DSC.630_Week8_SaidMoussadeq

July 28, 2024

DSC 630

Week 8 Assignment

Said Moussadeq

Change Working Directory and Import Libraries

[34]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import register_matplotlib_converters
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
from dateutil import parser
import warnings


# Suppress specific warnings
warnings.filterwarnings("ignore", category=FutureWarning, module='seaborn')

# Register converters to handle date formats
register_matplotlib_converters()

# Load the necessary libraries for plotting and analysis
sns.set(style="darkgrid")
```

Load and Transform Data

[35]:
```
# Import the CSV file as a dataframe
sales = pd.read_csv("us_retail_sales.csv")

# Melt the data to take a matrix and change it into a long dataframe
sales_long = sales.melt(id_vars=["YEAR"], var_name="MONTH", value_name="SALES")

# Convert the MONTH and YEAR columns to a single date column
```

```
sales_long['DATE'] = sales_long.apply(lambda row: parser.parse(f'{row["YEAR"]}␣
 ↪{row["MONTH"]}'), axis=1)

# Sort the data by date
sales_long = sales_long.sort_values(by='DATE')

# Display the first few rows to confirm
sales_long.head()
```

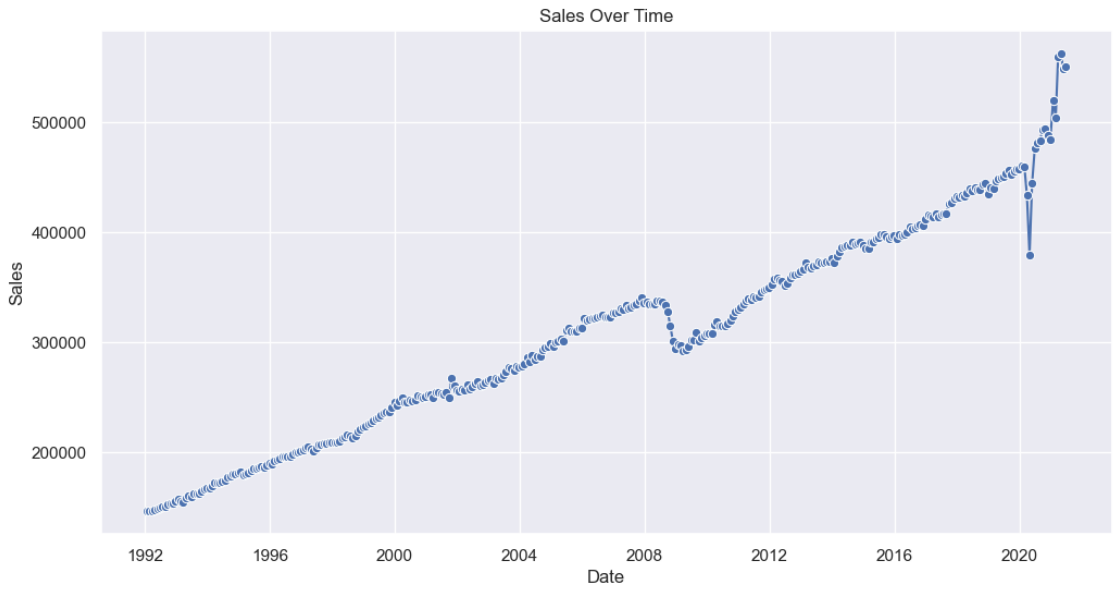
[35]:

	YEAR	MONTH	SALES	DATE
0	1992	JAN	146925.0	1992-01-28
30	1992	FEB	147223.0	1992-02-28
60	1992	MAR	146805.0	1992-03-28
90	1992	APR	148032.0	1992-04-28
120	1992	MAY	149010.0	1992-05-28

Plot the Data

[36]:
```
# Plot the data with proper labeling
plt.figure(figsize=(12, 6))
sns.lineplot(data=sales_long, x='DATE', y='SALES', marker='o')
plt.title("Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.show()

# Observations:
# - 2007-2009: Deviation from the smooth line correlating with the Great␣
 ↪Recession/Financial Crisis
# - 2020: Deviation from the smooth line correlating with the COVID-19 pandemic
```



Split Data into Training and Test Sets

[37]:
```
# Split the data into training and test sets
# Test Set: Last year of data (July 2020 - June 2021)
test_sales = sales_long[(sales_long['DATE'] >= pd.to_datetime("2020-07-01")) &
                        (sales_long['DATE'] <= pd.to_datetime("2021-06-30"))]

# Train Set: Rest of the data
train_sales = sales_long[sales_long['DATE'] < pd.to_datetime("2020-07-01")]
```

Build Predictive Model

[38]:
```
# Prepare the data for modeling
X_train = np.array(train_sales['DATE'].map(pd.Timestamp.toordinal)).reshape(-1,␣
 ↪1)
y_train = train_sales['SALES']

# Build a predictive model using the training set
model = LinearRegression()
model.fit(X_train, y_train)

# Print the model summary (coefficients)
print("Intercept: ", model.intercept_)
print("Coefficient: ", model.coef_[0])
```

```
Intercept:  -20708473.56864229
Coefficient:  28.68349853289882
```

Predict Monthly Retail Sales for Test Set

[39]:
```
# Prepare the test data for predictions
X_test = np.array(test_sales['DATE'].map(pd.Timestamp.toordinal)).reshape(-1, 1)

# Make a copy of the test set to avoid the SettingWithCopyWarning
test_sales_copy = test_sales.copy()

# Use the model to predict the monthly retail sales on the last year of data
test_sales_copy['PREDICTED'] = model.predict(X_test)
test_sales_copy.head()
```

[39]:

	YEAR	MONTH	SALES	DATE	PREDICTED
208	2020	JUL	481627.0	2020-07-28	449450.188174
238	2020	AUG	483716.0	2020-08-28	450339.376629
268	2020	SEP	493327.0	2020-09-28	451228.565083
298	2020	OCT	493991.0	2020-10-28	452089.070039

```
328  2020   NOV  488652.0 2020-11-28  452978.258494
```

Report RMSE

[40]:
```
# Report the RMSE of the model predictions on the test set
rmse = np.sqrt(mean_squared_error(test_sales_copy['SALES'],␣
 ↪test_sales_copy['PREDICTED']))
print("RMSE: ", rmse)

# Report the standard deviation of the actual values in the test set
std_dev = np.std(test_sales_copy['SALES'])
print("Standard Deviation: ", std_dev)
```

```
RMSE:  66429.10224838086
Standard Deviation:  30859.042900294593
```