

The screenshot shows a Google Colab notebook titled "Untitled12.ipynb". The code cell [1] contains imports for pandas, numpy, matplotlib, scikit-learn, and NLTK, along with a command to download stopwords. The output shows the download progress: "[nltk_data] Downloading package stopwords to /root/nltk_data... [nltk_data] Unzipping corpora/stopwords.zip". The code cell [2] reads a CSV file "content/retail_ai_dataset.csv" and displays its first five rows using df.head(). The resulting DataFrame is as follows:

	date	sales	temperature	holiday	promotion	customer_feedback
0	2023-01-01	210	25	1	1	Loved the discount, great value!
1	2023-01-02	180	26	0	0	Delivery was slow
2	2023-01-03	220	27	0	1	Amazing quality and fast shipping
3	2023-01-04	160	28	0	0	Product was damaged
4	2023-01-05	300	24	1	1	Festival offer was excellent

The code cell [3] converts the 'date' column to datetime and sets it as the index. The bottom status bar indicates the environment is "Python 3", the time is "6:58 PM", and the date is "11-02-2026".

```

[3] df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)

[4] exog = df[['temperature', 'holiday', 'promotion']]
model = SARIMAX(df['sales'], exog=exog, order=(1,1,1), seasonal_order=(1,1,1,1))
results = model.fit()

[5] forecast = results.predict(start=len(df), end=len(df)+6,
                               exog=exog[-6:])

print("\nSALES FORECAST:")
print(forecast)

[6] SALES FORECAST:
15 238.959764
16 178.889911
17 210.171173
18 269.879642
19 287.862732
20 143.554741
21 318.525082
Name: predicted_mean, dtype: float64
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:966: UserWarning: Non-stationary starting autoregressive parameters
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA parameters found.
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:986: UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parameters.
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:986: UserWarning: Too few observations to estimate starting parameters for seasonal ARMA. All parameters except for variances will be set to zeros
warn("Too few observations to estimate starting parameters$".format(len(df) - len(exog)))

[7] plt.figure(figsize=(10,5))
plt.plot(df['sales'], label='Actual')
plt.plot(forecast, label='Forecast', color='red')
plt.legend()
plt.title("Sales Forecast")
plt.show()

[8] Sales Forecast

```

The chart displays a historical sales dataset from 1970 to approximately 2010, with a six-year-ahead forecast extending to about 2026. The actual sales data shows a general upward trend with some fluctuations. The forecast, represented by a red line, starts at a value of approximately 239 in 2020 and rises to about 318 by 2026, remaining consistently below the actual sales line throughout the forecast period.

Untitled12.ipynb

```
def get_sentiment(text):
    return Textblob(text).sentiment.polarity

df['sentiment_score'] = df['customer_feedback'].apply(get_sentiment)
df['sentiment'] = df['sentiment_score'].apply(lambda x: "Positive" if x>0 else "Negative")

print("SENTIMENT RESULTS:")
print(df[['customer_feedback','sentiment']].head())
```

SENTIMENT RESULTS:

customer_feedback	sentiment
2020-01-01 Loved the discount, great value	Positive
2020-01-02 Delivery was slow	Negative
2020-01-03 Amazing quality and fast shipping	Positive
2020-01-04 Product was damaged	Negative
2020-01-05 Festival offer was excellent	Positive

```
df['label'] = df['sentiment'].map({"Positive":1, "Negative":0})

X_train, X_test, y_train, y_test = train_test_split(
    df[['customer_feedback']], df[['label']], test_size=0.2)

text_model = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression())
])
```

text_model.fit(X_train, y_train)
accuracy = text_model.score(X_test, y_test)

```
Feedback Classifier Accuracy: (accuracy*100;.2f)%"
```

Variables Terminal

29°C Partly cloudy

6:58 PM Python 3

ENG IN 18:59 11-02-2026

Untitled12.ipynb

```
df[['customer_feedback']], df[['label']], test_size=0.2)

text_model = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression())
])
```

text_model.fit(X_train, y_train)
accuracy = text_model.score(X_test, y_test)

```
Feedback Classifier Accuracy: (accuracy*100;.2f)%"
```

Feedback Classifier Accuracy: 66.6%

```
avg_sentiment = df['sentiment_score'].mean()
avg_sales = df['sales'].mean()

def generate_insight():
    if avg_sentiment > 0:
        mood = "positive"
    else:
        mood = "negative"

    return f"""
        All INSIGHT REPORT
        -----
        Average Sales: {avg_sales:.2f}
        Overall Customer Mood: {mood}
        Forecast Trend: Increasing sales expected due to promotions & holidays.
        Key Driver: Customer sentiment strongly influences sales performance.
        Recommendation: Increase marketing during high-sentiment periods.
        """
print(generate_insight())
```

Variables Terminal

29°C Partly cloudy

6:58 PM Python 3

ENG IN 18:59 11-02-2026

The screenshot shows a Google Colab notebook titled "Untitled12.ipynb". The code cell contains Python code for generating AI insights based on average sentiment. The output cell displays the generated report, which includes average sales, overall customer mood, forecast trend, key driver, and a recommendation. The report concludes with a success message. Below the notebook, the system tray shows the date (11-02-2026), time (6:58 PM), and Python 3 environment. A weather widget indicates it's 29°C and partly cloudy.

```
avg_sales = url['Sales'].mean()

def generate_insight():
    if avg_sentiment > 0:
        mood = "positive"
    else:
        mood = "negative"

    return f"""
    AI INSIGHT REPORT
    -----
    Average Sales: {avg_sales:2f}
    Overall Customer Mood: {mood}
    Forecast Trend: Increasing sales expected due to promotions & holidays.
    Key Driver: Customer sentiment strongly influences sales performance.
    Recommendation: Increase marketing during high-sentiment periods.
    """

print(generate_insight())

AI INSIGHT REPORT
-----
Average Sales: 221.33
Overall Customer Mood: positive
Forecast Trend: Increasing sales expected due to promotions & holidays.
Key Driver: Customer sentiment strongly influences sales performance.
Recommendation: Increase marketing during high-sentiment periods.

print("\nEND-TO-END AI SYSTEM EXECUTED SUCCESSFULLY")

...
END-TO-END AI SYSTEM EXECUTED SUCCESSFULLY
```

Variables Terminal 29°C Partly cloudy 6:58 PM Python 3 ENG IN 18:59 11-02-2026