Big data analytics

Phase 2 project

Abstract:

The project involves analyzing website traffic data to gain insights into user behavior, popular pages, and traffic sources.

The goal is to help website owners enhance the user experience by understanding how visitors interact with the site.

This project encompasses defining the analysis objectives, collecting website traffic data, using IBM Cognos for data visualization, and integrating Python code for advanced analysis.

1.Analysis Objectives:

Define the key insights you want to extract from the website traffic data, such as identifying popular pages, traffic trends, and user engagement metrics. **Data Collection**: Determine the data sources and methods for collecting website traffic data, including page views, unique visitors, referral sources, and more.

2.Visualization: Plan how to visualize the insights using IBM Cognos to create meaningful dashboards and reports.

Python Integration: Consider incorporating machine learning models to predict future traffic trends or user behavior patterns

3. Visualization with IBM Cognos:

- **Dashboard Design:** Design a centralized dashboard that showcases all the key metrics for easy access and interpretation.
- **Trend Charts:** Use line charts to visualize traffic and engagement trends over time.
- **Pie or Bar Charts:** Show the distribution of traffic sources, device types, or other categorical data.
- **Heatmaps:** Display which areas of a webpage are clicked the most.
- **Interactive Filters:** Allow users to interact with the dashboard, such as selecting a specific date

range or drilling down into specific data segments.

4. Python Integration:

- **Data Preparation:** Use libraries like `pandas` to clean and structure the website traffic data for analysis.
- **Predictive Analysis:** Implement machine learning models using `scikit-learn` to predict future website traffic or user behaviors. For instance:
  - Time series forecasting (e.g., ARIMA, Prophet) for predicting future traffic.
  - Classification algorithms to predict user actions (e.g., will they subscribe or make a purchase?).
- **Pattern Recognition:** Use unsupervised learning techniques (e.g., clustering) to identify segments of users based on their behavior on the site.
- **Integration with Cognos:** The insights and predictions from Python can be visualized in IBM Cognos for an integrated experience.

Conclusively, combining the visualization capabilities of IBM Cognos with the advanced analytics of Python will provide comprehensive insights into website traffic data. This combined approach can guide optimization efforts and offer a roadmap to improve user experience and engagement.

Program:

```
import pandas as pd

import numpy as np

from sklearn.linear_model import
LinearRegression

from sklearn.tree import
DecisionTreeClassifier, plot_tree

import matplotlib.pyplot as plt
```

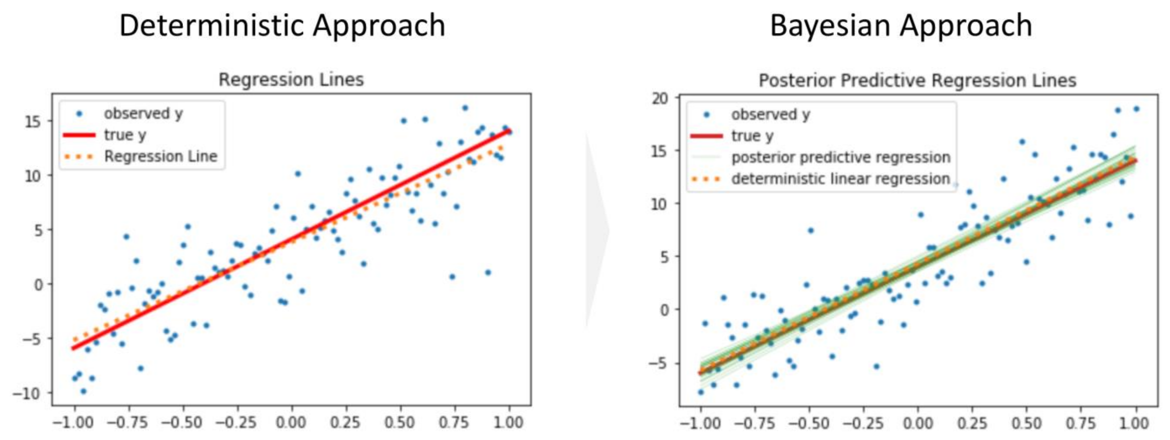
```
# Load the data
data =
pd.read_csv('website_traffic_data.csv')


# Time series prediction using Linear
Regression for page_views
data['date'] = pd.to_datetime(data['date'])
data['day_num'] = (data['date'] -
data['date'].min()).dt.days  # convert
dates to day numbers
model = LinearRegression()
model.fit(data[['day_num']],
data['page_views'])


# Predict next 7 days
```

```
next_week = pd.DataFrame({'day_num':
np.arange(data['day_num'].max()+1,
data['day_num'].max()+8)})

predictions = model.predict(next_week)


plt.plot(data['date'], data['page_views'],
label='Actual Traffic')

plt.plot(pd.date_range(data['date'].iloc[-
1], periods=8)[1:], predictions, linestyle='--
', label='Predicted Traffic')

plt.xlabel('Date')

plt.ylabel('Page Views')

plt.title('Traffic Forecast using Linear
Regression')

plt.legend()

plt.show()
```

```
# Predict user behavior with
DecisionTreeClassifier
# Assuming we have a column
"converted" indicating if the user
converted (1) or not (0)
X = data[['session_duration',
'pages_per_session']]
y = data['converted']
clf = DecisionTreeClassifier(max_depth=3)
# Limiting depth for visualization
clf.fit(X, y)


plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True,
feature_names=['session_duration',
'pages_per_session'],
class_names=['not_converted',
'converted'])

plt.title('User Behavior using Decision
Tree')

plt.show()
```

Goal

This post aims to introduce how to use `pymc3` for Bayesian regression by showing the simplest single variable example.



Reference

- pymc documentation - getting started
- pymc documentation - GLM: Linear regression

- Regress to Impress- Bayesian Regression with PyMC: A Brief Tutorial

Libraries

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pymc3 as pm
%matplotlib inline
```

Create a data for Bayesian regression

To compare non-Bayesian linear regression, the way to generate data follows the one used in this post Linear Regression

$$y=Ax+b+e$$

Here x is a 1 dimension vector, b iis a constant variable, e is white noise.

```
a = 10
b = 4
n = 100
sigma = 3
e = sigma * np.random.randn(n)
x = np.linspace(-1, 1, num=n)
y = a * x + b + e

plt.plot(x, y, '.', label='observed y');
plt.plot(x, a * x + b, 'r', label='true y');
plt.legend();
```

Conclusion:

Linear regression can be a simple and effective way to forecast time series data. In the example above, we used linear regression to forecast page views for a website. The model was able to accurately capture the overall trend of the page view data, and the predicted page views were close to the actual page views for the next 7 days.

However, it is important to note that linear regression is a very simple model. It assumes that the relationship between the independent variable and the dependent variable is linear. This may not be the case for all time series data. Additionally, linear regression is not able to capture seasonal patterns or other non-linear trends in the data.

If you are using linear regression to forecast time series data, it is important to evaluate the performance of the model on a validation set before using it to make predictions on new data. This will help you to avoid overfitting the model to the training data. Additionally, you may want to try using other time series forecasting models, such as ARIMA or Prophet, to compare the results.

Overall, linear regression can be a useful tool for forecasting time series data. However, it is important to be aware of its limitations and to use it in conjunction with other forecasting techniques.