Wifi-POV

July 27, 2024

[79]:
```
import os
import time
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import STL
```

1 Data Reading

[63]:
```
file_path =  os.path.join('Datasets','Wifi.csv')
df = pd.read_csv(file_path)
df.head(10)
```

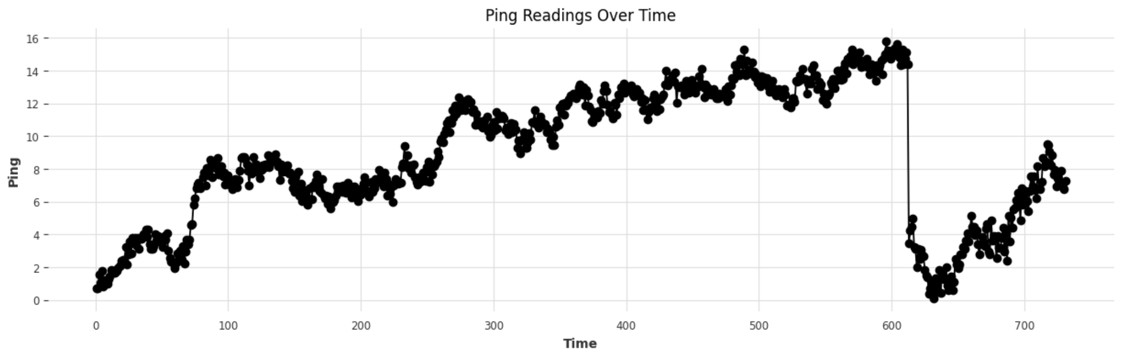
[63]:
```
    t      ping
0   1  0.702059
1   2  0.702852
2   3  1.527601
3   4  1.022392
4   5  1.784613
5   6  0.809713
6   7  1.195961
7   8  1.078758
8   9  1.006134
9  10  1.293807
```

2 EDA

2.1 Line Plots

The line plot shows the variation of ping readings over time. There seems to be some fluctuation, with no clear trend or periodicity visible at first glance

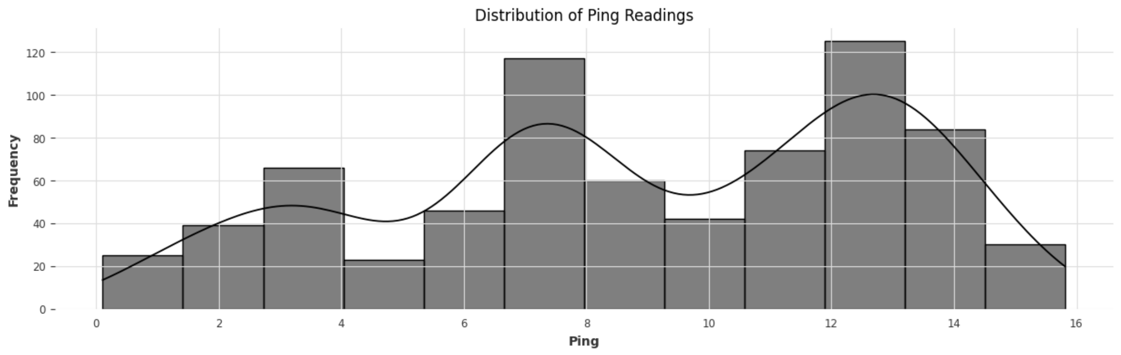
[72]:
```
plt.figure(figsize=(15, 4))
plt.plot(df['t'], df['ping'], marker='o')
plt.title('Ping Readings Over Time')
plt.xlabel('Time')
plt.ylabel('Ping')
plt.grid(True)
plt.show()
```



2.2 Distribution Plot

- The histogram displays the frequency distribution of ping readings. The x-axis represents the range of ping values, while the y-axis shows the frequency (or count) of ping readings within each bin.The height of each bar indicates how many ping values fall within the corresponding range.
- The KDE curve provides a smoothed estimate of the distribution of ping values.It helps visualize the probability density function of the ping data, showing where the data is concentrated and how it spreads out.Peaks in the KDE curve indicate values where the ping readings are most densely concentrated.
- The histogram combined with the KDE curve reveals the overall shape of the distribution. For example, if the KDE curve is unimodal (has a single peak), it suggests that the ping values are concentrated around a central value.

[73]:
```
plt.figure(figsize=(15, 4))
sns.histplot(df['ping'], kde=True)
plt.title('Distribution of Ping Readings')
plt.xlabel('Ping')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

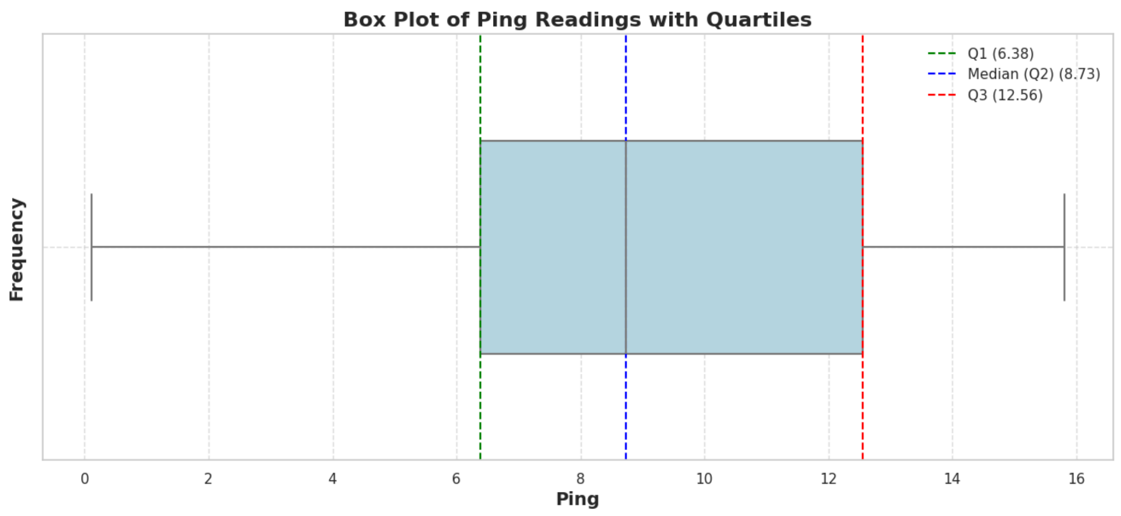


2.3 BoxPlots

Given Visualizes the distribution of ping readings, including the median, quartiles, and potential outliers where

- **Q1 Line (Green)**: Indicates the first quartile (25th percentile). 25% of the ping values are below this line.
- **Median Line (Blue)**: Shows the median (50th percentile), representing the central value of the ping readings.
- **Q3 Line (Red)**: Represents the third quartile (75th percentile). 75% of the ping values are below this line.
- **Outliers**: Points beyond the whiskers are considered outliers

[78]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(15, 6))

# Calculate quartiles
Q1 = df['ping'].quantile(0.25)
Q2 = df['ping'].median()  # Median (Q2)
Q3 = df['ping'].quantile(0.75)

sns.boxplot(x=df['ping'], width=0.5, fliersize=8, linewidth=1.5,␣
 ↪color='lightblue')
plt.axvline(Q1, color='green', linestyle='--', label=f'Q1 ({Q1:.2f})')
plt.axvline(Q2, color='blue', linestyle='--', label=f'Median (Q2) ({Q2:.2f})')
plt.axvline(Q3, color='red', linestyle='--', label=f'Q3 ({Q3:.2f})')
plt.title('Box Plot of Ping Readings with Quartiles', fontsize=16,␣
 ↪fontweight='bold')
plt.xlabel('Ping', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



3 Decomposing the Signal using STL

Seasonal-Trend decomposition using LOESS (STL) is a robust method of time series decomposition often used in economic and environmental analyses. The STL method uses locally fitted regression models to decompose a time series into trend, seasonal, and remainder components.we can apply STL to any dataset, but meaningful results are only returned if a recurring temporal pattern exists in the data.
Working
* The STL algorithm performs smoothing on the time series using LOESS in two loops; the inner loop iterates between seasonal and trend smoothing and the outer loop minimizes the effect of outliers. During the inner loop, the seasonal component is calculated first and removed to calculate the trend component. The remainder is calculated by subtracting the seasonal and trend components from the time series.

The three components of STL analysis relate to the raw time series as follows:
yi = si + ti + ri

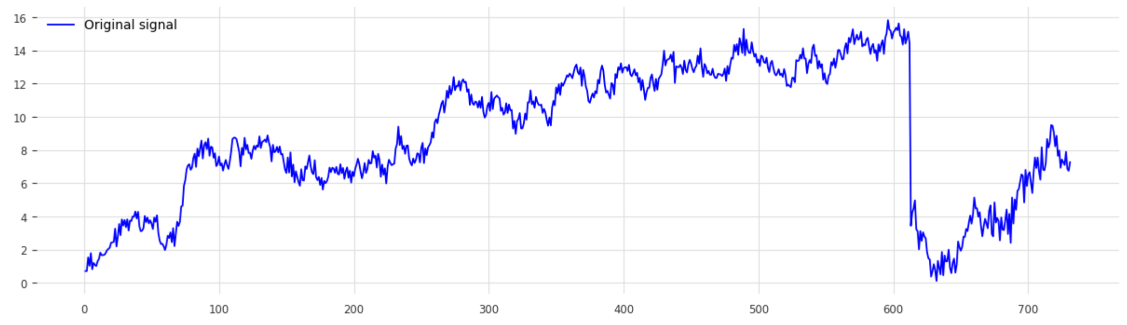
- yi = The value of the time series at point i.
- si = The value of the seasonal component at point i.
- ti = The value of the trend component at point i.
- ri = The value of the remainder component at point i.

[67]:
```
period = 60 # assume period is 60
stl = STL(df['ping'], period=period)
result = stl.fit()

trend = result.trend
seasonal = result.seasonal
residual = result.resid
```

```
residual_std = residual.std()
threshold = 3 * residual_std

anomalies = np.abs(residual) > threshold
```

3.1 Orignal Series

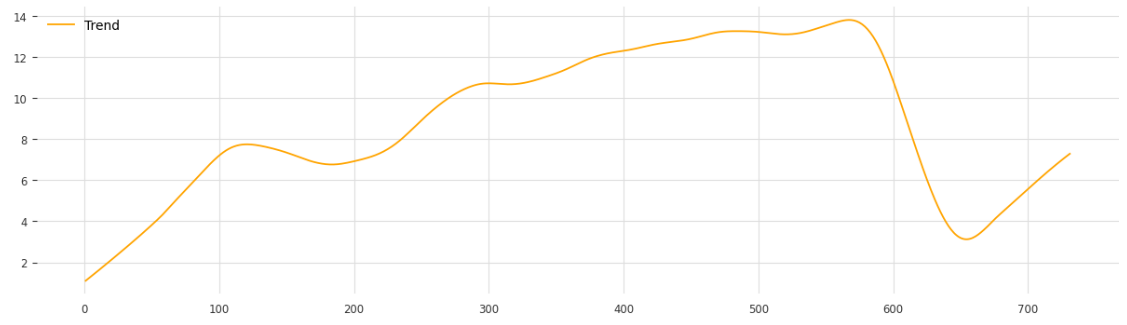
[68]:
```
plt.figure(figsize=(15, 4))
plt.plot(df['t'], df['ping'], label='Original signal', color='blue')
plt.legend(loc='upper left')
plt.show()
```



3.2 Trend

The orange line represents the trend component extracted from the time series data. This component shows the general direction or long-term movement.

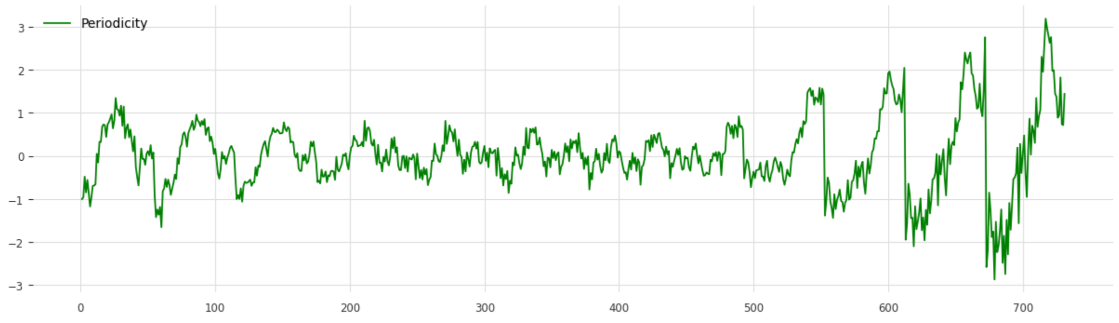
[69]:
```
plt.figure(figsize=(15, 4))
plt.plot(df['t'], trend, label='Trend', color='orange')
plt.legend(loc='upper left')
plt.show()
```



3.3 Periodicity

The green line represents the seasonal component extracted from the time series data. This component shows the periodic patterns or cycles that repeat over regular intervals.Given plot shows the recurring cycles or patterns within the data.from the observation data don't have strong consistent recursive pattern or the data contain have more then one periodicity.

```
[70]: plt.figure(figsize=(15, 4))
      plt.plot(df['t'], seasonal, label='Periodicity', color='green')
      plt.legend(loc='upper left')
      plt.show()
```

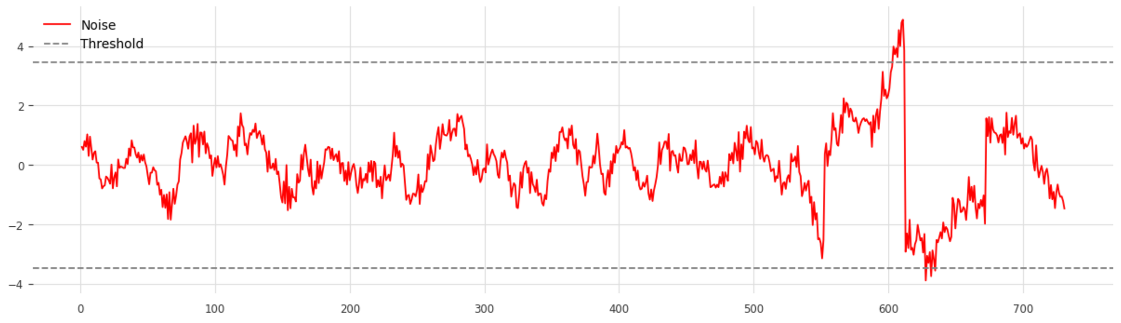


3.4 Noise with Threshold

- **Noise (Residual)**: The plot shows the residual component of the time series, which represents the part of the data that remains after removing the trend and seasonal components. It is plotted in red, indicating the deviations from the expected values.
- **Threshold Lines)**: The horizontal dashed lines represent the upper and lower thresholds. These thresholds are used to detect significant deviations or anomalies in the residuals.

The regions where the noise consistently crosses the threshold lines are of particular interest as they indicate points where the data deviates significantly from the norm. These are potential anomalies that may warrant further investigation

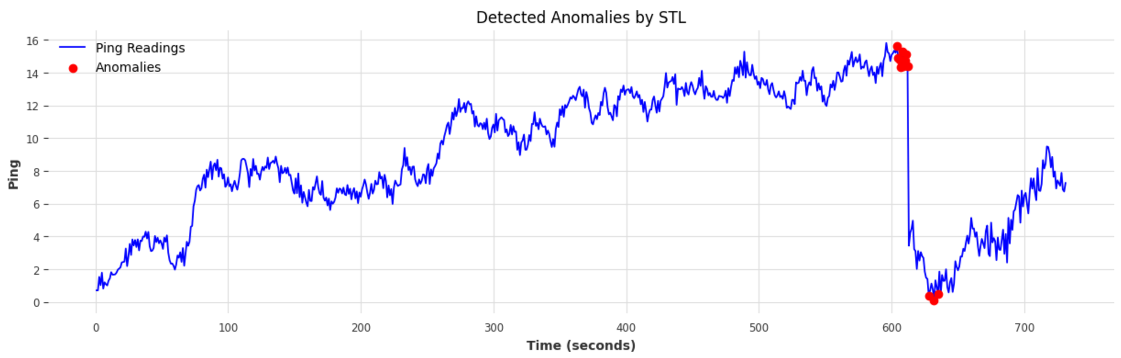
```
[71]: plt.figure(figsize=(15, 4))
      plt.plot(df['t'], residual, label='Noise', color='red')
      plt.axhline(y=threshold, color='gray', linestyle='--', label='Threshold')
      plt.axhline(y=-threshold, color='gray', linestyle='--')
      plt.legend(loc='upper left')
      plt.show()
```



3.5 Anomalies detected

- The blue line represents the original ping readings over time, showing the overall behavior of the data, including any trends and seasonal fluctuations.
- The red scatter points indicate the locations where anomalies have been detected. These points represent deviations from the expected pattern, as identified by the anomaly detection using STL.
- These deviations could be due to unusual spikes or drops in latency.

[37]:
```
plt.figure(figsize=(15, 4))
plt.plot(df['t'], df['ping'], label='Ping Readings ', color='blue')
plt.scatter(df['t'][anomalies], df['ping'][anomalies], color='red',␣
  ↪label='Anomalies', zorder=5)
plt.xlabel('Time (seconds)')
plt.ylabel('Ping')
plt.title('Detected Anomalies by STL')
plt.legend(loc='upper left')
plt.show()
```



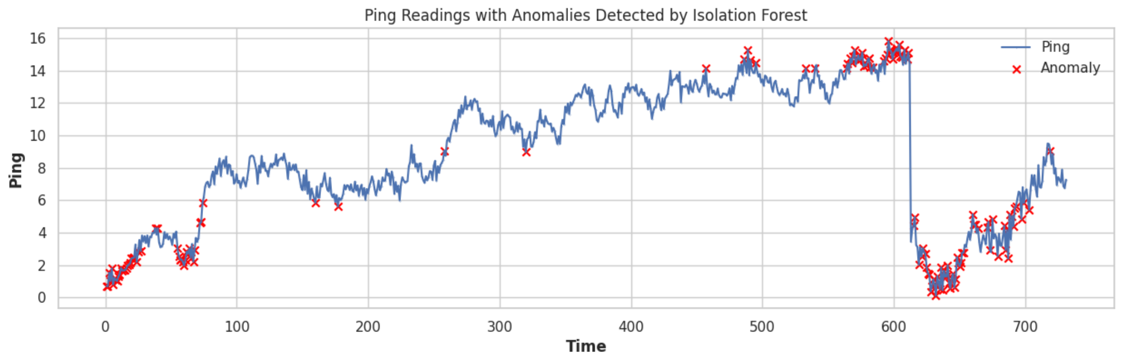
4 Anomaly Detection

5 Using ML Model

5.1 Isolation Forest

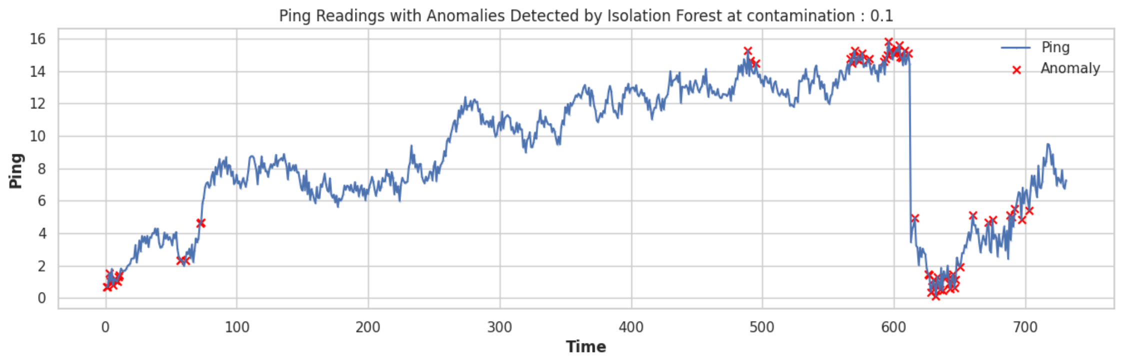
- **Working**
  - Isolation Forest is an anomaly detection algorithm based on the concept of isolating observations.
  - The algorithm isolates anomalies instead of profiling normal data points. It constructs a number of random trees (decision trees) where each tree partitions the data randomly.
  - Anomalies are more likely to be isolated quickly because they are different from the majority of the data.
  - Each observation's path length in the trees is measured. Anomalies are expected to have shorter path lengths because they are isolated earlier in the trees.
  - **Contamination Parameters**
    - This parameter is used to estimate the proportion of outliers in the data. It helps in adjusting the threshold for classifying an observation as an anomaly

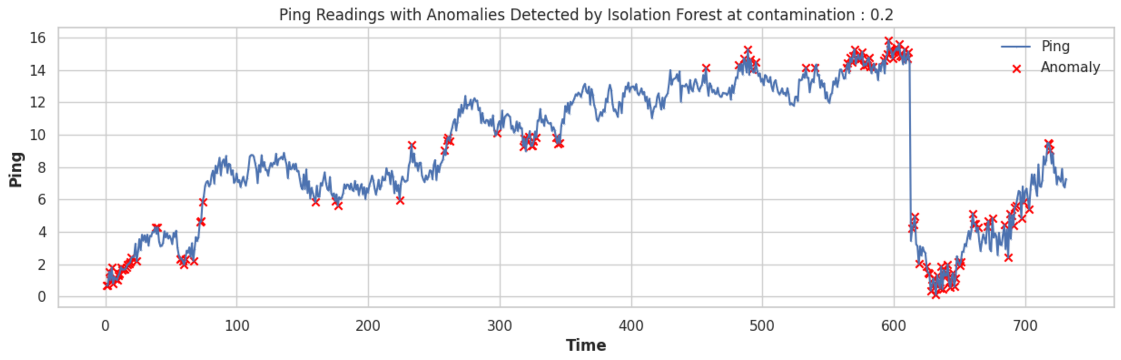
[94]:
```python
from sklearn.ensemble import IsolationForest

iso_forest = IsolationForest(contamination=0.2)
df['anomaly_if'] = iso_forest.fit_predict(df[['ping']])

plt.figure(figsize=(15, 4))
plt.plot(df['t'], df['ping'], marker='o', label='Ping',markersize=0.2)
plt.scatter(df['t'][df['anomaly_if'] == -1], df['ping'][df['anomaly_if'] ==␣
 ↪-1], color='red', label='Anomaly', marker='x')
plt.title('Ping Readings with Anomalies Detected by Isolation Forest')
plt.xlabel('Time')
plt.ylabel('Ping')
plt.legend()
plt.grid(True)
plt.show()
```

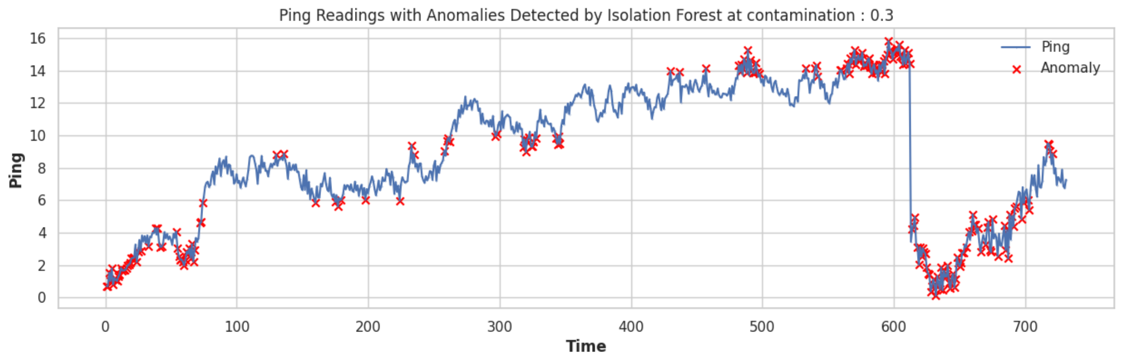


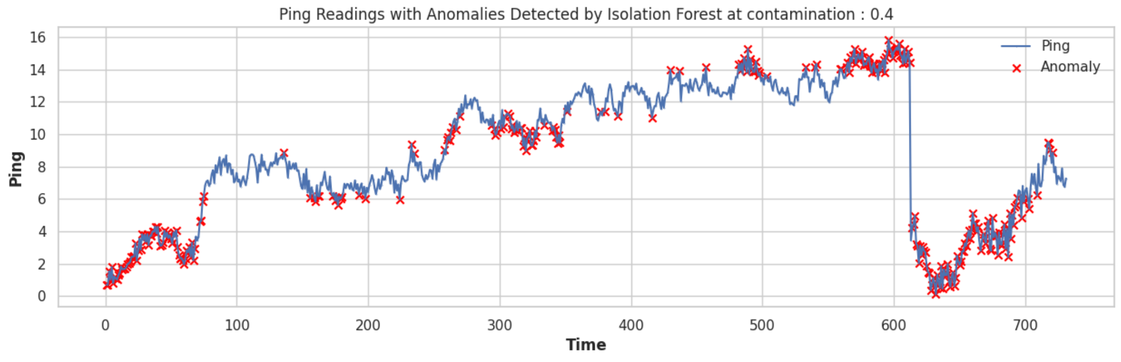
5.1.1 Anomolies base different contamination parameter of Isolation forecast Model

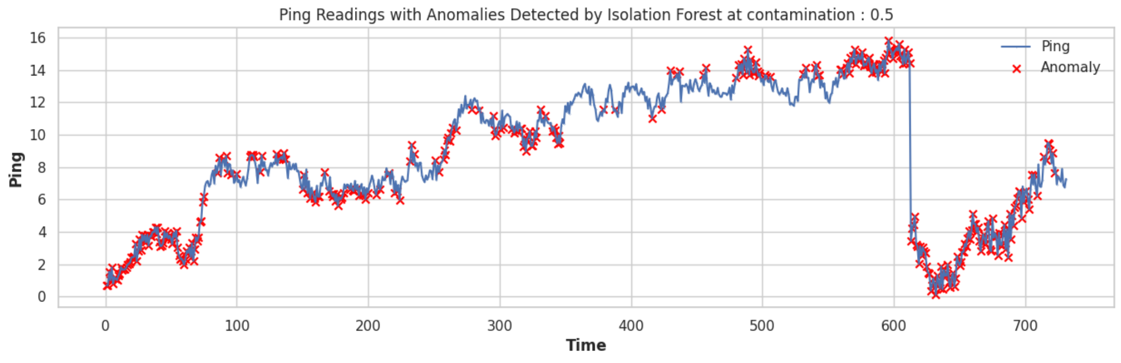
[95]:
```
from sklearn.ensemble import IsolationForest
for contamination in [0.1,0.2,0.3,0.4,0.5]:

    iso_forest = IsolationForest(contamination=contamination)
    df[f'anomaly_if_{contamination}'] = iso_forest.fit_predict(df[['ping']])

    plt.figure(figsize=(15, 4))
    plt.plot(df['t'], df['ping'], marker='o', label='Ping',markersize=0.2)
    plt.scatter(df['t'][df[f'anomaly_if_{contamination}'] == -1],␣
 ↪df['ping'][df[f'anomaly_if_{contamination}'] == -1], color='red',␣
 ↪label='Anomaly', marker='x')
    plt.title(f'Ping Readings with Anomalies Detected by Isolation Forest at␣
 ↪contamination : {contamination}')
    plt.xlabel('Time')
    plt.ylabel('Ping')
    plt.legend()
    plt.grid(True)
    plt.show()
    time.sleep(1)
    plt.close()
```











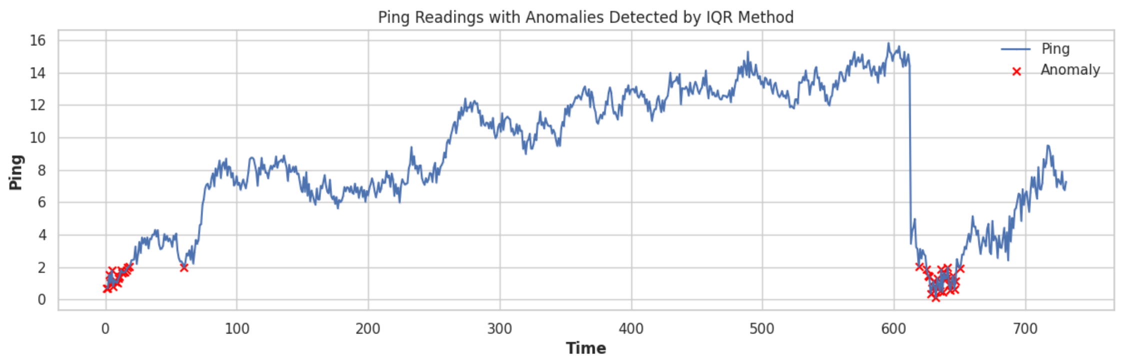
6 Statistical Approach

6.1 IQR Method

- Q1 and Q3 divide the data into four equal parts. Q1 is the value below which 25% of the data falls, and Q3 is the value below which 75% of the data falls.

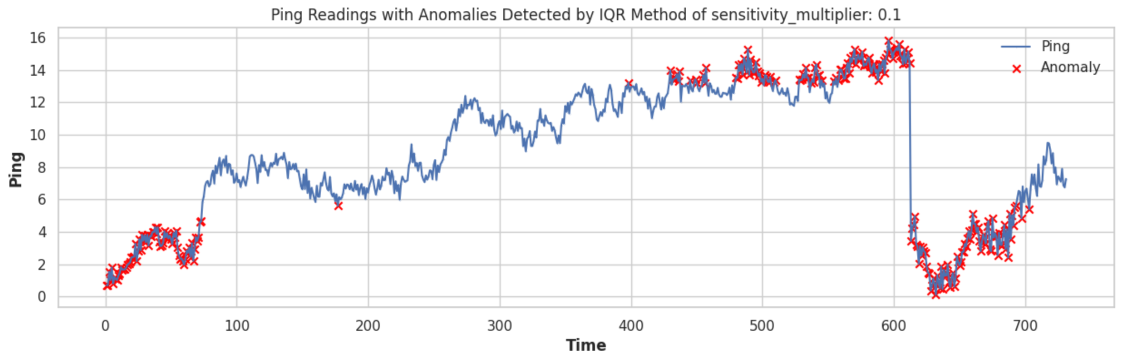
- The range between Q1 and Q3, which contains the middle 50% of the data. It measures the spread of the central portion of the data.
- Adjusts the width of the anomaly detection bounds. A smaller multiplier results in a narrower range, making it more sensitive to anomalies, while a larger multiplier broadens the range

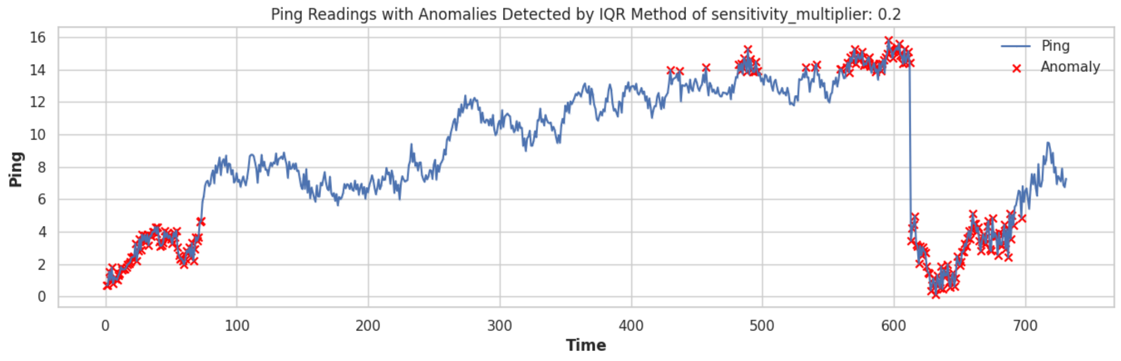
```
[89]: Q1 = df['ping'].quantile(0.25)
Q3 = df['ping'].quantile(0.75)
IQR = Q3 - Q1

sensitivity_multiplier = 0.7
lower_bound = Q1 - sensitivity_multiplier * IQR
upper_bound = Q3 + sensitivity_multiplier * IQR
df['anomaly_iqr'] = ((df['ping'] < lower_bound) | (df['ping'] > upper_bound))

plt.figure(figsize=(15, 4))
plt.plot(df['t'], df['ping'], marker='o', label='Ping',markersize=0.1)
plt.scatter(df['t'][df['anomaly_iqr']], df['ping'][df['anomaly_iqr']],␣
 ↪color='red', label='Anomaly', marker='x')
plt.title('Ping Readings with Anomalies Detected by IQR Method')
plt.xlabel('Time')
plt.ylabel('Ping')
plt.legend()
plt.grid(True)
plt.show()
```

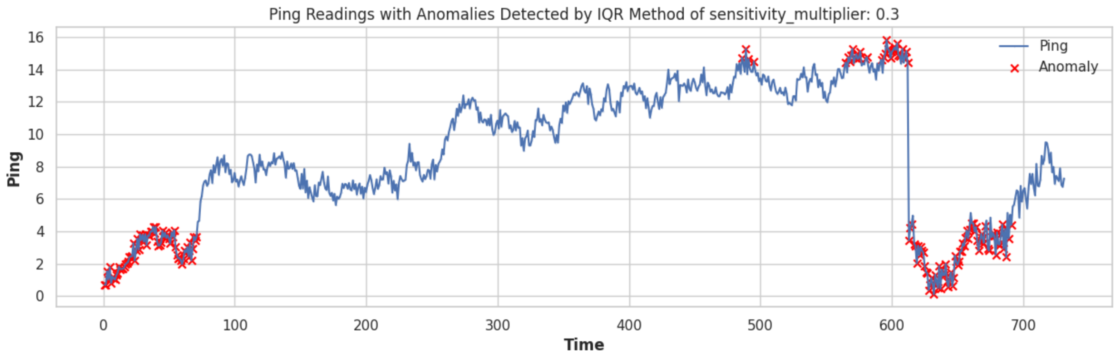


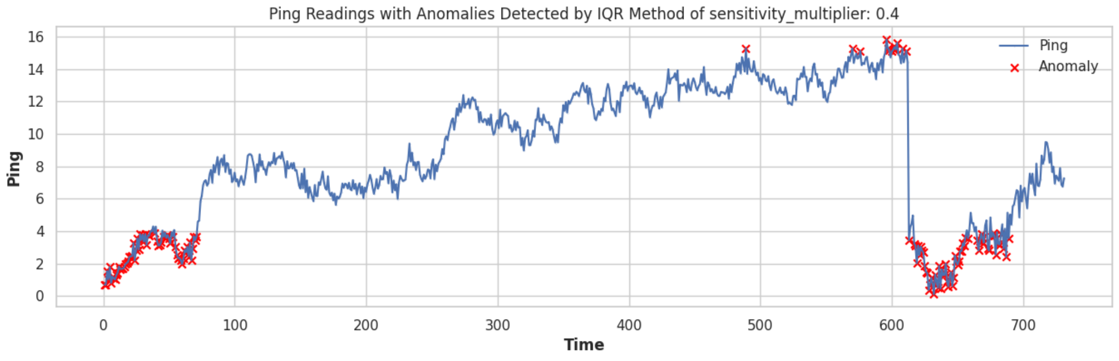
6.1.1 Anomolies base on different senstivity of IQR Method

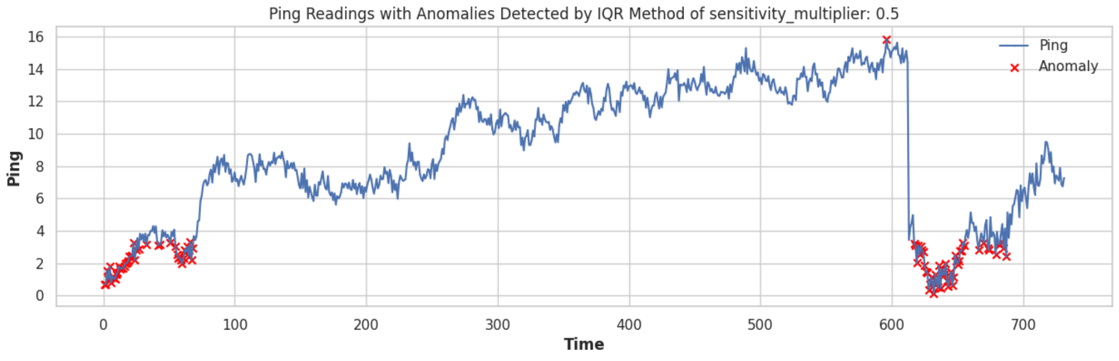
```
[96]: for sensitivity_multiplier in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]:

    Q1 = df['ping'].quantile(0.25)
    Q3 = df['ping'].quantile(0.75)
    IQR = Q3 - Q1

    # sensitivity_multiplier = 0.1
    lower_bound = Q1 - sensitivity_multiplier * IQR
    upper_bound = Q3 + sensitivity_multiplier * IQR
    df[f'anomaly_iqr_{sensitivity_multiplier}'] = ((df['ping'] < lower_bound) |␣
↪(df['ping'] > upper_bound))

    plt.figure(figsize=(15, 4))
    plt.plot(df['t'], df['ping'], marker='o', label='Ping',markersize=0.1)
    plt.scatter(df['t'][df[f'anomaly_iqr_{sensitivity_multiplier}']],␣
↪df['ping'][df[f'anomaly_iqr_{sensitivity_multiplier}']], color='red',␣
↪label='Anomaly', marker='x')
    plt.title(f'Ping Readings with Anomalies Detected by IQR Method of␣
↪sensitivity_multiplier: {sensitivity_multiplier}')
    plt.xlabel('Time')
    plt.ylabel('Ping')
    plt.legend()
    plt.grid(True)
    plt.show()
    time.sleep(1)
    plt.close()
    # break
```

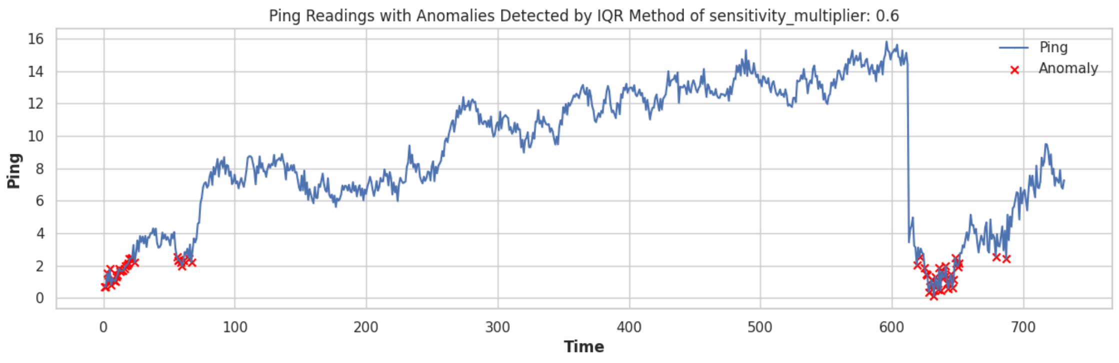


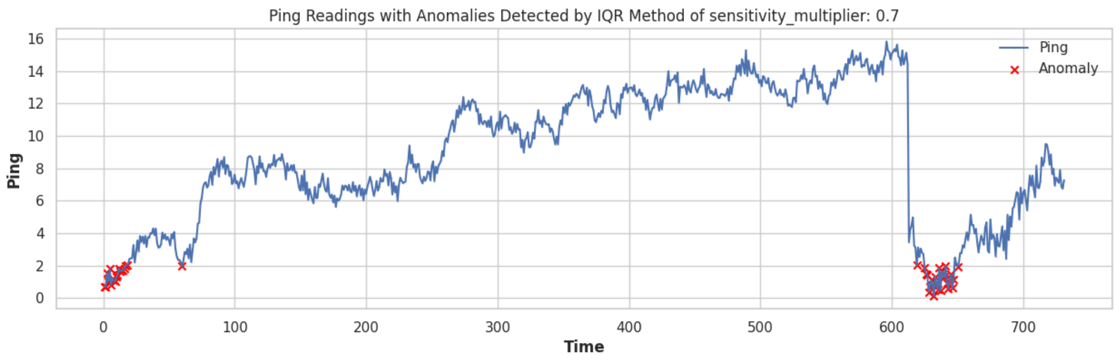


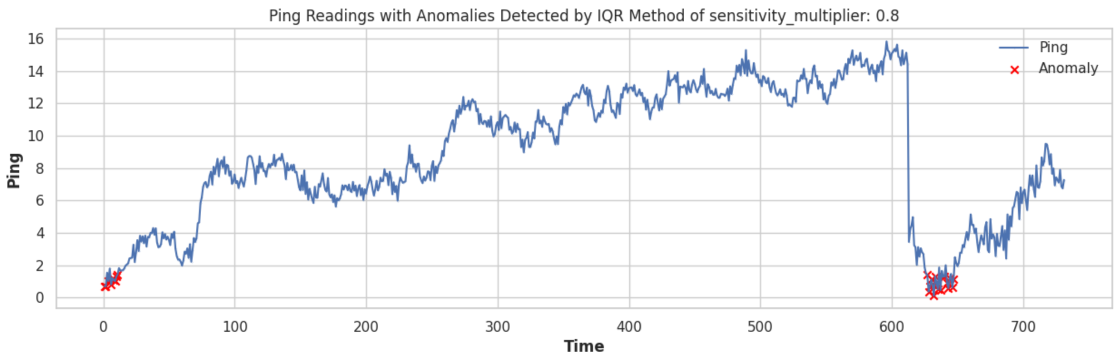

Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.3
Ping
Anomaly
Ping
Time


Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.4
Ping
Anomaly
Ping
Time


Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.5
Ping
Anomaly
Ping
Time


Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.6
Ping
Anomaly
Time
Ping


Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.7
Ping
Anomaly
Time
Ping


Ping Readings with Anomalies Detected by IQR Method of sensitivity_multiplier: 0.8
Ping
Anomaly
Time
Ping

```
[ ]:
```

```
[ ]:
```