IIoT Network Analysis: Age of Information and Reliability Trade-offs

```python
# Import necessary libraries
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import tensorflow as tf

# Load the dataset
df = pd.read_csv('iiot_network_data.csv')
```

1. Conceptual Understanding (20 points)

Instructions:

a) Explain the concept of Age of Information (AoI) in your own words and why it's important for IIoT applications.

The Age of Information (AoI) measures how "fresh" the information is that a system receives. At any moment (let's call it t), AoI is the time passed since the last update was created. Mathematically, if the most recent update at the receiver was generated at time u, then the AoI at time t is $\Delta(t)=t-u$. Whenever a new update arrives, the AoI resets to 1 (indicating it's one unit of time since the update was made) and grows steadily over time when no new updates come in.

AoI is different from usual network measurements like delay or throughput. Delay shows how long it takes for a packet to travel across the network, while throughput measures how many packets are delivered over time. On the other hand, AoI focuses on how current or "fresh" the information at the receiver is, which is especially important for systems that rely on real-time updates for monitoring or making decisions.

In Industrial Internet of Things (IIoT) systems, having up-to-date data is essential for managing processes, ensuring safety, and improving efficiency. There are two main types of data traffic:

- AoI-Oriented Traffic (Monitoring Data): This includes regular updates from sensors (like temperature or pressure readings) that a central system uses to make decisions in real time. It's important for this data to be fresh, as using outdated information can lead to incorrect actions. AoI helps measure how current this data is at the controller.

- Deadline-Oriented Traffic (Safety-Critical Data): This type of traffic consists of urgent messages, like alarms or emergency alerts, which must reach their destination within strict time limits. While AoI is less important for these, they share the same network with monitoring data, leading to competition for resources and creating a balance to manage.

b) Describe the difference between AoI-oriented traffic and deadline-oriented traffic in IIoT networks. Provide real-world examples for each.

AoI-oreiented traffic is different from deadline-oriented traffic by following factors:

1. Purpose:
   - AoI-oreiented traffic: Regular monitoring of industrial processes through periodic sensor updates (e.g., temperature, vibration, humidity).
   - Deadline-oriented traffic: Transmitting critical or emergency information (e.g., fault alarms, system failure alerts) that impacts system safety or stability.
2. Trigger Type:
   - AoI-oreiented traffic: Time-triggered — generated at regular intervals regardless of system events.
   - Deadline-oriented traffic: Event-triggered — generated sporadically when specific events or emergencies occur.
3. Performance Goal:
   - AoI-oreiented traffic: Freshness of information. The central controller needs to have the most up-to-date data to make timely and effective decisions.
   - Deadline-oriented traffic: Reliability within a strict time constraint. The information must be delivered before a fixed deadline to be useful.
4. Key Metric:
   - AoI-oreiented traffic: Age of Information (AoI) — measures how old the most recently received data is.
   - Deadline-oriented traffic: Packet Loss Probability (PLP) — the probability that a packet fails to be delivered before its deadline.
5. Delivery Behavior:
   - AoI-oreiented traffic: A new update is generated and transmitted when needed (generate-at-will model). There is no retransmission if a packet is lost.

- Deadline-oriented traffic: Packets are retransmitted until successfully delivered or the deadline expires. If not delivered on time, they are considered lost.

6. Sensitivity:

   - AoI-oreiented traffic: Sensitive to how recent the data is, not necessarily if every single packet is delivered.
   - Deadline-oriented traffic: Sensitive to timely and reliable delivery. A single missed alarm can lead to dangerous consequences.

Real life example:

- AoI-oriented traffic: Temperature Monitoring in a Chemical Plant. Sensors installed in different parts of a chemical processing unit measure the temperature every 5 seconds and send updates to a central control system. The controller must always know the current temperature to regulate heating or cooling processes. If the temperature rises unexpectedly and the data is outdated, the system may fail to react in time. Age of Information (AoI) — low AoI means the temperature reading is recent and trustworthy.
- Deadline-oriented traffic: Emergency Gas Leak Alert in an Oil Refinery. A gas sensor detects a dangerous leak and immediately sends an emergency alert to the control system. The alert must reach the controller within, say, 1 second. If the packet arrives late or not at all, safety systems (like alarms or shut-off valves) may not activate in time, leading to hazardous situations. Packet Loss Probability (PLP) — the probability that the alert fails to arrive within the strict deadline must be very low.

2. Data Exploration and Visualizaton

Instructions:

a) Explore the dataset using pandas. Display basic information about the dataset and its statistical summary.

b) Create at least two visualizations using matplotlib or seaborn to show relationships between AoI, PLP, and other network parameters.

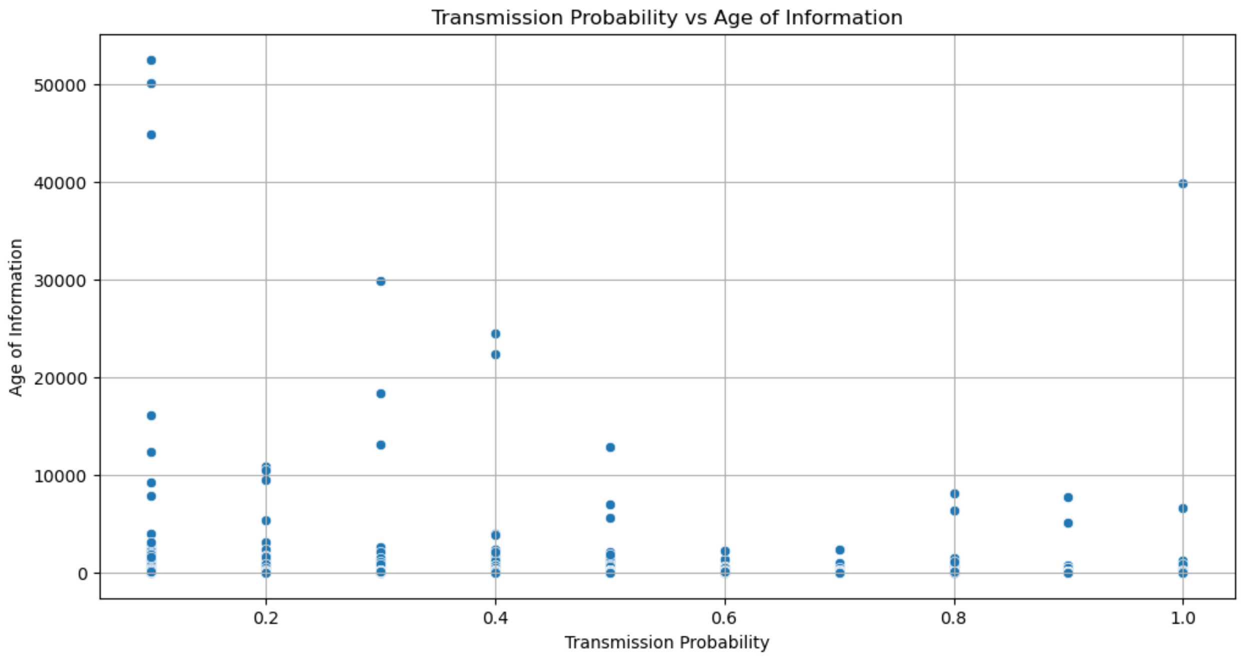
c) Identify and discuss any patterns or trends you observe in the data.

Complete the code below and add your observations.

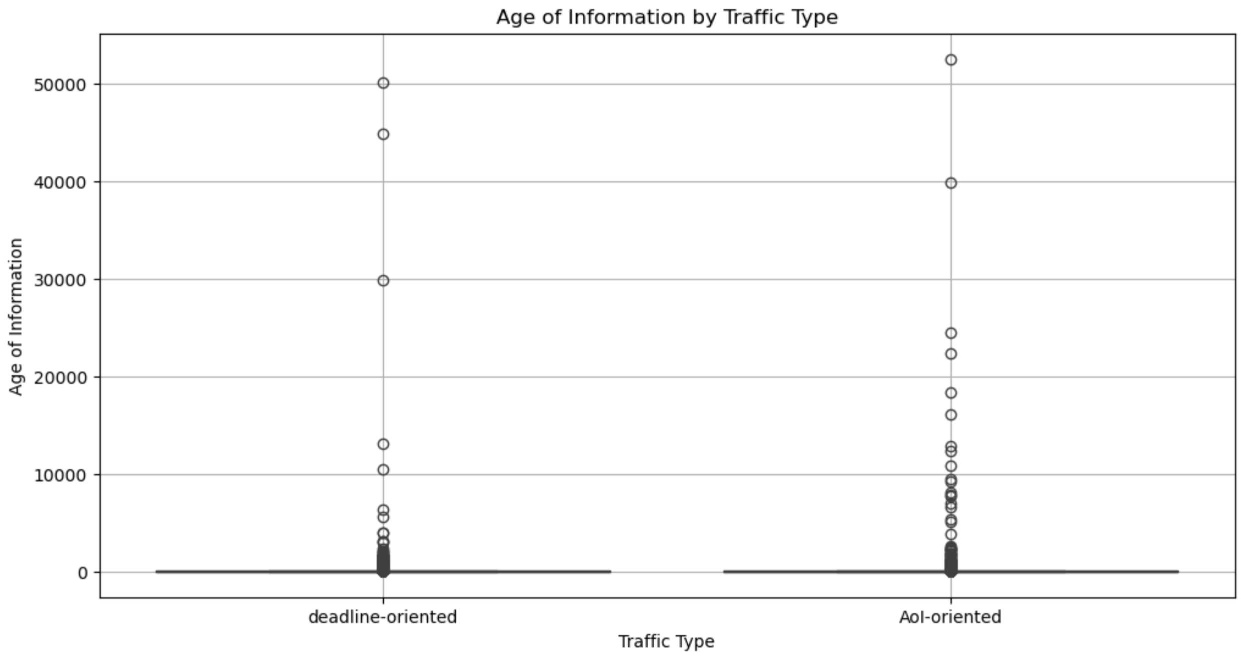
```
# Display the first few rows of the dataset
df.head()
```

```
                    timestamp  node_id       traffic_type  \
0  2024-06-30 17:10:10.430548       61  deadline-oriented
1  2024-07-01 03:12:10.430548       55       AoI-oriented
2  2024-06-30 17:44:10.430548       63  deadline-oriented
3  2024-07-01 08:23:10.430548       77  deadline-oriented
4  2024-06-30 17:05:10.430548       44  deadline-oriented

   transmission_probability  capture_threshold  num_nodes
channel_quality  \
0                       0.9               -0.5          3
0.6
1                       0.4               -2.0          2
0.7
2                       0.3                0.0          4
0.6
3                       0.4                0.0          1
0.3
4                       0.7                0.5          2
0.4

   age_of_information  packet_loss_probability
0            4.760106                 0.724432
1            4.068644                 0.480900
2           19.007878                 0.835932
3           10.467934                 0.730784
4           14.010374                 0.906584

# Show basic information (data types, non-null counts)
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   timestamp                 10000 non-null  object
 1   node_id                   10000 non-null  int64
 2   traffic_type              10000 non-null  object
 3   transmission_probability  10000 non-null  float64
 4   capture_threshold         10000 non-null  float64
 5   num_nodes                 10000 non-null  int64
 6   channel_quality           10000 non-null  float64
 7   age_of_information        10000 non-null  float64
 8   packet_loss_probability   10000 non-null  float64
dtypes: float64(5), int64(2), object(2)
memory usage: 703.3+ KB

# Display summary statistics for numerical columns
df. describe()
```

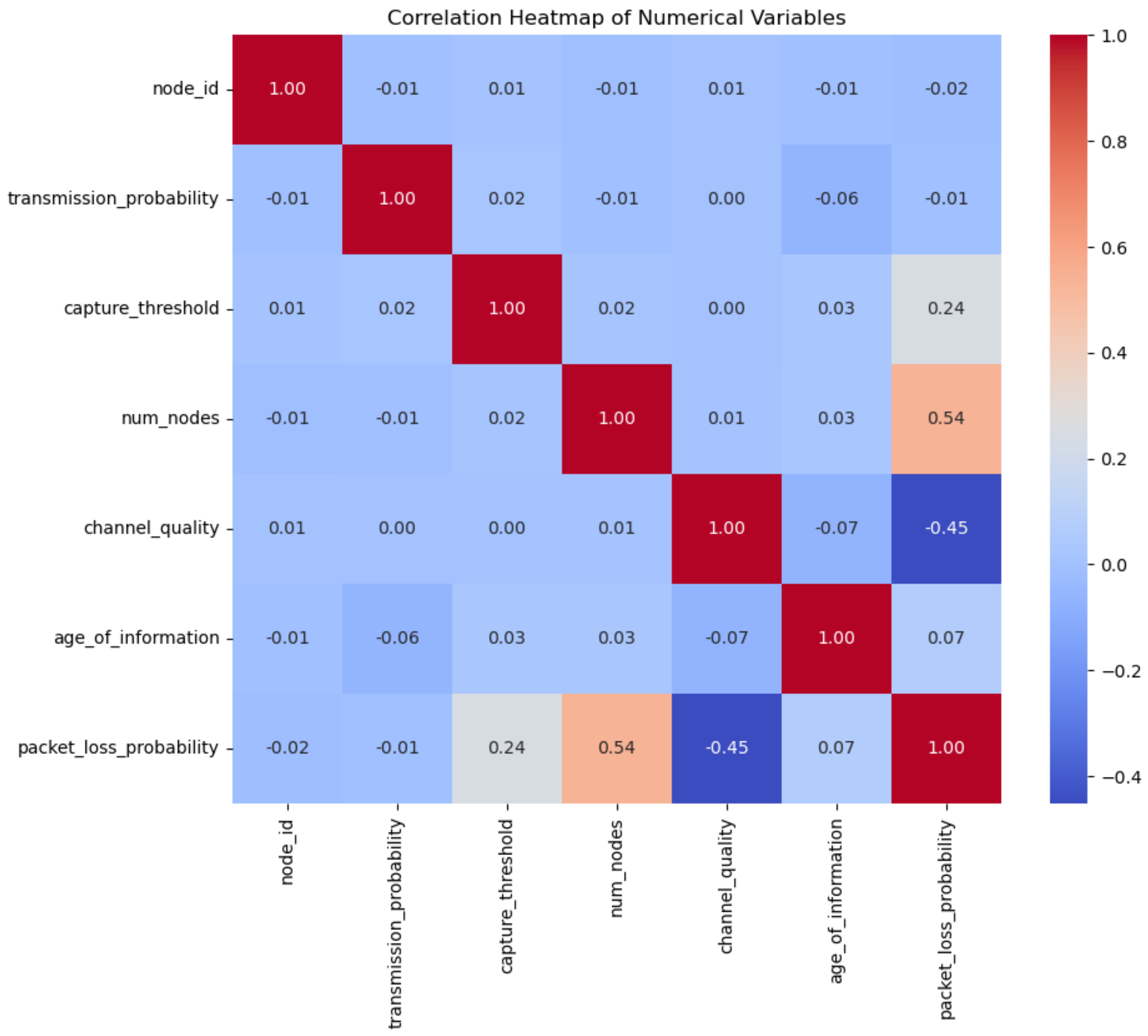
```
           node_id  transmission_probability  capture_threshold  \
count  10000.000000              10000.000000       10000.000000   
mean      50.638400                  0.548460          -0.001800   
std       29.020101                  0.288548           1.284664   
min        1.000000                  0.100000          -2.000000   
25%       26.000000                  0.300000          -1.000000   
50%       51.000000                  0.500000           0.000000   
75%       76.000000                  0.800000           1.000000   
max      100.000000                  1.000000           2.000000   

          num_nodes  channel_quality  age_of_information  \
count  10000.000000     10000.000000        1.000000e+04   
mean       5.553100         0.499100                 inf   
std        2.850122         0.317656                 NaN   
min        1.000000         0.000000        1.000000e+00   
25%        3.000000         0.200000        1.032026e+01   
50%        6.000000         0.500000        2.468121e+01   
75%        8.000000         0.800000        9.462189e+01   
max       10.000000         1.000000                 inf   

       packet_loss_probability  
count             10000.000000  
mean                  0.853774  
std                   0.184140  
min                   0.000000  
25%                   0.819893  
50%                   0.908372  
75%                   0.968325  
max                   1.000000  

# Create visualizations
plt.figure(figsize=(12, 6))
# Add your code here to create a scatter plot of
transmission_probability vs age_of_information
sns.scatterplot(data=df, x='transmission_probability',
y='age_of_information')
plt.title('Transmission Probability vs Age of Information')
plt.xlabel('Transmission Probability')
plt.ylabel('Age of Information')
plt.grid(True)
plt.show()
```



```
plt.figure(figsize=(12, 6))
# Add your code here to create another relevant visualization
sns.boxplot(data=df, x='traffic_type', y='age_of_information')
plt.title('Age of Information by Traffic Type')
plt.xlabel('Traffic Type')
plt.ylabel('Age of Information')
plt.grid(True)
plt.show()
```



```
# Heatmap of correlations between numerical variables
plt.figure(figsize=(12, 6))
# Compute correlation matrix
corr_matrix = df.corr(numeric_only=True) # calculates pairwise
correlations between all numerical columns

# Create the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True)
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()

<Figure size 1200x600 with 0 Axes>
```



	node_id	transmission_probability	capture_threshold	num_nodes	channel_quality	age_of_information	packet_loss_probability
node_id	1.00	-0.01	0.01	-0.01	0.01	-0.01	-0.02
transmission_probability	-0.01	1.00	0.02	-0.01	0.00	-0.06	-0.01
capture_threshold	0.01	0.02	1.00	0.02	0.00	0.03	0.24
num_nodes	-0.01	-0.01	0.02	1.00	0.01	0.03	0.54
channel_quality	0.01	0.00	0.00	0.01	1.00	-0.07	-0.45
age_of_information	-0.01	-0.06	0.03	0.03	-0.07	1.00	0.07
packet_loss_probability	-0.02	-0.01	0.24	0.54	-0.45	0.07	1.00

Write your observations about the data and visualizations here:

1. The scatter plot shows a small decrease overall, and the correlation coefficient is -0.062, indicating a very weak negative link between the variables. This can mean that raising the transmission probability might slightly lower AoI (making the data fresher), but it’s not a major influence—likely because of issues like competition for the network or packet collisions.
2. The box plot shows that deadline-oriented traffic usually has a lower and more stable AoI, while AoI-oriented traffic is less predictable and often has higher AoI values. This fits the system's design. Deadline-oriented packets are repeatedly sent until they either arrive or are dropped, ensuring timely delivery. On the other hand, AoI-oriented packets are sent only once, focusing on keeping the data fresh rather than guaranteeing it gets delivered.
3. The heatmap shows that all other factors have a very weak connection with age_of_information, with the strongest being packet_loss_probability at just 0.0685. AoI seems to be shaped by several factors working together, rather than one clear, strong cause. This kind of complexity is common in diverse IIoT networks, where different traffic types and access probabilities interact.

Overall, I believe AoI and PLP need to be managed together through efficient network design, using tools like flexible access control, scheduling methods, and policies that consider traffic patterns. Network settings work together, not separately, and finding the right balance between freshness (AoI) and reliability (PLP) is crucial for IIoT systems.

3. Machine Learning Model Development (35 points)

Instructions:

a) Prepare the data for machine learning (feature selection, scaling).

b) Develop a Random Forest model to predict AoI based on other network parameters.

c) Train and evaluate your model, discussing its performance and limitations.

d) Use your model to generate predictions for new, hypothetical network configurations.

Complete the code below and add your analysis.

```
# Prepare the data
X = df[['transmission_probability', 'capture_threshold', 'num_nodes',
'channel_quality']]
y_aoi = df['age_of_information']

# Combine X and y for safe filtering
df_clean = df[['transmission_probability', 'capture_threshold',
```

```python
'num_nodes', 'channel_quality', 'age_of_information']].copy()

# Replace inf with NaN, then drop rows with NaN
df_clean.replace([np.inf, -np.inf], np.nan, inplace=True)
df_clean.dropna(inplace=True)

# Redefine features and target
X = df_clean[['transmission_probability', 'capture_threshold',
'num_nodes', 'channel_quality']]
y_aoi = df_clean['age_of_information']


# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y_aoi,
test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
# Add your code here to fit the model
rf_model.fit(X_train_scaled, y_train)
```

```
RandomForestRegressor(random_state=42)
```

```python
# Make predictions
# Add your code here to make predictions on the test set
y_pred = rf_model.predict(X_test_scaled)

# Evaluate the model
# Add your code here to calculate MSE and R2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R² Score: {r2:.2f}")

# Feature importance
# Add your code here to display feature importances
# Get feature importances from the trained model
importances = rf_model.feature_importances_

# Create a DataFrame for better readability
feature_names = X.columns
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)
```

```
print(importance_df)
```

```
Mean Squared Error (MSE): 3423000.63
R² Score: -0.55
                 Feature  Importance
1      capture_threshold    0.440095
2              num_nodes    0.315242
3        channel_quality    0.153468
0  transmission_probability    0.091195
```

Write your analysis of the model performance and feature importances here:

- MSE: 3423000.63

  This means on average, the squared difference between the predicted AoI and the actual AoI is over 3 million. That's a very high error, especially if the typical age_of_information values in your dataset aren't in the millions. It suggests that the model's predictions are far off from the actual values.

- R^2 Score: -0.55

  This means it is making worse predictions than if you just guessed the average Age of Information every time. This model is clearly doing very poorly at capturing patterns in the data.

- Some conclusions about this model:

  Random Forest may not be suitable in its current setup — it could be underfitting, overfitting, or just not the right approach without more data tuning.

  Feature-target relationship may be weak — your selected features may not strongly predict age_of_information, which we already suspected from the correlation heatmap.

  Data issues may persist — outliers or high variance in age_of_information could be skewing the results.

- capture_threshold (44%): this is the most important feature. It shows that the model considers the threshold for successful packet decoding to have the biggest impact on Age of Information. A higher (stricter) capture threshold probably makes it more difficult to deliver packets successfully, which results in older and less fresh information.

- num_nodes (35%): this is the second most impactful feature. More nodes likely cause more contention on the network, leading to longer waiting times or failed transmissions. This means that a crowded network tends to hurt freshness of information.

```python
# Generate predictions for new, hypothetical network configurations:


# Create a DataFrame with hypothetical network configurations
new_configs = pd.DataFrame({
    'transmission_probability': [0.5, 0.7, 0.9],
    'capture_threshold': [0, 1, -1],
    'num_nodes': [3, 5, 7],
    'channel_quality': [0.6, 0.8, 0.4]
})

# Add your code here to make predictions for these new configurations
# Scale the hypothetical configurations
new_configs_scaled = scaler.transform(new_configs)

# Predict Age of Information using the trained model
predicted_aoi = rf_model.predict(new_configs_scaled)

# Display predictions
for i, aoi in enumerate(predicted_aoi, 1):
    print(f"Configuration {i}: Predicted Age of Information =
{aoi:.2f}")
```

```
Configuration 1: Predicted Age of Information = 10.61
Configuration 2: Predicted Age of Information = 16.05
Configuration 3: Predicted Age of Information = 11.21
```

4. Analysis and Insights (20 points)

Instructions:

Based on your data exploration and machine learning results:

a) Discuss the key factors that appear to influence the AoI-PLP trade-off in IIoT networks.

The results show that Age of Information (AoI) in IIoT networks is mainly affected by the capture threshold, number of nodes, and to a lesser extent, transmission probability and channel quality. A higher capture threshold makes it harder to decode packets, leading to older information. More nodes create more competition for the network, causing delays. While increasing transmission probability might seem helpful, it can actually result in more collisions in busy networks, raising AoI. Channel quality matters too, but its effect often depends on how strict the capture threshold is. Overall, balancing AoI (freshness) and PLP (reliability) is key, as improving one often reduces the other, requiring careful network design for better performance.

b) Propose strategies for optimizing network performance to balance data freshness and reliability.

- Adjust the transmission probability based on the current state of the network. For instance, lower the access rate during congestion to reduce collisions, or raise it when the network isn't busy to keep the data fresher.
- Adjust the physical-layer settings based on the link quality. When the channel is clear, lowering the capture threshold can help reduce packet loss and improve AoI. In noisy conditions, raising the threshold can avoid incorrect decoding.
- Use priority-based scheduling methods, like assigning specific time slots or using round-robin with weighted priorities, to make sure deadline-sensitive traffic is delivered reliably. At the same time, ensure AoI-sensitive traffic gets frequent updates without being blocked or delayed.
- Create efficient queueing strategies that restrict how many times deadline-oriented packets are resent to prevent network congestion. At the same time, allow outdated AoI-oriented packets to be dropped, freeing up bandwidth for newer, fresher data.

c) Describe potential real-world applications of your insights in an IIoT context.

- In industries working in remote locations (like pipelines or power grids), network bandwidth is often limited. By optimizing AoI and PLP, it's possible to monitor infrastructure status on time and make sure critical alerts (such as voltage spikes or pressure drops) are delivered reliably, even over poor or unstable connections.
- Automated forklifts and delivery robots in warehouses rely on up-to-date data, like location tracking and obstacle detection, to move safely. At the same time, critical control signals need to be sent reliably to prevent collisions. Using flexible access and scheduling methods helps keep operations smooth and safe.
- In risky places like chemical plants or oil refineries, sensors need to send alerts about gas leaks or pressure spikes with high dependability. At the same time, data like humidity or temperature must be kept up-to-date to prevent dangerous changes. Managing AoI and PLP together ensures both kinds of data work properly.

5. Bonus Challenge (10 points)

""" Instructions: Implement a simple deep learning model (e.g., a basic neural network) to predict both AoI and PLP simultaneously. Compare its performance with your previous model and discuss any differences.

Complete the code below and add your analysis.

```python
# Prepare data for deep learning model
y_plp = df['packet_loss_probability']
X_train, X_test, y_aoi_train, y_aoi_test, y_plp_train, y_plp_test =
train_test_split(
    X, y_aoi, y_plp, test_size=0.2, random_state=42)
```

```
# Create a simple neural network
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(2)  # Output layer for AoI and PLP
])

model.compile(optimizer='adam', loss='mse')

# Train the model
# Add your code here to fit the model

# Evaluate the model
# Add your code here to make predictions and calculate MSE for both
AoI and PLP
```

Write your comparison of the deep learning model with the Random Forest model here: