project.py

```python
from fpdf import FPDF

# Initialize PDF object
pdf = FPDF()
pdf.set_auto_page_break(auto=True, margin=15)
pdf.add_page()

# Title
pdf.set_font("Arial", style='B', size=16)
pdf.cell(200, 10, txt="Web Traffic Analysis for Predicting Website Growth", ln=True, align='C')

# Line break
pdf.ln(10)

# Introduction
pdf.set_font("Arial", size=12)
intro_text = """This Python script demonstrates a basic approach to analyzing web traffic data and predicting future website growth.
The script covers data preprocessing, feature engineering, building a simple predictive model (Linear Regression), and forecasting future traffic."""
pdf.multi_cell(0, 10, intro_text)

# Line break
pdf.ln(10)

# Code Section 1: Import Libraries
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="1. Import Necessary Libraries", ln=True)
pdf.set_font("Arial", size=10)
code1 = """
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
"""
pdf.multi_cell(0, 10, code1)

# Line break
pdf.ln(10)

# Code Section 2: Load and Preprocess Data
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="2. Load and Preprocess Data", ln=True)
pdf.set_font("Arial", size=10)
code2 = """
# Load the data (you should replace 'traffic_data.csv' with your actual data file)
df = pd.read_csv('traffic_data.csv')
```

```python

# Check the first few rows of the data
print(df.head())

# Convert the date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Set date as index (optional, depending on your analysis needs)
df.set_index('date', inplace=True)

# Plot the web traffic over time
plt.figure(figsize=(10,6))
plt.plot(df.index, df['page_views'], label='Page Views')
plt.title('Website Traffic Over Time')
plt.xlabel('Date')
plt.ylabel('Page Views')
plt.grid(True)
plt.legend()
plt.show()
"""
pdf.multi_cell(0, 10, code2)

# Line break
pdf.ln(10)

# Code Section 3: Feature Engineering
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="3. Feature Engineering", ln=True)
pdf.set_font("Arial", size=10)
code3 = """
# Feature engineering (day of the week, month, etc.)
df['day_of_week'] = df.index.dayofweek
df['month'] = df.index.month
df['year'] = df.index.year

# Display the updated dataframe
print(df.head())
"""
pdf.multi_cell(0, 10, code3)

# Line break
pdf.ln(10)

# Code Section 4: Train/Test Split
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="4. Train/Test Split", ln=True)
pdf.set_font("Arial", size=10)
code4 = """
# Let's use the past data to predict future growth (train/test split)
X = df[['day_of_week', 'month', 'year']]  # Features
y = df['page_views']  # Target variable (traffic count)

# Split into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

```python

# Feature scaling (optional, especially if using algorithms like SVM, neural networks)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
"""
pdf.multi_cell(0, 10, code4)

# Line break
pdf.ln(10)

# Code Section 5: Linear Regression Model
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="5. Linear Regression Model", ln=True)
pdf.set_font("Arial", size=10)
code5 = """
# Initialize the Linear Regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R^2 Score: {r2}")

# Plot the actual vs predicted values
plt.figure(figsize=(10,6))
plt.plot(y_test.index, y_test, label='Actual', color='blue')
plt.plot(y_test.index, y_pred, label='Predicted', color='red', linestyle='dashed')
plt.title('Actual vs Predicted Web Traffic')
plt.xlabel('Date')
plt.ylabel('Page Views')
plt.legend()
plt.grid(True)
plt.show()
"""
pdf.multi_cell(0, 10, code5)

# Line break
pdf.ln(10)

# Code Section 6: Future Prediction (Forecasting)
pdf.set_font("Arial", style='B', size=12)
pdf.cell(200, 10, txt="6. Future Prediction (Forecasting)", ln=True)
pdf.set_font("Arial", size=10)
```

```python
code6 = """
# Create a DataFrame for future dates (e.g., next 30 days)
future_dates = pd.date_range(df.index[-1] + pd.Timedelta(days=1), periods=30, freq='D')

# Generate features for the future dates
future_df = pd.DataFrame(index=future_dates)
future_df['day_of_week'] = future_df.index.dayofweek
future_df['month'] = future_df.index.month
future_df['year'] = future_df.index.year

# Scale the future data using the previously fitted scaler
future_scaled = scaler.transform(future_df)

# Predict future page views using the model
future_predictions = model.predict(future_scaled)

# Visualize the future predictions
plt.figure(figsize=(10,6))
plt.plot(df.index, df['page_views'], label='Historical Data', color='blue')
plt.plot(future_df.index, future_predictions, label='Predicted Future Traffic', color='red', linestyle='dashed')
plt.title('Web Traffic Prediction (Next 30 Days)')
plt.xlabel('Date')
plt.ylabel('Page Views')
plt.legend()
plt.grid(True)
plt.show()

# Output the predictions for the next 30 days
future_df['predicted_page_views'] = future_predictions
print(future_df)
"""
pdf.multi_cell(0, 10, code6)

# Save the PDF
pdf.output('web_traffic_analysis.pdf')

print("PDF generated successfully!")

```