Question 1: What is Anomaly Detection? Explain its types (point, contextual, and collective anomalies) with examples.

Answer:

Anomaly Detection

Anomaly Detection is the process of identifying data points, events, or observations that significantly differ from the majority of the data. These unusual patterns are called **anomalies** or **outliers**.

It is widely used in fraud detection, network security, fault detection, healthcare monitoring, and time-series forecasting.

Types of Anomalies

1. Point Anomalies

A single data point that is significantly different from the rest.

Example:
In credit card transactions, a ₹5,00,000 transaction when the usual spending is ₹2,000–₹5,000.

In temperature data, if daily temperature is around 30°C and one day shows 60°C.

---

2. Contextual Anomalies

A data point that is anomalous in a specific context (time, location, etc.).

Example:
30°C is normal in summer but abnormal in winter.

High electricity usage at 3 PM is normal, but the same at 3 AM may be abnormal.

---

3. Collective Anomalies

A group of data points that together form an anomaly, even if individual points are normal.

Example:
Gradual unusual increase in server traffic over 10 minutes indicating a DDoS attack.

A sudden drop in energy usage for multiple consecutive intervals.

Question 2: Compare Isolation Forest, DBSCAN, and Local Outlier Factor in terms of approach and suitable use cases.

Answer:

Method	Approach	Key Idea	Suitable Use Cases
Isolation Forest	Tree-based	Randomly partitions data; anomalies need fewer splits	High-dimensional data, large datasets
DBSCAN	Density-based clustering	Points in low-density regions are anomalies	Spatial data, clustering-based anomaly detection
Local Outlier Factor (LOF)	Density-based local comparison	Compares local density of a point to its neighbors	When local density variation is important

---

Isolation Forest

- Works by randomly splitting data.
- Anomalies are easier to isolate.
- Efficient and scalable.

Best for: Fraud detection, high-dimensional datasets.

DBSCAN

- Groups points based on density.
- Points not belonging to any cluster are anomalies.

Best for: Spatial clustering problems.

Local Outlier Factor (LOF)

- Measures local density deviation.
- Detects anomalies relative to neighbors.

Best for: Datasets with varying density.

Question 3: What are the key components of a Time Series? Explain each with one example.

Answer:

Time series consists of:

1. Trend

Long-term upward or downward movement.

Example: Increase in airline passengers over years.

---

2. Seasonality

Regular repeating patterns over fixed time intervals.

Example: Ice cream sales increasing every summer.

---

3. Cyclical Component

Long-term fluctuations without fixed period.

Example: Economic boom and recession cycles.

---

4. Residual (Noise)

Random variations after removing trend and seasonality.

Example: Sudden unexpected stock market change.

Question 4: Define Stationarity in Time Series. How can you test and transform a non-stationary series into a stationary one?

Answer:

Stationary Time Series

A time series is stationary if:

- Mean is constant
- Variance is constant
- Autocorrelation is constant over time

Testing Stationarity

1. **Augmented Dickey-Fuller (ADF) Test**
    - Null hypothesis: Non-stationary
    - p-value < 0.05 → stationary
2. Rolling mean & variance plots

Transformations to Make Stationary

1. Differencing
   yt−yt−1y_t - y_{t-1}yt−yt−1
2. Log Transformation
3. Seasonal Differencing

4. Detrending

Question 5: Differentiate between AR, MA, ARIMA, SARIMA, and SARIMAX

Answer:

Model	Structure	Use Case
AR(p)	Depends on past values	When data depends on previous values
MA(q)	Depends on past errors	When shocks influence series
ARIMA(p,d,q)	AR + Differencing + MA	Non-seasonal data
SARIMA(p,d,q)(P,D,Q,m)	ARIMA + Seasonality	Seasonal data
SARIMAX	SARIMA + External variables	When external predictors exist

---

Examples

- ARIMA → Stock prices
- SARIMA → Monthly airline passengers
- SARIMAX → Energy demand with weather factors

Question 6: AirPassengers Decomposition

Answer (Python Code):

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.datasets import airpassengers

# Load dataset
data = airpassengers.load_pandas().data
```

```
data['Month'] = pd.date_range(start='1949-01', periods=len(data), freq='M')
data.set_index('Month', inplace=True)

# Plot original series
plt.figure()
plt.plot(data['AirPassengers'])
plt.title("Original Time Series")
plt.show()

# Decomposition
decomposition = seasonal_decompose(data['AirPassengers'], model='multiplicative')
decomposition.plot()
plt.show()
```

Output:

- Original series plot
- Trend component
- Seasonal component
- Residual component

Question 7: Isolation Forest on NYC Taxi Fare

```
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import pandas as pd

# Sample numeric dataset
df = pd.read_csv("nyc_taxi.csv")

model = IsolationForest(contamination=0.05)
df['anomaly'] = model.fit_predict(df[['fare_amount', 'trip_distance']])

plt.scatter(df['fare_amount'], df['trip_distance'], c=df['anomaly'])
plt.title("Isolation Forest Anomaly Detection")
plt.show()
```

Question 8: SARIMA Forecast

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
model = SARIMAX(data['AirPassengers'], order=(1,1,1),
             seasonal_order=(1,1,1,12))
result = model.fit()

forecast = result.forecast(12)

plt.plot(data['AirPassengers'])
plt.plot(forecast)
plt.title("SARIMA Forecast")
plt.show()
```

Question 9: Local Outlier Factor

```
from sklearn.neighbors import LocalOutlierFactor
import numpy as np

X = df[['fare_amount', 'trip_distance']].values

lof = LocalOutlierFactor(n_neighbors=20)
y_pred = lof.fit_predict(X)

plt.scatter(X[:,0], X[:,1], c=y_pred)
plt.title("LOF Anomaly Detection")
plt.show()
```

Question 10: Real-Time Power Grid Monitoring Workflow

Answer:

1. Anomaly Detection (Streaming Data)

I would use:

- **Isolation Forest** for high-dimensional streaming data.
- Sliding window approach.
- Retrain periodically.

---

2. Short-Term Forecasting

Use **SARIMAX** because:

- Energy demand is seasonal (daily/weekly).
- Weather affects usage.
- Region-specific external factors.

---

3. Validation & Monitoring

- Use RMSE, MAE.
- Rolling forecast validation.
- Monitor drift.
- Retrain model weekly/monthly.

---

4. Business Impact

- Prevent power outages.
- Detect equipment failure early.
- Optimize load balancing.
- Improve demand planning.
- Reduce operational cost.

Sample Python Workflow

```
# SARIMAX with weather
model = SARIMAX(data['energy_usage'],
            exog=data[['temperature']],
            order=(1,1,1),
            seasonal_order=(1,1,1,96))

result = model.fit()
forecast = result.forecast(steps=96)
```