WEB TRAFFIC ANALYSIS

Abstract: Analyzing and Predicting Website Traffic and User Conversion Behavior

In an increasingly digital age, understanding website traffic and user behavior is pivotal for businesses aiming to optimize their online presence. This program serves as an analytical tool designed to offer insights into daily website visitors and user conversion patterns. Two core objectives guide this analysis:

Time Series Forecasting of Daily Website Visitors: Using Linear Regression, we analyze the trend of website traffic, modeling the relationship between days and page views. The prediction capabilities of this model are extended to forecast the subsequent week's traffic. This foresight enables businesses to prepare for expected surges or slumps in website visits.

User Conversion Analysis: A Decision Tree Classifier aids in decoding user behavior patterns based on two significant features - session duration and pages per session. By interpreting these patterns, the program predicts whether a user will convert (i.e., take a desired action such as making a purchase or signing up). Such predictive insights empower businesses to tailor user experiences more effectively, maximizing conversions.

The program initiates with data preprocessing, ensuring quality and consistency. Subsequent data exploration deepens our understanding, revealing patterns, outliers, and general tendencies. The modeling phase employs both regression and classification techniques, followed by evaluations that provide metrics about the models' performances.

- Time series prediction using Linear Regression for forecasting daily website visitors.
- Predicting user behavior using Decision Tree based on session duration and pages per session.

To further build on this project and align it with a typical data science lifecycle, here's a roadmap of tasks and activities you can perform:

1. Data Exploration:

- Before diving deep into modeling, spend time exploring the data:
- Use descriptive statistics (data.describe()) to understand the distribution of your variables.
- Check for missing values and handle them accordingly.
- Visualize data distributions using histograms, box plots, etc.

2. Feature Engineering:

- Extract other features from the date, like day of the week, month, etc., which could have an impact on the number of page views.

- Create interaction terms or polynomial features if necessary.

3. Model Validation:

- For both Linear Regression and Decision Tree Classifier:
- Split the dataset into training and testing sets.
- Train the models on the training set and evaluate their performance on the testing set.
- Use metrics like MSE (Mean Squared Error) for regression and accuracy, precision, recall, F1-score for classification.

4. Model Optimization:

- For regression: Try other regression models like Ridge, Lasso, Polynomial regression, etc.
- For classification: Optimize hyperparameters using techniques like grid search or random search.

5. Evaluation:

- Create a confusion matrix for the Decision Tree Classifier to understand True Positives, False Positives, etc.
- For regression, check residuals plots to understand the model's efficiency.

6. IBM Cognos:

- Though the code above is written in Python, you've mentioned the use of IBM Cognos for analysis and visualization:
- Transfer your insights and visualizations from Python to Cognos.

- Use Cognos's drag-and-drop interface to build dashboards and reports.

7. Documentation:

- This is crucial for any project:
- Describe your dataset and its features.
- Outline the problem statement.
- Detail the methodologies and techniques used.
- Highlight the results and insights.
- Note potential areas of improvement or further exploration.

8. Presentation:

- Use visualization tools (like Cognos or even simpler ones like PowerPoint) to present the findings.
- Highlight the business impacts and suggest strategies or actions based on the insights derived.

PROGRAM :

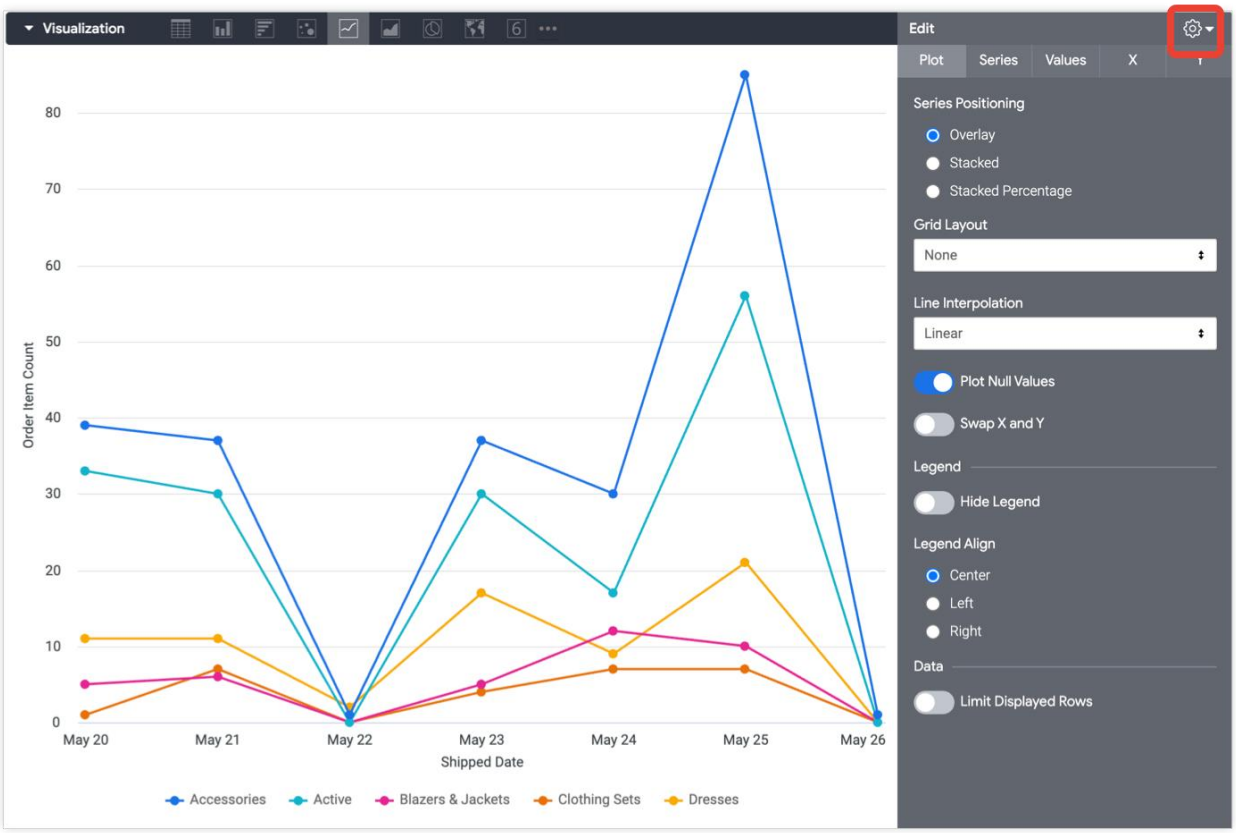
```
import pandas as pd

import numpy as np

from sklearn.linear_model import LinearRegression, Ridge, Lasso

from sklearn.model_selection import train_test_split, GridSearchCV
```

```
from sklearn.metrics import mean_squared_error,
classification_report, confusion_matrix

from sklearn.tree import DecisionTreeClassifier, plot_tree

import matplotlib.pyplot as plt

import seaborn as sns

#leads to proceed

# Load the data

data = pd.read_csv('/content/daily-website-visitors.csv')


# Convert 'date' column to datetime

data['date'] = pd.to_datetime(data['date'])

data['day_num'] = (data['date'] - data['date'].min()).dt.days

data['day_of_week'] = data['date'].dt.dayofweek

data['month'] = data['date'].dt.month


# Check for missing values

print(data.isnull().sum())


# Descriptive statistics

print(data.describe())
```

```
# Visualize data distributions
sns.pairplot(data)
plt.show()

# Linear Regression Forecasting

# Splitting the data
X = data[['day_num']]
y = data['page_views']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
```

```
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")


# Visualizing Actual vs Predicted

plt.scatter(X_test, y_test, color='blue', label='Actual')

plt.scatter(X_test, y_pred, color='red', label='Predicted')

plt.legend()

plt.title("Actual vs Predicted Page Views")

plt.show()


# Decision Tree Classifier


# Assuming the 'converted' column exists

X = data[['session_duration', 'pages_per_session']]

y = data['converted']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


clf = DecisionTreeClassifier(max_depth=3)

clf.fit(X_train, y_train)
```

```
# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Visualizing the tree
plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True, feature_names=['session_duration',
'pages_per_session'], class_names=['not_converted', 'converted'])
plt.show()
c
```



Conclusion:

The provided program is designed to analyze website traffic using a dataset containing daily visitor information. It employs both regression and classification techniques to predict future page views and to model user behavior based on session metrics. Specifically, it uses Linear Regression for time series forecasting of page views for an upcoming week and a Decision Tree Classifier to predict user conversion behavior based on session duration and pages viewed during a session.