7_FEB_EDA_interpretation_of_plots

February 9, 2025

1 EDA Lab

1.1 7 February

2 Tufan Kundu

2.1 Reg No. 24MDT0184

2.2 Interpretation of the Plots

2.3 Titanic dataset

2.4 Loading the dataset

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

titanic = pd.read_csv(r"D:\study␣
↪material\VIT_Data_Science\Winter_Sem\Exploratory Data Analysis␣
↪Lab\7_Feb\titanic_new.csv")
```

[2]:
```
titanic
```

[2]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …      …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                          Name     Sex   Age  SibSp  \
```

```
0                              Braund, Mr. Owen Harris    male  22.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …    …      …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …       …    …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[3]:
```
titanic.shape
```

[3]:
```
(891, 12)
```

2.5 Checking for missing values

[4]:
```
titanic.isnull().sum().sort_values(ascending=False)
```

[4]:
```
Cabin          687
Age            177
Embarked         2
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
SibSp            0
Parch            0
Ticket           0
Fare             0
```

```
dtype: int64
```

2.5.1 Inference:

- cabin column has the most missing values followed by Age and Embarked

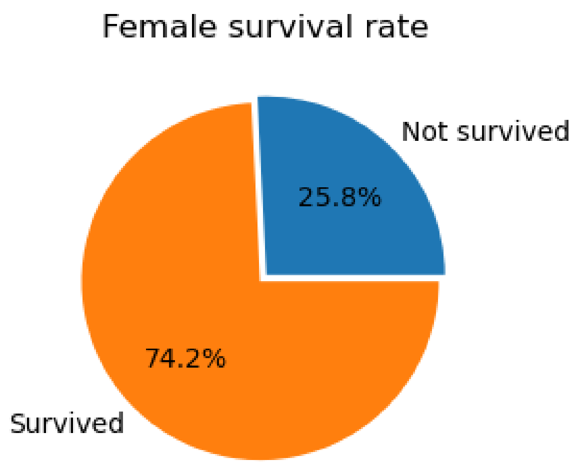
[5]:
```
## Percentage of women survived

women = titanic.loc[titanic.Sex == 'female']['Survived']
rate_women = sum(women)/len(women)
rate_women

## percentage of men survived

men = titanic.loc[titanic.Sex == 'male']['Survived']
rate_men = sum(men)/len(men)

print(f"{round(rate_women,3)*100}% women survived")
print(f"{round(rate_men,3)*100}% men survived")
```

```
74.2% women survived
18.9% men survived
```

2.5.2 Inference:

- Women had a significantly higher chance of survival due to the “women and children first” policy.

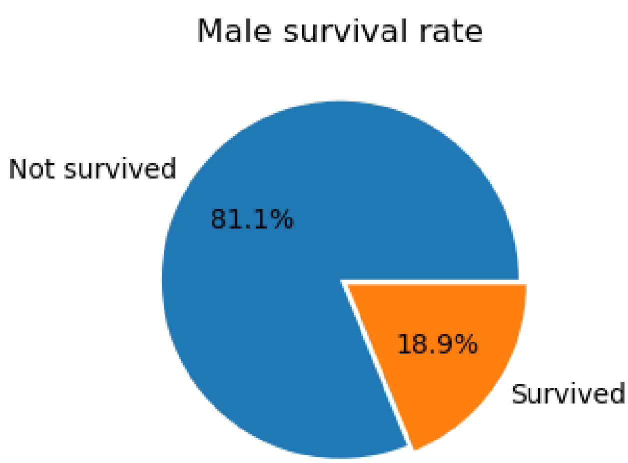
[6]:
```
titanic[titanic['Sex'] == 'female'].Survived.groupby(titanic.Survived).count().
 ↪plot(kind='pie',
figsize=(3, 6),explode=[0,0.05],autopct='%1.1f%%',labels=["Not␣
 ↪survived","Survived"])
plt.title("Female survival rate")
plt.ylabel("")
plt.show()
```



2.5.3 Inference:

- 74.2% of women survived, indicating majority of women survived.

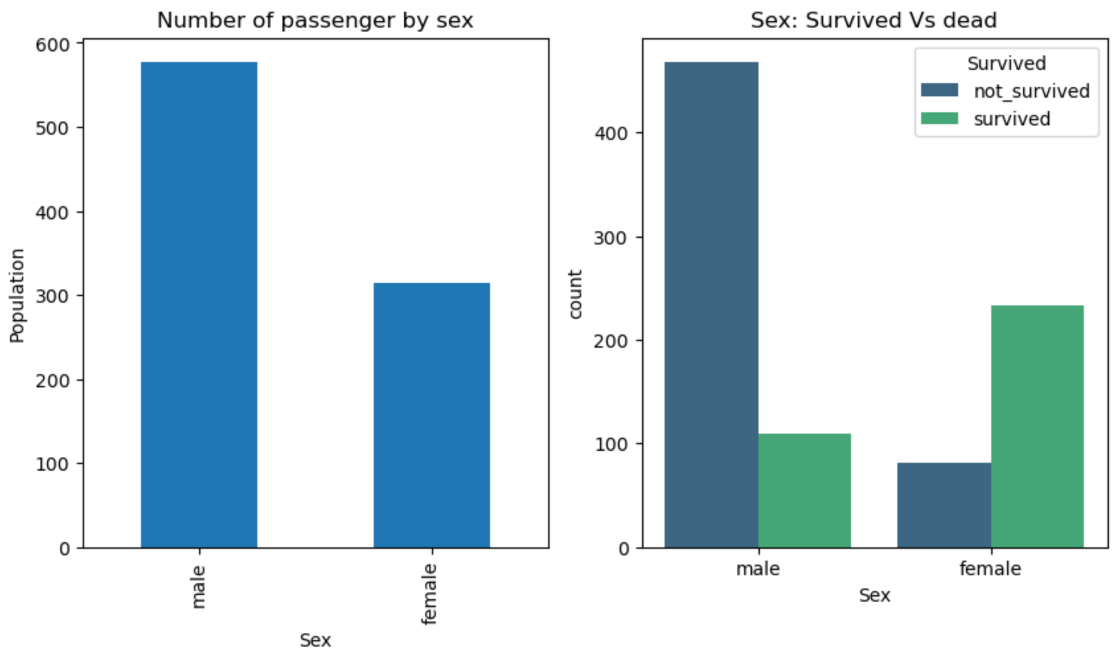
[7]:
```
titanic[titanic['Sex'] == 'male'].Survived.groupby(titanic.Survived).count().
 ↪plot(kind='pie',
figsize=(3, 6),explode=[0,0.05],autopct='%1.1f%%',labels=["Not␣
 ↪survived","Survived"])
plt.title("Male survival rate")
plt.ylabel("")
plt.show()
```



2.5.4 Inference:

- only 18.9% of men survived, indicating majority of men could not survive.

[8]:
```
titanic['Survived'] = titanic['Survived'].map({0:"not_survived",1:"survived"})
plt.figure(figsize = (10,5))
plt.subplot(1,2,1)
titanic['Sex'].value_counts().plot.bar()
plt.title("Number of passenger by sex")
plt.ylabel("Population")
plt.subplot(1,2,2)
sns.countplot(x ="Sex",data = titanic, hue="Survived",palette='viridis')
plt.title("Sex: Survived Vs dead")
plt.show()
```

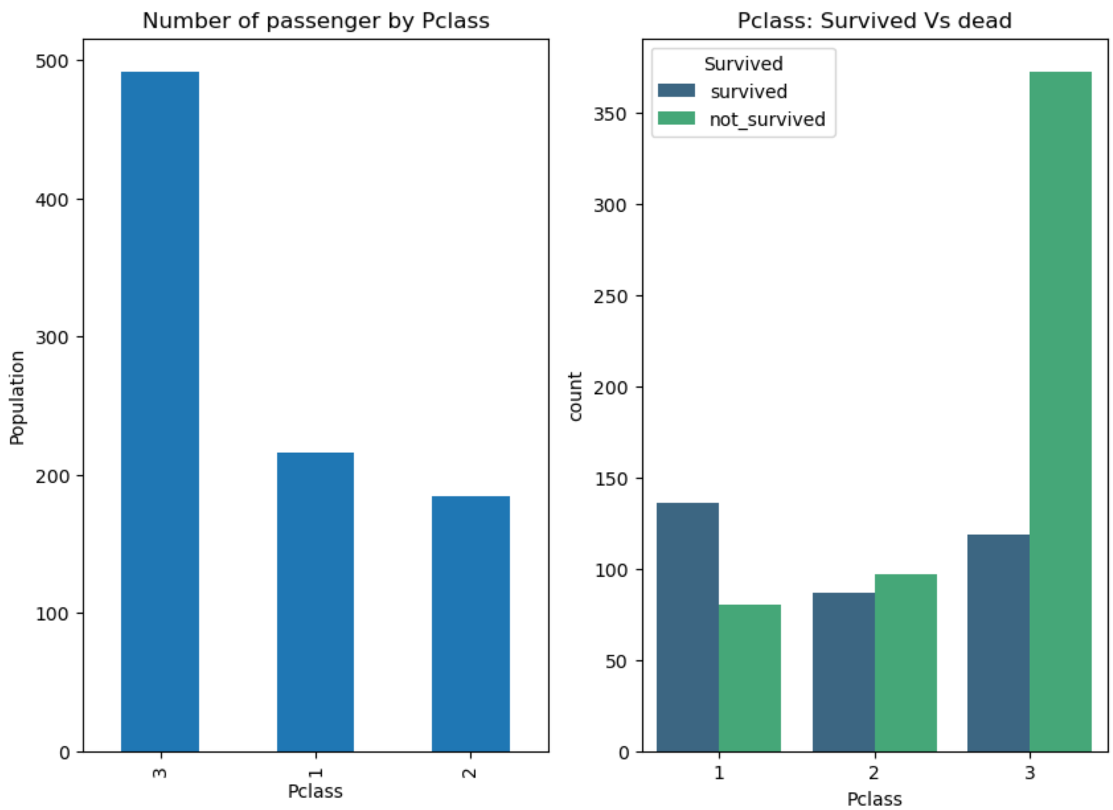


2.5.5 Inference:

- Though more males than females onboarded, still majority of males did not survive.

[9]:
```
plt.figure(figsize = (10,7))
plt.subplot(1,2,1)
titanic['Pclass'].value_counts().plot.bar()
plt.title("Number of passenger by Pclass")
plt.ylabel("Population")
plt.subplot(1,2,2)
sns.countplot(x ="Pclass",data = titanic, hue="Survived",palette='viridis')
plt.title("Pclass: Survived Vs dead")
```

```
plt.show()
```



2.5.6 Inference:

- Most people were from pclass 3
- Survival rate was the lowest in Pclass 3
- Pclass 1 had the highest survival rate (higher-class passengers had better access to lifeboats).

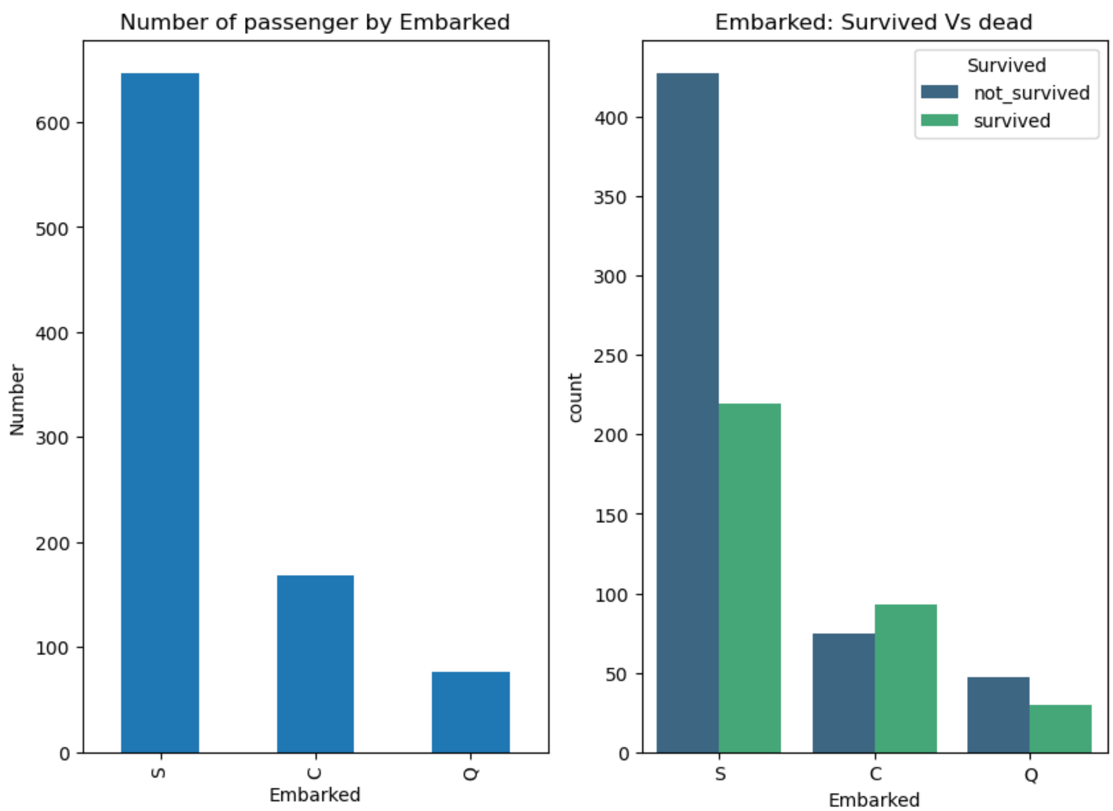
[10]:
```
titanic["Embarked"] = titanic["Embarked"].fillna("S")
titanic
```

[10]:
```
     PassengerId      Survived  Pclass  \
0              1  not_survived       3
1              2      survived       1
2              3      survived       3
3              4      survived       1
4              5  not_survived       3
..           …             …      …
886          887  not_survived       2
887          888      survived       1
888          889  not_survived       3
```

```
889          890      survived       1
890          891  not_survived       3

                                                   Name     Sex   Age  SibSp  \
0                               Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                                Heikkinen, Miss. Laina  female  26.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                              Allen, Mr. William Henry    male  35.0      0
..                                                  …     …   …    …
886                               Montvila, Rev. Juozas    male  27.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                               Behr, Mr. Karl Howell    male  26.0      0
890                                 Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …       …     …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[11]:

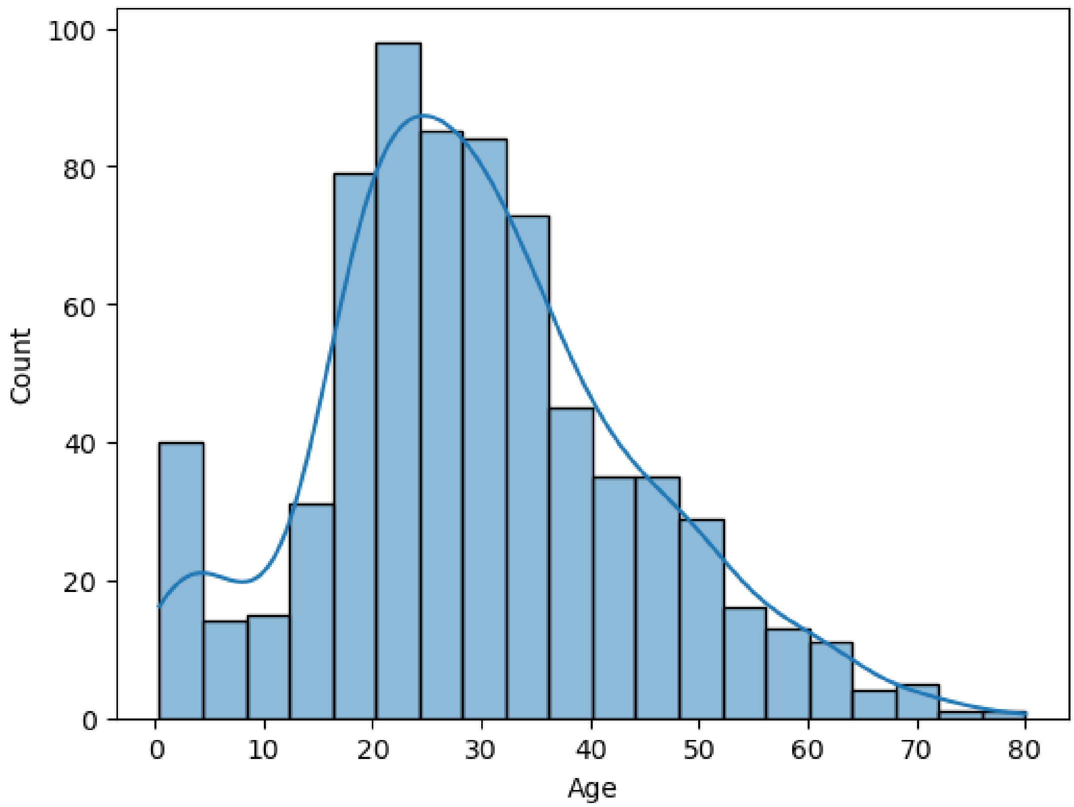
```
plt.figure(figsize = (10,7))
plt.subplot(1,2,1)
titanic['Embarked'].value_counts().plot.bar()
plt.title("Number of passenger by Embarked")
plt.ylabel("Number")
plt.subplot(1,2,2)
sns.countplot(x ="Embarked",data = titanic, hue="Survived",palette='viridis')
plt.title("Embarked: Survived Vs dead")
plt.show()
```



2.5.7 Inference:

- Most passengers embarked from Southampton (S).
- Embarked from "C" (Cherbourg) had a higher survival rate (possibly more first-class passengers).
- Embarked from "S" (Southampton) had the lowest survival rate (many third-class passengers).

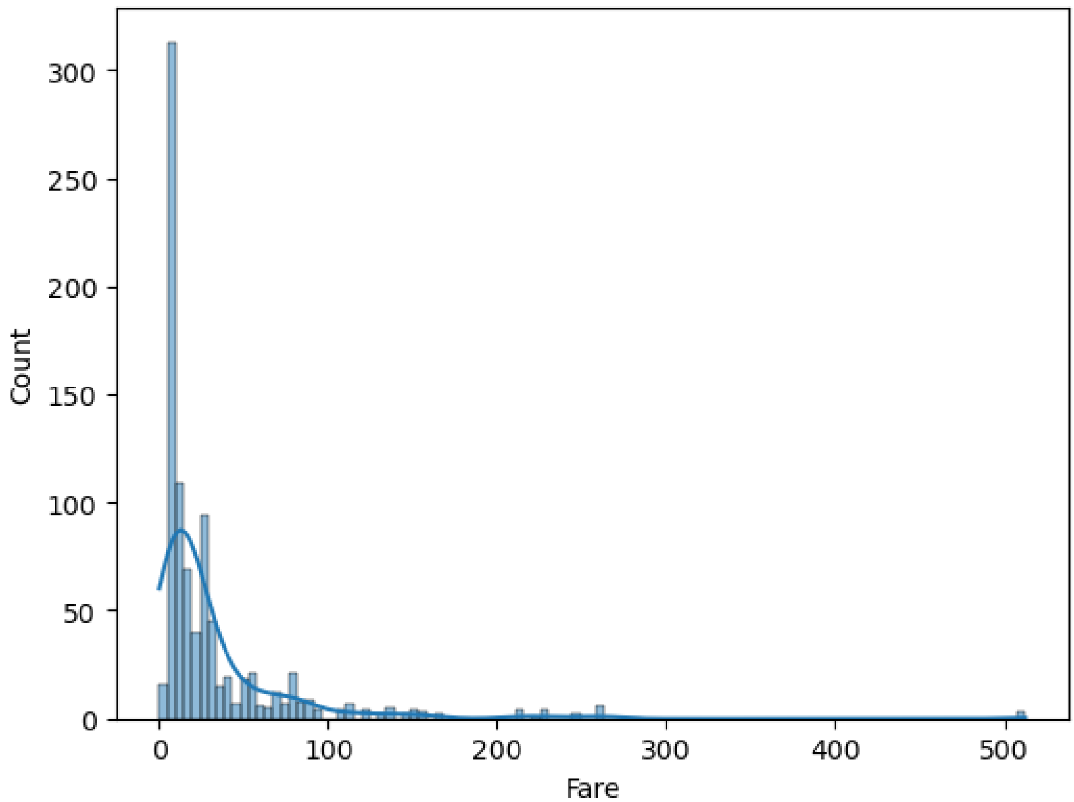
[12]:
```
sns.histplot(titanic['Age'].dropna(), kde = True)
plt.show()
```



2.5.8 Inference:

- Most passengers were between 20-40 years old.

[13]:
```
sns.histplot(titanic['Fare'], kde=True)
plt.show()
```

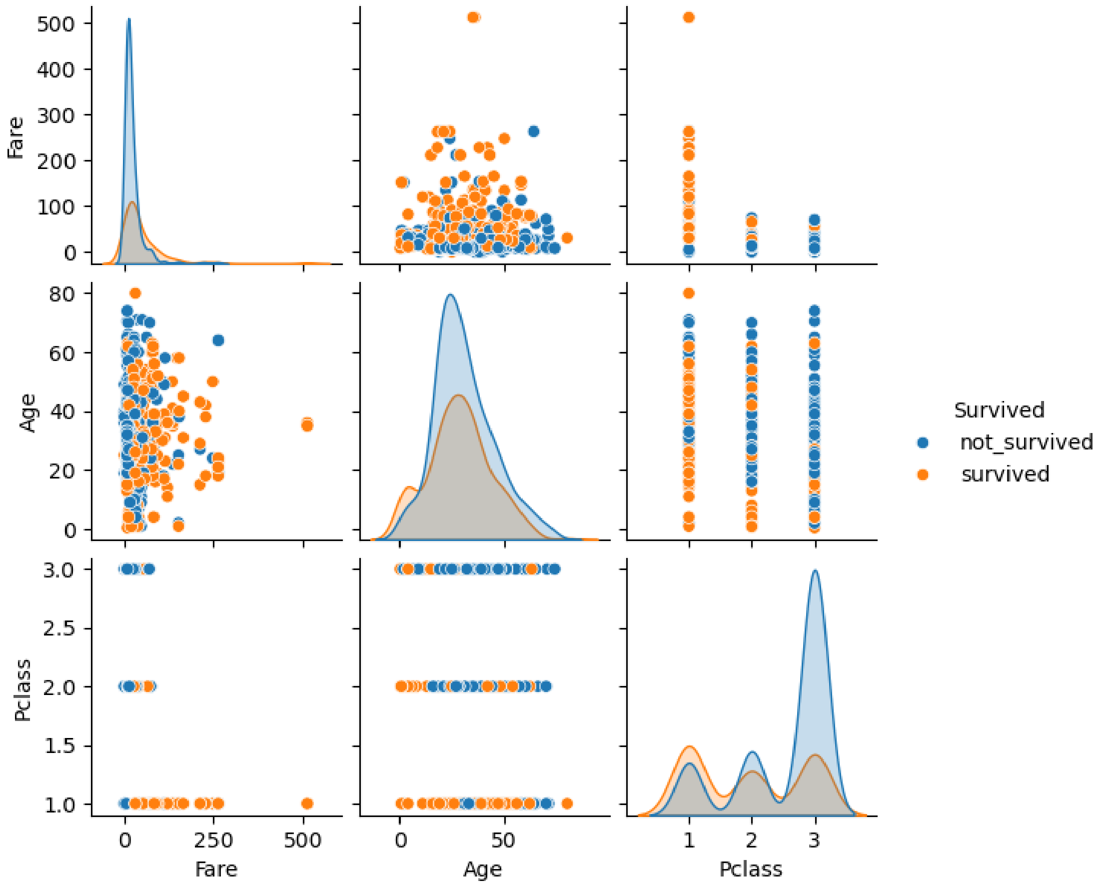


2.5.9 Inference:

- Right-skewed distribution → Most fares were low, but a few passengers paid very high fares.
- Indicates wealthier passengers in first-class paid significantly more.

2.5.10 Multivariate analysis

[14]:
```
sns.pairplot(titanic,height=2,vars = [ 'Fare','Age','Pclass'], hue="Survived")
plt.show()
```



2.5.11 Inference:

- Higher fares were associated with higher survival rates

2.6 Correlation table with heatmap

[15]:
```python
titanic['Embarked'] = titanic['Embarked'].map({"S":1, "C":2,"Q":2,"NaN":0})
Tcorrelation = titanic.corr(method='pearson', numeric_only=True)
Tcorrelation
```

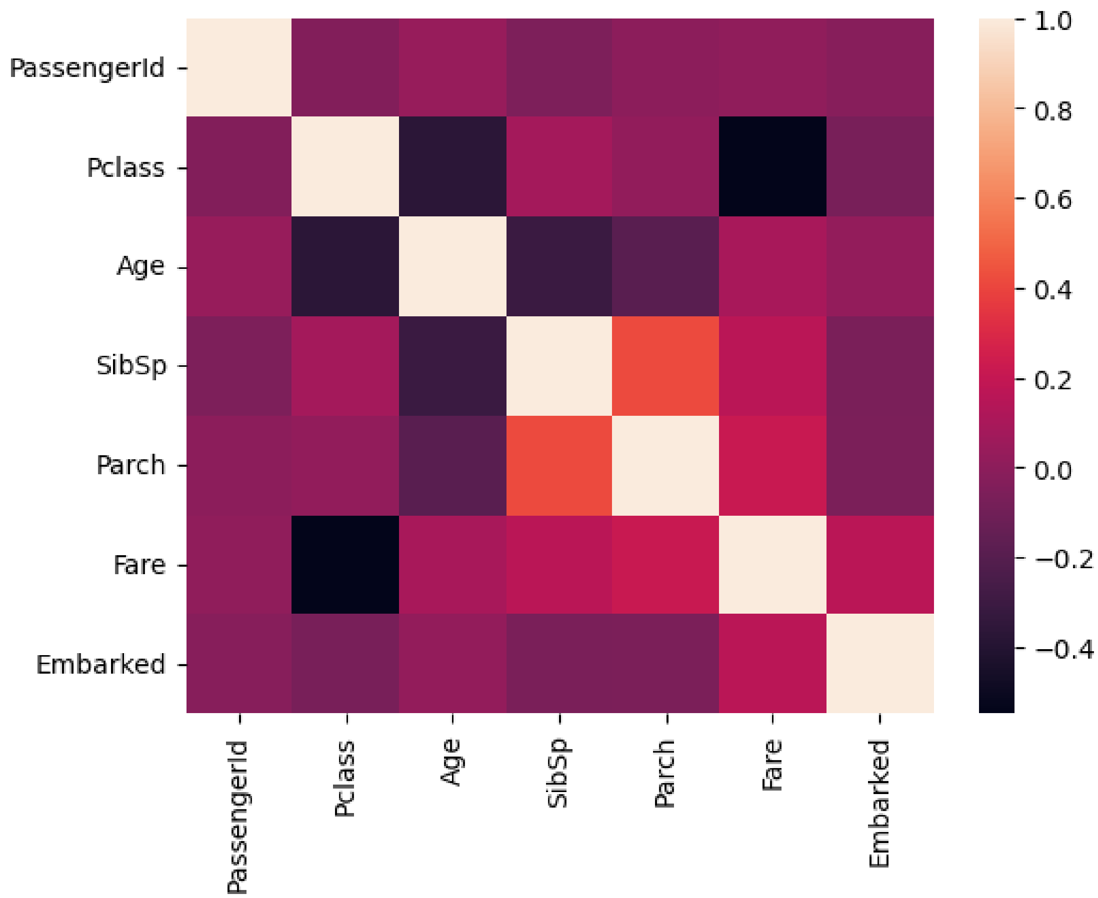
[15]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	\
PassengerId	1.000000	-0.035144	0.036847	-0.057527	-0.001652	0.012658	
Pclass	-0.035144	1.000000	-0.369226	0.083081	0.018443	-0.549500	
Age	0.036847	-0.369226	1.000000	-0.308247	-0.189119	0.096067	
SibSp	-0.057527	0.083081	-0.308247	1.000000	0.414838	0.159651	
Parch	-0.001652	0.018443	-0.189119	0.414838	1.000000	0.216225	
Fare	0.012658	-0.549500	0.096067	0.159651	0.216225	1.000000	
Embarked	-0.022204	-0.074053	0.023233	-0.068734	-0.060814	0.162184	

```
             Embarked
PassengerId -0.022204
Pclass      -0.074053
Age          0.023233
SibSp       -0.068734
Parch       -0.060814
Fare         0.162184
Embarked     1.000000
```

[16]:
```
sns.heatmap(Tcorrelation,xticklabels=Tcorrelation.columns,
            yticklabels=Tcorrelation.columns)
```

[16]: `<Axes: >`



2.6.1 Inference:

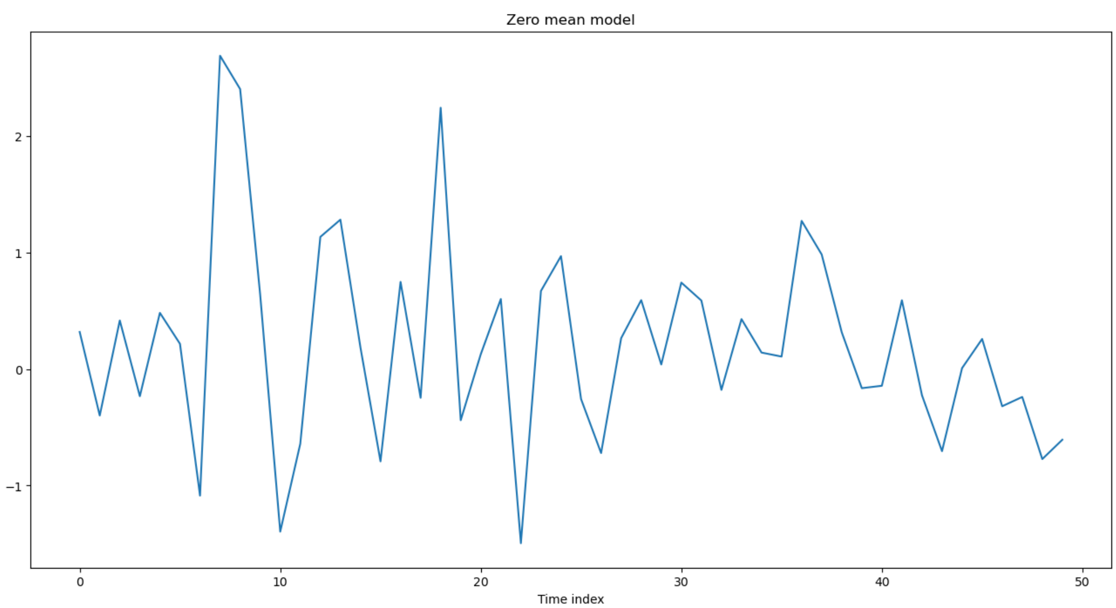
- Fare and Pclass has strongest negative correlation
- Age had little correlation with survival

3 Time series analysis

[17]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

zero_mean_series = np.random.normal(loc = 0.0, scale = 1., size = 50)
zero_mean_series
```

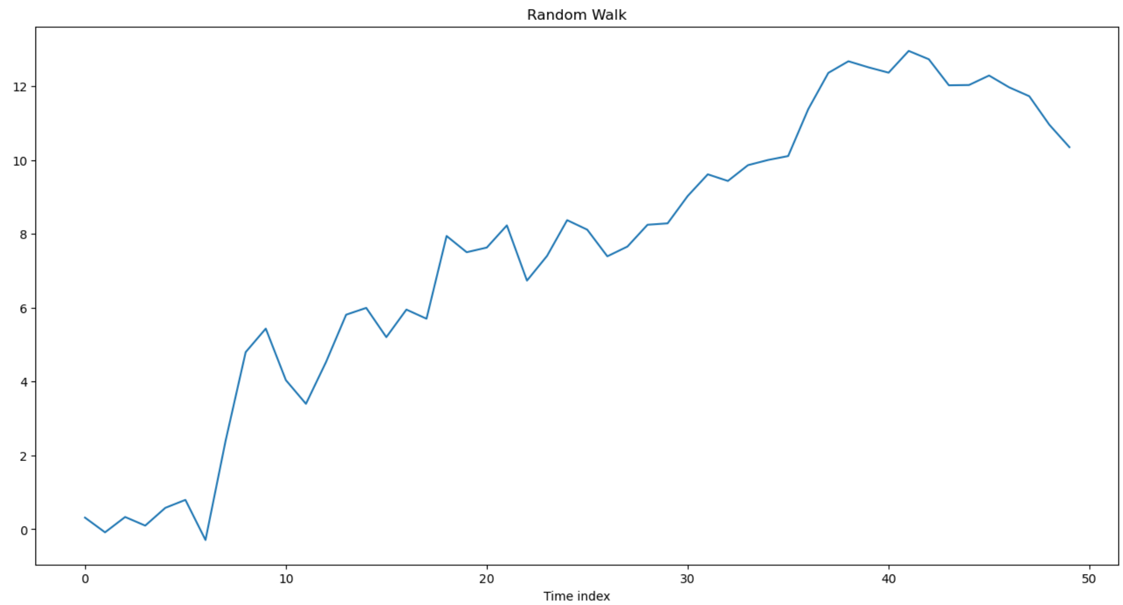
[17]:
```
array([ 0.31725832, -0.39958418,  0.41585006, -0.23364894,  0.48110827,
        0.21572602, -1.08674158,  2.68668795,  2.40092511,  0.63678105,
       -1.39538987, -0.64133054,  1.13271997,  1.280581  ,  0.1842885 ,
       -0.7927226 ,  0.74674999, -0.24743869,  2.24069045, -0.4399123 ,
        0.12639721,  0.60011777, -1.49506548,  0.66911955,  0.96791617,
       -0.25789412, -0.7213727 ,  0.26417692,  0.59009659,  0.03823919,
        0.74082901,  0.58670692, -0.17957267,  0.42757738,  0.14048686,
        0.10622285,  1.27041735,  0.98185353,  0.31460675, -0.16486628,
       -0.14424799,  0.58920466, -0.22556404, -0.70532822,  0.0071992 ,
        0.25717507, -0.31991982, -0.24006193, -0.77315687, -0.60716803])
```

[18]:
```
plt.figure(figsize=(16, 8))
sns.lineplot(data=zero_mean_series)
plt.title('Zero mean model')
plt.xlabel('Time index')
plt.show()
```



[19]:
```
random_walk = np.cumsum(zero_mean_series)
```

[20]:
```
plt.figure(figsize=(16, 8))
sns.lineplot(data=random_walk)
plt.title('Random Walk')
plt.xlabel('Time index')
plt.show()
```



3.1 loading a dataset

[21]:
```
df = pd.read_csv(r"D:\study material\VIT_Data_Science\Winter_Sem\Exploratory␣
↪Data Analysis Lab\7_Feb\opsd_germany_daily.csv")
df
```

[21]:

	Date	Consumption	Wind	Solar	Wind+Solar
0	2006-01-01	1069.18400	NaN	NaN	NaN
1	2006-01-02	1380.52100	NaN	NaN	NaN
2	2006-01-03	1442.53300	NaN	NaN	NaN
3	2006-01-04	1457.21700	NaN	NaN	NaN
4	2006-01-05	1477.13100	NaN	NaN	NaN
…	…	…	…	…	…
4378	2017-12-27	1263.94091	394.507	16.530	411.037
4379	2017-12-28	1299.86398	506.424	14.162	520.586
4380	2017-12-29	1295.08753	584.277	29.854	614.131
4381	2017-12-30	1215.44897	721.247	7.467	728.714
4382	2017-12-31	1107.11488	721.176	19.980	741.156

```
[4383 rows x 5 columns]
```

[22]:
```
df.shape
```

[22]:
```
(4383, 5)
```

[23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4383 entries, 0 to 4382
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         4383 non-null   object
 1   Consumption  4383 non-null   float64
 2   Wind         2920 non-null   float64
 3   Solar        2188 non-null   float64
 4   Wind+Solar   2187 non-null   float64
dtypes: float64(4), object(1)
memory usage: 171.3+ KB
```

[24]:
```
#convert object to datetime format
df['Date'] = pd.to_datetime(df['Date'])
df
```

[24]:

	Date	Consumption		Wind	Solar	Wind+Solar
0	2006-01-01	1069.18400		NaN	NaN	NaN
1	2006-01-02	1380.52100		NaN	NaN	NaN
2	2006-01-03	1442.53300		NaN	NaN	NaN
3	2006-01-04	1457.21700		NaN	NaN	NaN
4	2006-01-05	1477.13100		NaN	NaN	NaN
…	…	…	…	…	…	
4378	2017-12-27	1263.94091		394.507	16.530	411.037
4379	2017-12-28	1299.86398		506.424	14.162	520.586
4380	2017-12-29	1295.08753		584.277	29.854	614.131
4381	2017-12-30	1215.44897		721.247	7.467	728.714
4382	2017-12-31	1107.11488		721.176	19.980	741.156

```
[4383 rows x 5 columns]
```

[25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4383 entries, 0 to 4382
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         4383 non-null   datetime64[ns]
 1   Consumption  4383 non-null   float64
```

```
 2   Wind          2920 non-null   float64
 3   Solar         2188 non-null   float64
 4   Wind+Solar    2187 non-null   float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 171.3 KB
```

Now that the Date column is in correct datatype, let's set it as the DataFrame's index because in time series analysis the index column is always datetime column.

[26]:
```
df = df.set_index('Date')
```

[27]:
```
df.tail()
```

[27]:

Date	Consumption	Wind	Solar	Wind+Solar
2017-12-27	1263.94091	394.507	16.530	411.037
2017-12-28	1299.86398	506.424	14.162	520.586
2017-12-29	1295.08753	584.277	29.854	614.131
2017-12-30	1215.44897	721.247	7.467	728.714
2017-12-31	1107.11488	721.176	19.980	741.156

[28]:
```
df.index
```

[28]:
```
DatetimeIndex(['2006-01-01', '2006-01-02', '2006-01-03', '2006-01-04',
               '2006-01-05', '2006-01-06', '2006-01-07', '2006-01-08',
               '2006-01-09', '2006-01-10',
               …
               '2017-12-22', '2017-12-23', '2017-12-24', '2017-12-25',
               '2017-12-26', '2017-12-27', '2017-12-28', '2017-12-29',
               '2017-12-30', '2017-12-31'],
              dtype='datetime64[ns]', name='Date', length=4383, freq=None)
```

[29]:
```
## Adding columns with year, month and weekday name
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Weekday Name'] = df.index.day_name()
```

[30]:
```
## displaying a random sample of 5 rows

df.sample(5,random_state=42)
```

[30]:

Date	Consumption	Wind	Solar	Wind+Solar	Year	Month	Weekday Name
2007-11-02	1408.209	NaN	NaN	NaN	2007	11	Friday
2012-08-14	1269.779	64.136	153.658	217.794	2012	8	Tuesday
2007-08-20	1373.403	NaN	NaN	NaN	2007	8	Monday
2013-03-14	1420.149	28.595	62.718	91.313	2013	3	Thursday
2009-10-27	1405.611	NaN	NaN	NaN	2009	10	Tuesday

[31]:
```
df.loc['2015-10-02']
```

[31]:
```
Consumption     1391.05
Wind             81.229
Solar           160.641
Wind+Solar       241.87
Year               2015
Month                10
Weekday Name     Friday
Name: 2015-10-02 00:00:00, dtype: object
```

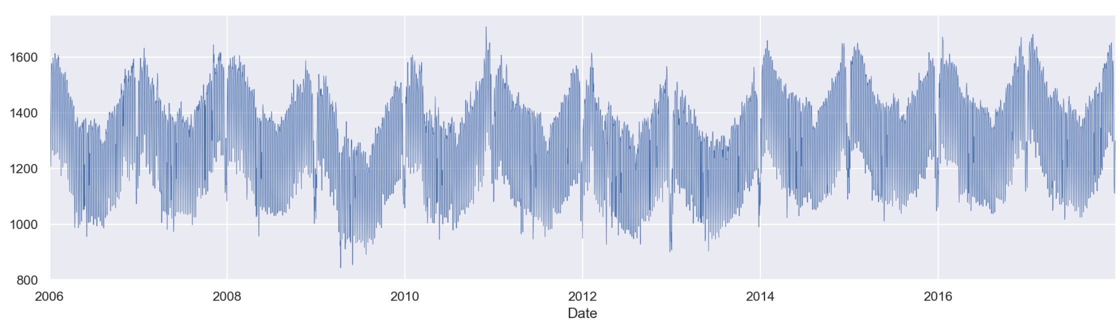
[32]:
```
df.loc['2017-01-01':'2017-12-30']
```

[32]:

	Consumption	Wind	Solar	Wind+Solar	Year	Month	Weekday Name
Date							
2017-01-01	1130.41300	307.125	35.291	342.416	2017	1	Sunday
2017-01-02	1441.05200	295.099	12.479	307.578	2017	1	Monday
2017-01-03	1529.99000	666.173	9.351	675.524	2017	1	Tuesday
2017-01-04	1553.08300	686.578	12.814	699.392	2017	1	Wednesday
2017-01-05	1547.23800	261.758	20.797	282.555	2017	1	Thursday
…	…	…	…	…	…	…	…
2017-12-26	1130.11683	717.453	30.923	748.376	2017	12	Tuesday
2017-12-27	1263.94091	394.507	16.530	411.037	2017	12	Wednesday
2017-12-28	1299.86398	506.424	14.162	520.586	2017	12	Thursday
2017-12-29	1295.08753	584.277	29.854	614.131	2017	12	Friday
2017-12-30	1215.44897	721.247	7.467	728.714	2017	12	Saturday

[364 rows x 7 columns]

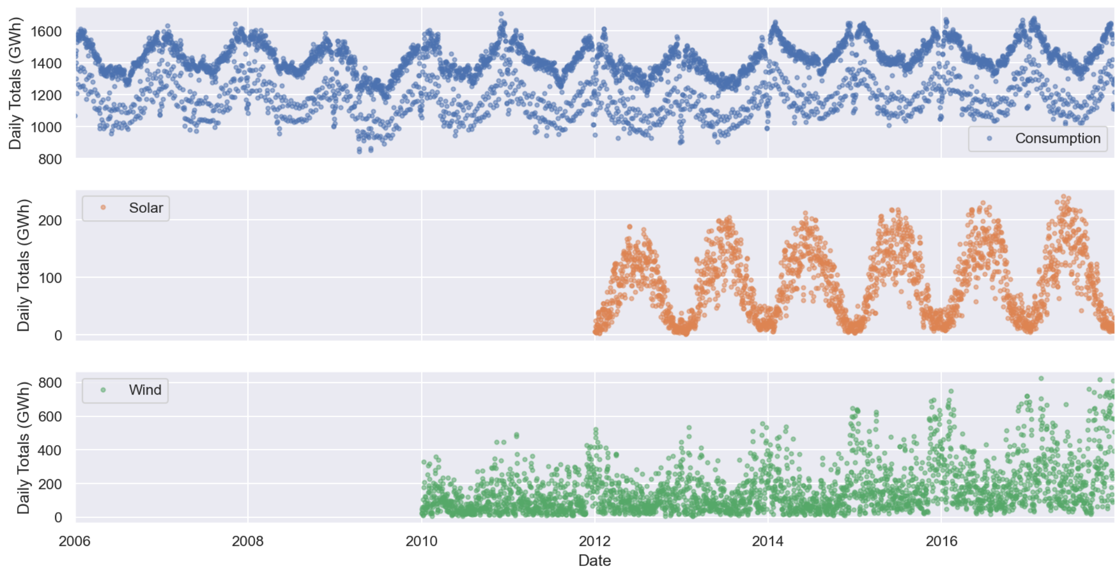
[33]:
```
# Visualization for Time series analysis
sns.set_theme(rc={'figure.figsize':(16, 4)})
plt.rcParams['figure.dpi'] = 150
df['Consumption'].plot(linewidth = 0.4)
plt.show()
```



3.1.1 Inference:

- Electricity consumption varies over time.
- Periodic spikes suggest seasonal trends.

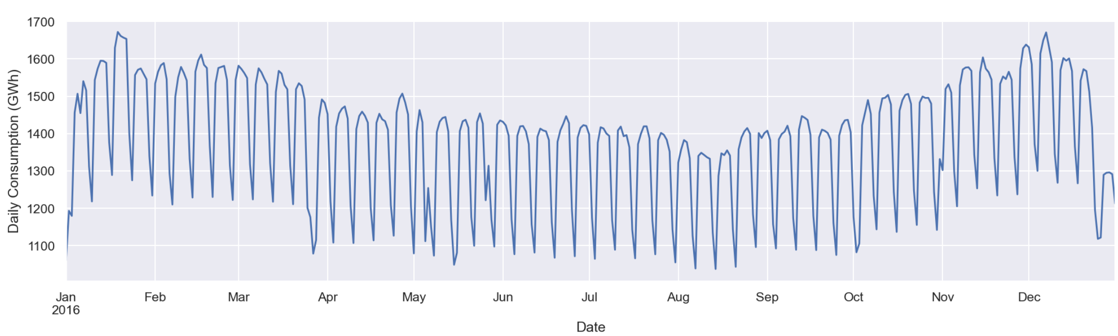
[34]:
```
cols_to_plot = ['Consumption', 'Solar', 'Wind']
axes = df[cols_to_plot].plot(marker='.', alpha=0.5,␣
↪linestyle='None',figsize=(14, 7), subplots=True,grid=True)
for ax in axes:
    ax.set_ylabel('Daily Totals (GWh)')
```



3.1.2 Inference:

- Solar power peaks in summer (more sunshine).

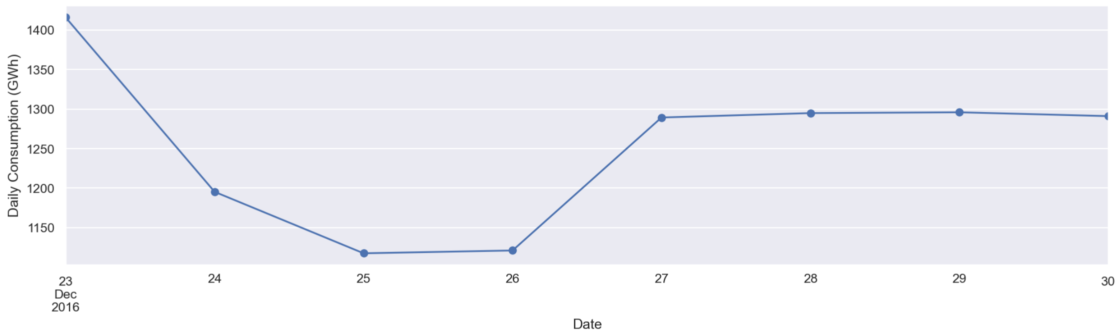
[35]:
```
df.loc['2016', 'Consumption'].plot()
plt.ylabel('Daily Consumption (GWh)')
plt.show()
```



3.1.3 Inference:

- Consumption of electricity is the most in winter

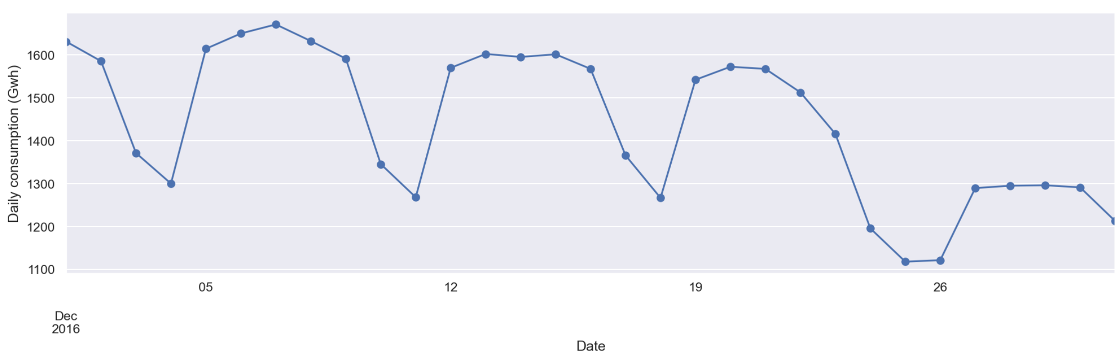
[36]:
```
df.loc['2016-12-23':'2016-12-30', 'Consumption'].plot(marker='o', linestyle='-')
plt.ylabel('Daily Consumption (GWh)')
plt.show()
```



3.1.4 Inference:

- shows the consumption pattern of electricity from '2016-12-23' to '2016-12-30'

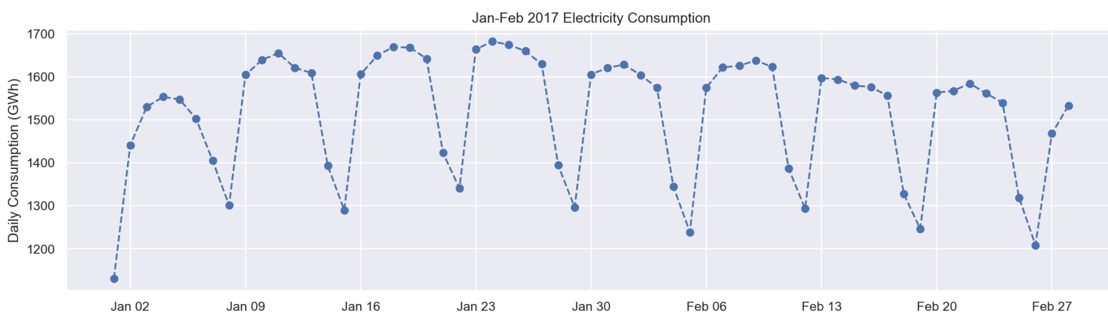
[37]:
```
df.loc['2016-12','Consumption'].plot(marker = 'o', linestyle = '-')
plt.ylabel("Daily consumption (Gwh)")
plt.show()
```



3.1.5 Inference:

- shows the weekly pattern of electricity consumption, indicating consumption decreases in the weekends

[38]:
```
# import dates module from matplotlib
import matplotlib.dates as mdates

# plot graph
fig, ax = plt.subplots()

ax.plot(df.loc['2017-01':'2017-02', 'Consumption'], marker='o', linestyle='--')
ax.set_ylabel('Daily Consumption (GWh)')
ax.set_title('Jan-Feb 2017 Electricity Consumption')

# to set x-axis major ticks to weekly interval, on Mondays
ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MONDAY))
# to set format for x-tick labels as 3-letter month name and day number
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %d'))
```



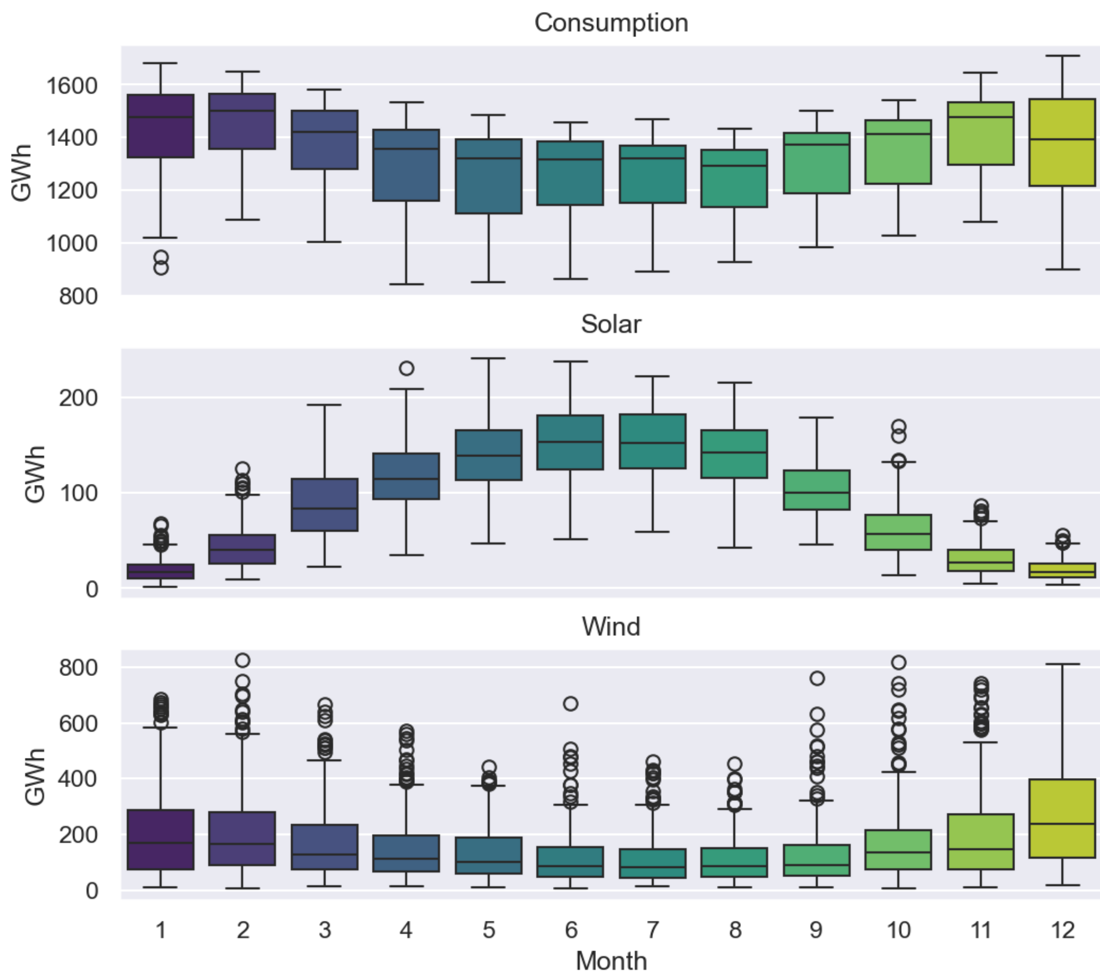
3.1.6 Inference:

- Shows a smoother version of the trends by removing daily fluctuations.

[39]:
```
fig, axes = plt.subplots(3, 1, figsize=(8, 7), sharex=True)
for name, ax in zip(['Consumption', 'Solar', 'Wind'], axes):
  sns.boxplot(data=df, x='Month', y=name, ax=ax, palette='viridis')
  ax.set_ylabel('GWh')
  ax.set_title(name)
  if ax != axes[-1]:
    ax.set_xlabel('')
```

```
C:\Users\TUFAN\AppData\Local\Temp\ipykernel_12948\2142063830.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=df, x='Month', y=name, ax=ax, palette='viridis')
C:\Users\TUFAN\AppData\Local\Temp\ipykernel_12948\2142063830.py:3:
```

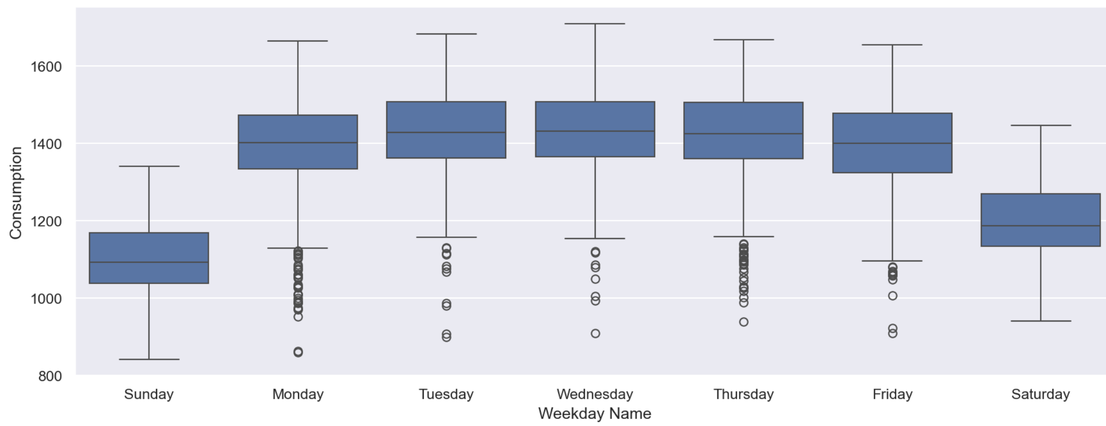
```
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=df, x='Month', y=name, ax=ax, palette='viridis')
C:\Users\TUFAN\AppData\Local\Temp\ipykernel_12948\2142063830.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=df, x='Month', y=name, ax=ax, palette='viridis')
```



3.1.7 Inference:

- Higher consumption in winter months (possibly due to heating).
- Lower consumption in summer months.

[40]:
```
plt.figure(figsize=(14,5))
sns.boxplot(data=df, x='Weekday Name', y='Consumption')
plt.show()
```



3.1.8 Inference:

- Consumption is lowest on the weekends

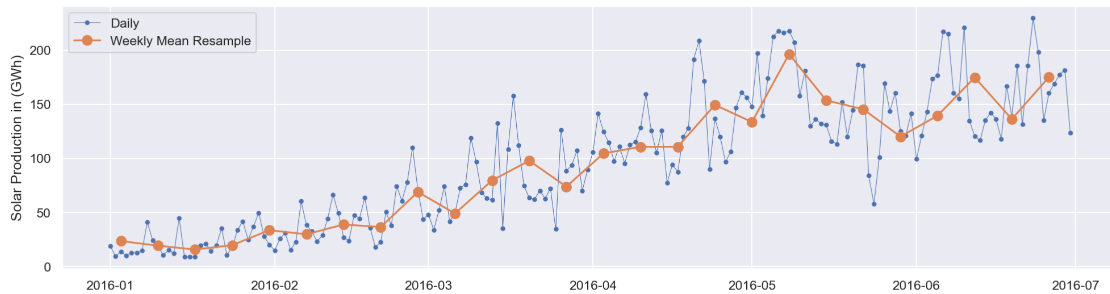
[41]:
```
columns = ['Consumption', 'Wind', 'Solar', 'Wind+Solar']

power_weekly_mean = df[columns].resample('W').mean()
power_weekly_mean.head(10)
```

[41]:

Date	Consumption	Wind	Solar	Wind+Solar
2006-01-01	1069.184000	NaN	NaN	NaN
2006-01-08	1381.300143	NaN	NaN	NaN
2006-01-15	1486.730286	NaN	NaN	NaN
2006-01-22	1490.031143	NaN	NaN	NaN
2006-01-29	1514.176857	NaN	NaN	NaN
2006-02-05	1501.403286	NaN	NaN	NaN
2006-02-12	1498.217143	NaN	NaN	NaN
2006-02-19	1446.507429	NaN	NaN	NaN
2006-02-26	1447.651429	NaN	NaN	NaN
2006-03-05	1439.727857	NaN	NaN	NaN

[42]:
```
start, end = '2016-01', '2016-06'
```

[43]:
```
fig, ax = plt.subplots()

ax.plot(df.loc[start:end, 'Solar'],
marker='.', linestyle='-', linewidth=0.5, label='Daily')
ax.plot(power_weekly_mean.loc[start:end, 'Solar'],
marker='o', markersize=8, linestyle='-', label='Weekly Mean Resample')
ax.set_ylabel('Solar Production in (GWh)')
ax.legend()
plt.show()
```



3.1.9 Inference:

- Helps in identifying trends across weeks.
- Shows that consumption has increases on an average over the weeks

4 Time Series analysis for Bitcoin dataset

[114]:
```
df = pd.read_csv(r"D:\study material\VIT_Data_Science\Winter_Sem\Exploratory␣
↪Data Analysis Lab\7_Feb\btc-eth-prices.csv")
df
```

[114]:
```
      Timestamp      Bitcoin   Ether
0    2017-04-02  1099.169125   48.55
1    2017-04-03  1141.813000   44.13
2    2017-04-04  1141.600363   44.43
3    2017-04-05  1133.079314   44.90
4    2017-04-06  1196.307937   43.23
..          …            …      …
360  2018-03-28  7960.380000  445.93
361  2018-03-29  7172.280000  383.90
362  2018-03-30  6882.531667  393.82
363  2018-03-31  6935.480000  394.07
364  2018-04-01  6794.105000  378.85

[365 rows x 3 columns]
```

[115]:
```
df.shape
```

[115]: (365, 3)

[116]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Timestamp  365 non-null    object
 1   Bitcoin    365 non-null    float64
 2   Ether      362 non-null    float64
dtypes: float64(2), object(1)
memory usage: 8.7+ KB
```

[117]:
```
# Convert Date column to datetime format
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Set Date as index
df.set_index('Timestamp', inplace=True)
```

[118]:
```
# Extract year, month, and weekday name
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Weekday Name'] = df.index.day_name()
```

[119]:
```
df.loc["2017-04-08"]
```

```
[119]: Bitcoin         1181.149838
Ether                 44.37
Year                   2017
Month                     4
Weekday Name       Saturday
Name: 2017-04-08 00:00:00, dtype: object
```

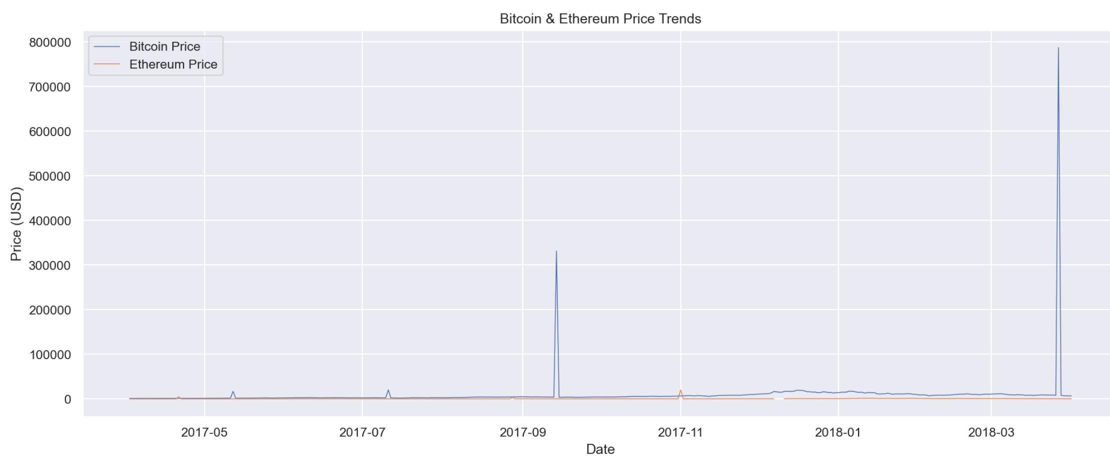
[120]:
```
df.loc["2016-04-05":"2019-03-05"]
```

[120]:

	Bitcoin	Ether	Year	Month	Weekday Name
Timestamp					
2017-04-02	1099.169125	48.55	2017	4	Sunday
2017-04-03	1141.813000	44.13	2017	4	Monday
2017-04-04	1141.600363	44.43	2017	4	Tuesday
2017-04-05	1133.079314	44.90	2017	4	Wednesday
2017-04-06	1196.307937	43.23	2017	4	Thursday
…	…	…	…	…	…
2018-03-28	7960.380000	445.93	2018	3	Wednesday

```
2018-03-29  7172.280000  383.90  2018      3     Thursday
2018-03-30  6882.531667  393.82  2018      3       Friday
2018-03-31  6935.480000  394.07  2018      3     Saturday
2018-04-01  6794.105000  378.85  2018      4       Sunday

[365 rows x 5 columns]
```

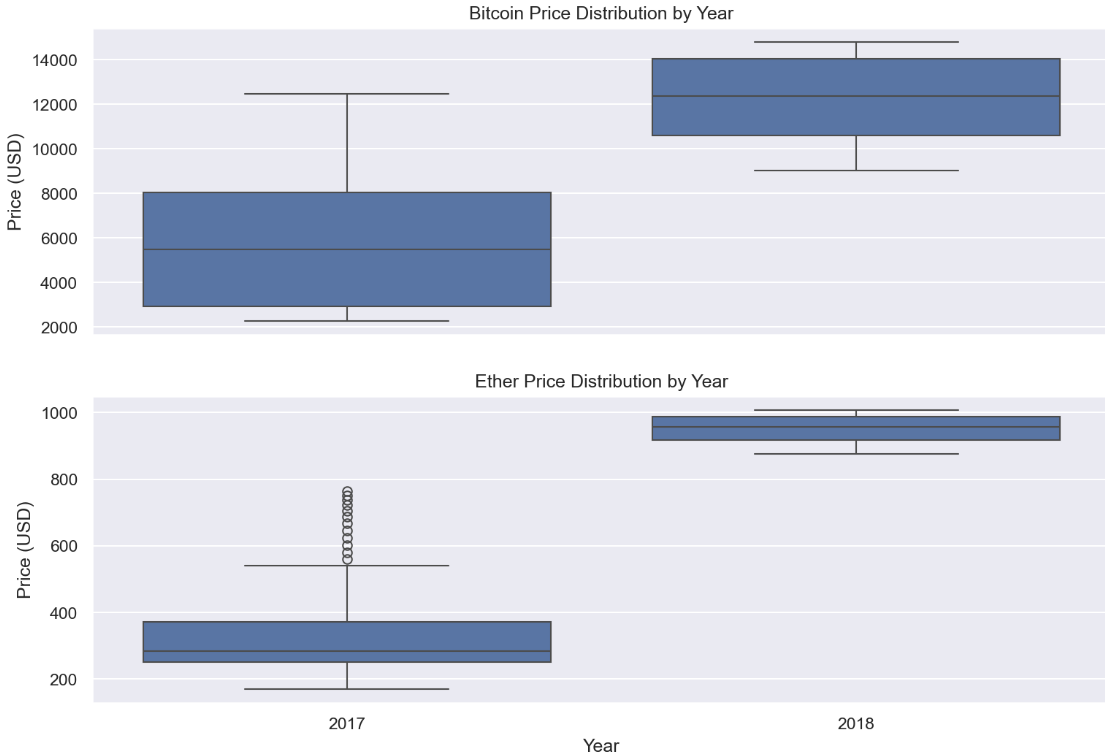
[121]:
```python
# Plot Bitcoin and Ethereum prices over time
plt.figure(figsize=(16, 6))
plt.plot(df.index, df['Bitcoin'], label="Bitcoin Price", linewidth=0.7)
plt.plot(df.index, df['Ether'], label="Ethereum Price", linewidth=0.7)
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.title("Bitcoin & Ethereum Price Trends")
plt.legend()
plt.show()
```



4.0.1 Inference:

- Both show rising trends with high volatility.
- Bitcoin prices fluctuate more than Ethereum.

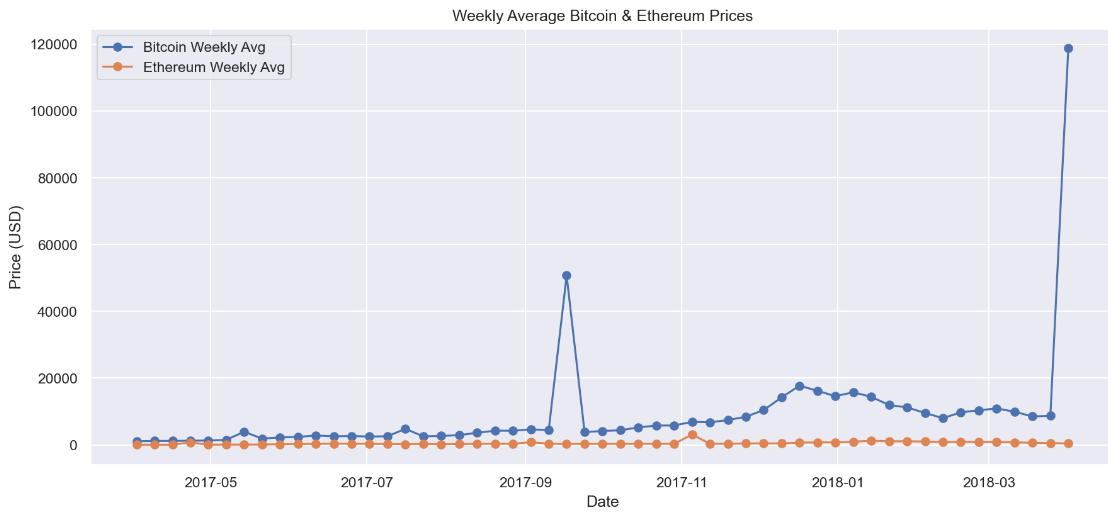
[134]:
```python
# Boxplot to see monthly variation in prices
fig, axes = plt.subplots(2, 1, figsize=(12, 8), sharex=True)
for name, ax in zip(['Bitcoin', 'Ether'], axes):
    sns.boxplot(data=df, x='Year', y=name, ax=ax)
    ax.set_ylabel('Price (USD)')
    ax.set_title(f"{name} Price Distribution by Year")
plt.show()
```



4.0.2 Inference:

- Shows how price increases significantly over time

[127]:
```
# Resample data to weekly average
weekly_prices = df[['Bitcoin', 'Ether']].resample('W').mean()

# Plot weekly averages
plt.figure(figsize=(14, 6))
plt.plot(weekly_prices.index, weekly_prices['Bitcoin'], marker='o',␣
↪linestyle='-', label='Bitcoin Weekly Avg')
plt.plot(weekly_prices.index, weekly_prices['Ether'], marker='o',␣
↪linestyle='-', label='Ethereum Weekly Avg')
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.title("Weekly Average Bitcoin & Ethereum Prices")
plt.legend()
plt.show()
```



4.0.3 Inference:

- Weekly moving average for bitcoin and ethereum
- Helps smooth short term price flactuations

[136]:
```python
df['Bitcoin'].isna().sum()
```

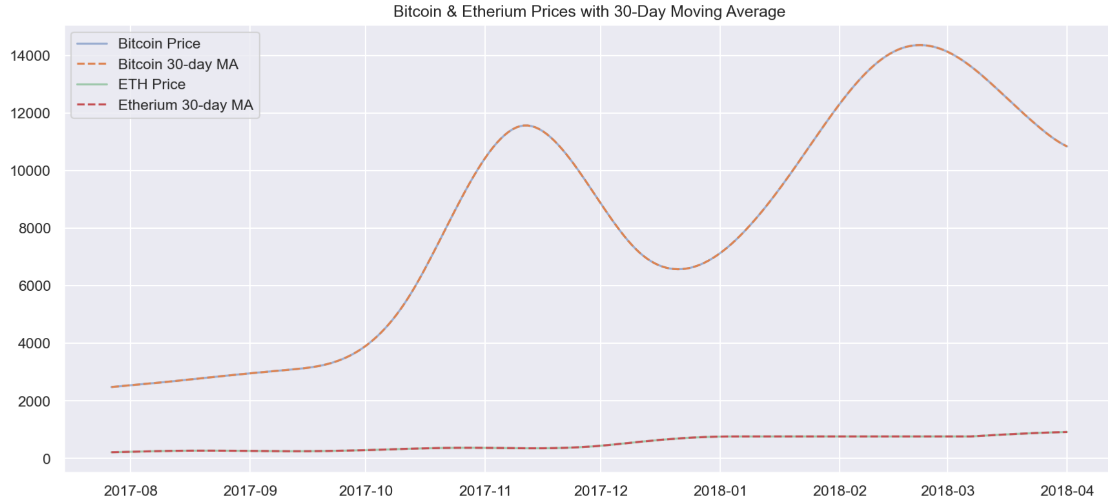
[136]: 87

[137]:
```python
df['Ether'].isna().sum()
```

[137]: 177

[138]:
```python
df['Bitcoin'].fillna(method='ffill', inplace=True)
df['Ether'].fillna(method='ffill', inplace=True)
```

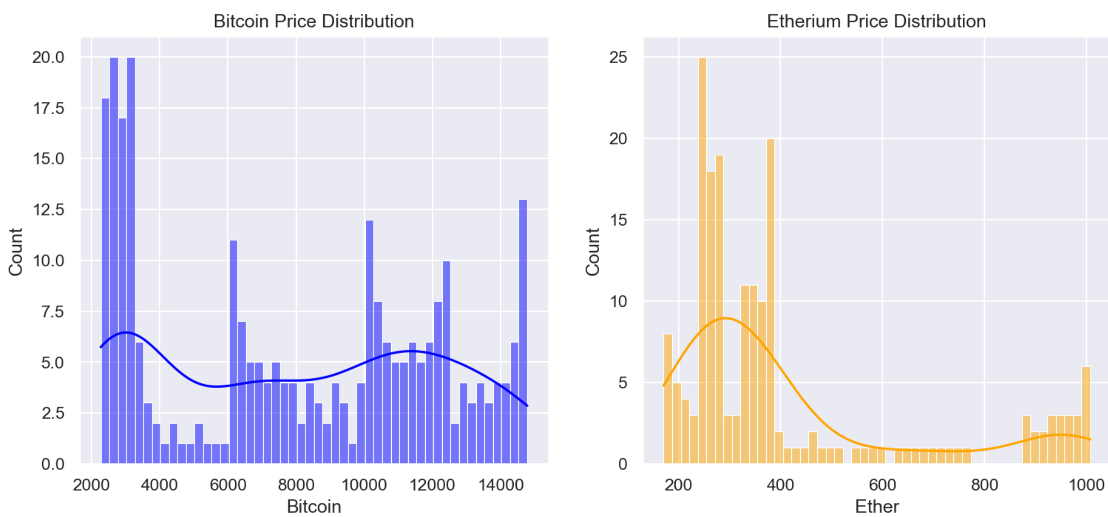
[139]:
```python
# Moving average
df['Bitcoin'] = df['Bitcoin'].rolling(window=30).mean()
df['Ether'] = df['Ether'].rolling(window=30).mean()

plt.figure(figsize=(14, 6))
plt.plot(df.index, df['Bitcoin'], alpha=0.5, label='Bitcoin Price')
plt.plot(df.index, df['Bitcoin'], label='Bitcoin 30-day MA', linestyle='--')
plt.plot(df.index, df['Ether'], alpha=0.5, label='ETH Price')
plt.plot(df.index, df['Ether'], label='Etherium 30-day MA', linestyle='--')
plt.legend()
plt.title('Bitcoin & Etherium Prices with 30-Day Moving Average')
plt.show()
```



4.0.4 Inference:

- 30 days moving average for Bitcoin and Ethereum
- Helps identify long-term trend
- Bitcoin price has increased significantly than Ethereum
- Bitcoin was more volatile than Ethereum

[131]:
```
# Histogram
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
sns.histplot(df['Bitcoin'], bins=50, kde=True, ax=ax[0], color='blue')
sns.histplot(df['Ether'], bins=50, kde=True, ax=ax[1], color='orange')
ax[0].set_title('Bitcoin Price Distribution')
ax[1].set_title('Etherium Price Distribution')
plt.show()
```



4.0.5 `Inference:`

- Prices are right-skewed, indicating some extreme high values.