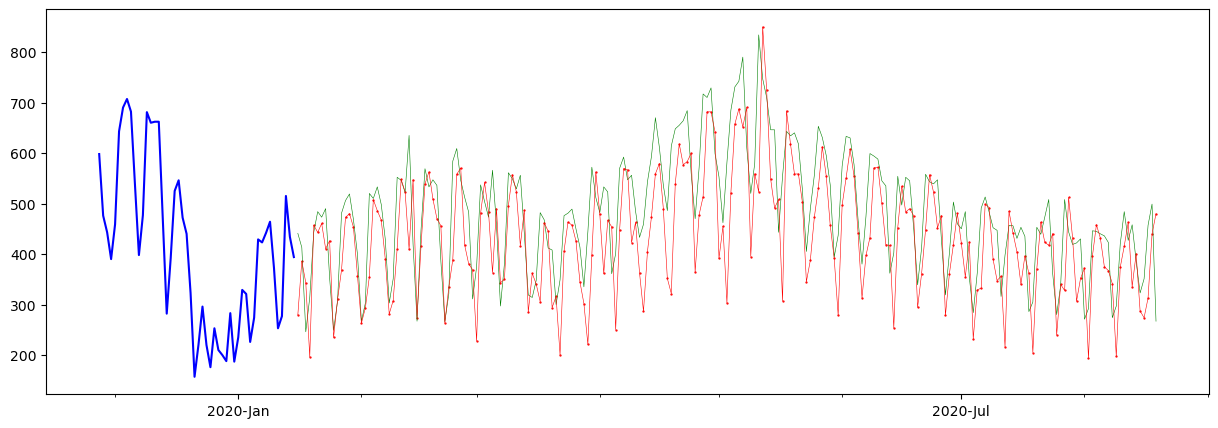
**out [15]:**

{'mae': 72.46600053497174,

'mse': 8797.218902262757,

'rmse': 93.79349072437147,

'mape': 0.15300125677432075}



**Data Exploration**:

* Visualize the data using tools like Matplotlib, Seaborn, or Plotly to gain insights into patterns and trends.
* Calculate basic statistics, such as means, medians, and standard deviations, to better understand your data.
* Perform exploratory data analysis to identify interesting patterns or anomalies in the website traffic data.

**Feature Engineering:**

* Create additional features that can help in your analysis. For instance, you can derive features like day of the week, time of day, or even holidays if relevant to your analysis.

**Data Splitting**:

* Split your dataset into training and testing subsets if you plan to build predictive models. This is important to evaluate the model's performance.

**Data Modeling:**

* If your goal is predictive analytics, you can build machine learning models to predict website traffic patterns or user behavior. Common techniques include regression, classification, and time series forecasting.

**Conclusion:**

* In conclusion, website traffic analysis using machine learning is a dynamic and evolving field with numerous applications. It empowers website owners, marketers, and IT teams to make data-driven decisions that enhance user experience, increase conversion rates, and ultimately drive the success of online platforms. As the volume of data continues to grow, machine learning will play an increasingly vital role in harnessing its full potential for website optimization.