In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from matplotlib.pyplot import axes, title
import numpy as np
```

In [5]:
```
df0 = pd.read_csv("data_tunis.csv")
```

In [7]:
```
df0.head()
```

Out[7]:

	date	Tmin	Tmax	vent	pluie
0	01/01/2010	12.0	19.0	50.0	0.0
1	02/01/2010	12.0	17.0	35.0	2.7
2	03/01/2010	11.0	18.0	15.0	0.5
3	04/01/2010	9.0	19.0	13.0	0.0
4	05/01/2010	8.0	23.0	16.0	0.0

In [8]:

```
# Fill some null values
df0 = df0.bfill(axis=0)

# Remove null acquisations
df0 = df0.dropna()

# Replace min_temp and max_temp with avg_temp
df0['temp'] = (df0['Tmin'] + df0['Tmax']) / 2
df0 = df0.drop('Tmin',axis=1)
df0 = df0.drop('Tmax',axis=1)

df0
```

Out[8]:

	date	vent	pluie	temp
0	01/01/2010	50.0	0.0	15.5
1	02/01/2010	35.0	2.7	14.5
2	03/01/2010	15.0	0.5	14.5
3	04/01/2010	13.0	0.0	14.0
4	05/01/2010	16.0	0.0	15.5
...	...	...	...	...
2915	25/12/2017	15.0	0.2	8.0
2916	26/12/2017	29.0	0.0	11.5
2917	27/12/2017	29.0	0.0	6.0
2918	28/12/2017	28.0	3.5	11.0
2919	29/12/2017	39.0	6.3	12.5

2920 rows × 4 columns

In [18]:

```
# Create empty dataframes for each month
df_jan = pd.DataFrame(columns=df0.columns)
df_feb = pd.DataFrame(columns=df0.columns)
df_mar = pd.DataFrame(columns=df0.columns)
df_apr = pd.DataFrame(columns=df0.columns)
df_may = pd.DataFrame(columns=df0.columns)
df_jun = pd.DataFrame(columns=df0.columns)
df_jul = pd.DataFrame(columns=df0.columns)
df_aug = pd.DataFrame(columns=df0.columns)
df_sep = pd.DataFrame(columns=df0.columns)
df_oct = pd.DataFrame(columns=df0.columns)
df_nov = pd.DataFrame(columns=df0.columns)
df_dec = pd.DataFrame(columns=df0.columns)

# Fill the dataframes of each month
for i in range(len(df0)):
    if df0.loc[i]['date'][3:5] == '01':
        df_jan = df_jan.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '02':
        df_feb = df_feb.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '03':
        df_mar = df_mar.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '04':
        df_apr = df_apr.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '05':
        df_may = df_may.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '06':
        df_jun = df_jun.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '07':
        df_jul = df_jul.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '08':
        df_aug = df_aug.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '09':
        df_sep = df_sep.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '10':
        df_oct = df_oct.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '11':
        df_nov = df_nov.append(df0.loc[i])
    if df0.loc[i]['date'][3:5] == '12':
        df_dec = df_dec.append(df0.loc[i])
```

```
future version. Use pandas.concat instead.
  df_jan = df_jan.append(df0.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\4269858596.py:18: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  df_jan = df_jan.append(df0.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\4269858596.py:18: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  df_jan = df_jan.append(df0.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\4269858596.py:18: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  df_jan = df_jan.append(df0.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\4269858596.py:18: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  df_jan = df_jan.append(df0.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\4269858596.py:18: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a
```

In [24]:

```
# Choose a month to work with; for example: January
df_curr = df_jan

# Remove the columns 'date' and 'index'
df_curr = df_curr.reset_index().drop(['date','index'],axis=1)

# Get xdata, ydata and zdata to make 3D plot later on
xdata = df_curr['vent']
ydata = df_curr['pluie']
zdata = df_curr['temp']

# # Fill the dataframe of reference points
for i in range(len(df_curr)):
    if df_curr.loc[i, 'pluie'] > MAX_PLUIE or df_curr.loc[i, 'vent'] > MAX_VENT
        df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])

print("The number of anomalies reference is: " + str(len(df_anomaly_reference))
```

```
The number of anomalies reference is: 6

C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\3524759750.py:15: FutureWarnin
g: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df_anomaly_reference = df_anomaly_reference.append(df_curr.loc[i])
```

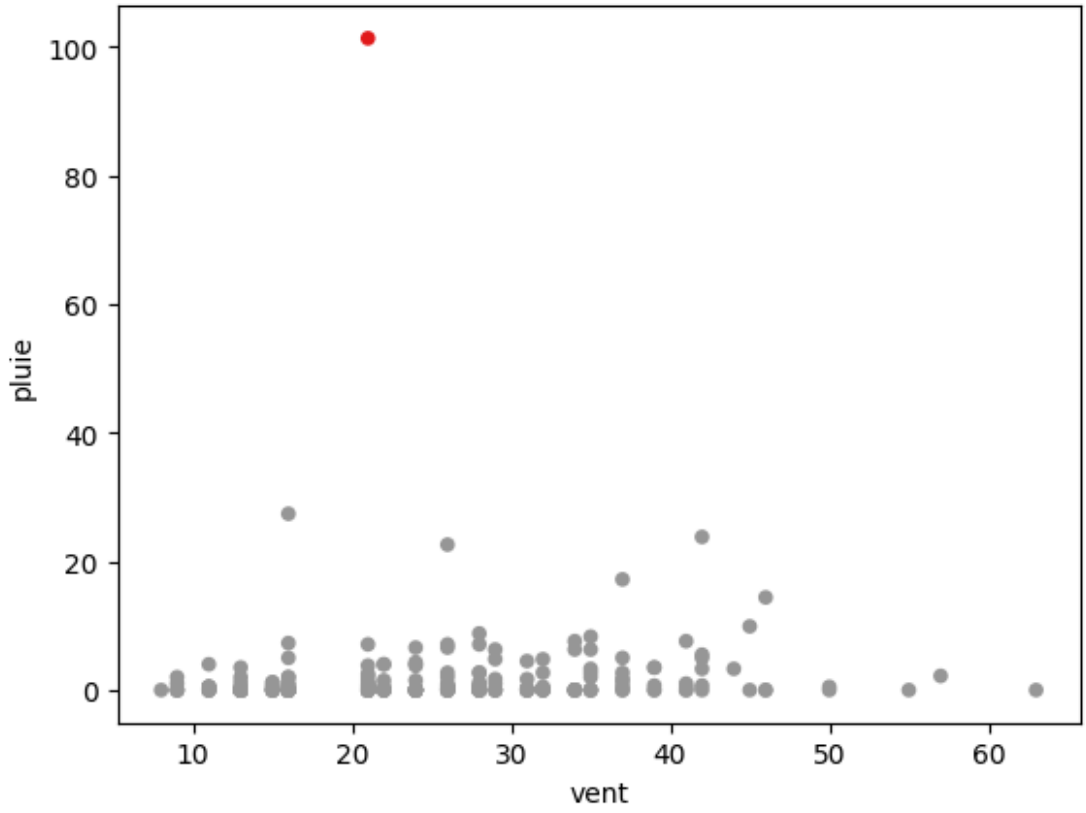
In [ ]:

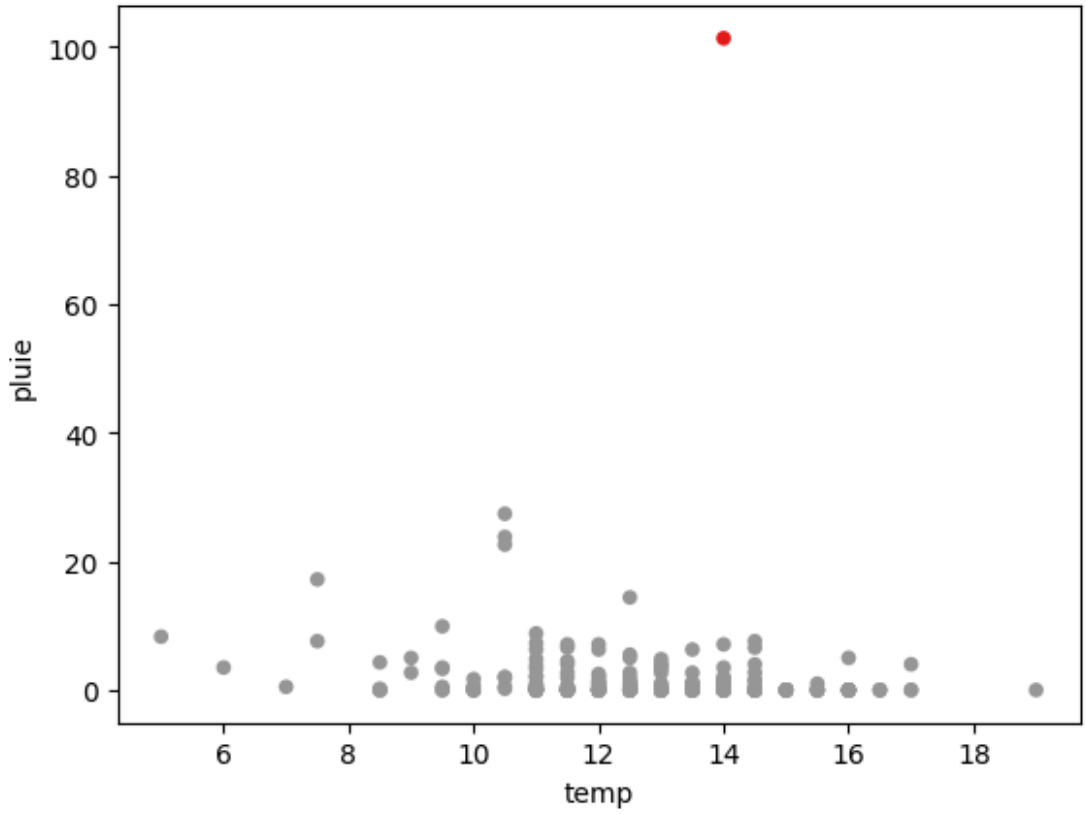
```

```

In [14]:
```
 #1st algorithm: DBSCAN
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
outlier_detection = DBSCAN(eps = .2, metric="euclidean", min_samples = 5, n_job
num2 = scaler.fit_transform(df_curr)
num2 = pd.DataFrame(num2, columns = df_curr.columns)
clusters = outlier_detection.fit_predict(num2)

# 2D Plot: pluie = f(vent)
from matplotlib import cm
cmap = cm.get_cmap('Set1')
df_curr.plot.scatter(x='vent',y='pluie', c=clusters, cmap=cmap, colorbar = Fals

# 2D Plot: pluie = f(temp)
from matplotlib import cm
cmap = cm.get_cmap('Set1')
df_curr.plot.scatter(x='temp',y='pluie', c=clusters, cmap=cmap, colorbar = Fals

# PS: This will be the only time that we show 2D plots
```

```
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\2706376578.py:12: MatplotlibDe
precationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will
be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.g
et_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap('Set1')
C:\Users\SRISHTI\AppData\Local\Temp\ipykernel_23676\2706376578.py:17: MatplotlibDe
precationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will
be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.g
et_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap('Set1')
```

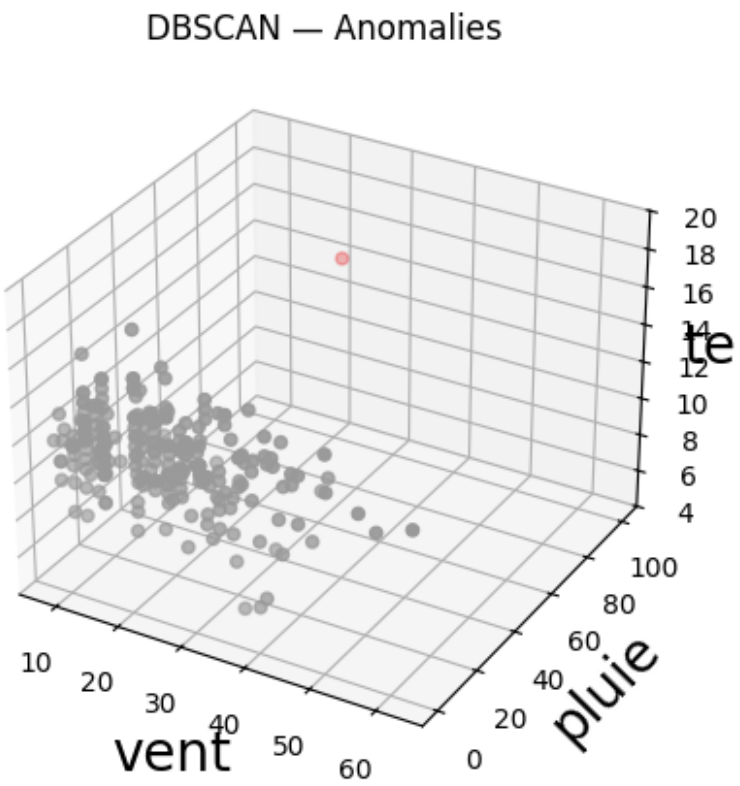
Out[14]:
```
<Axes: xlabel='temp', ylabel='pluie'>
```


pluie
vent
0
20
40
60
80
100
10
20
30
40
50
60


pluie
temp
0
20
40
60
80
100
6
8
10
12
14
16
18

In [19]:
```
# 3D Plot: pluie = f(temp, vent)
ax = axes(projection='3d')
ax.scatter3D(xdata, ydata, zdata, c=clusters, cmap=cmap)
ax.set_xlabel('vent',fontsize=20)
ax.set_ylabel('pluie',fontsize=20)
ax.set_zlabel('temp',fontsize=20)
title('DBSCAN — Anomalies')

true_predicted = 0
mistakes = 0

# Get True Negative and False Positive
for i in range(len(clusters)):
    if clusters[i] == -1 and i in df_anomaly_reference.index:
        true_predicted += 1
    if clusters[i] == -1 and i not in df_anomaly_reference.index:
        mistakes += 1

print("DBSCAN  | True Negative = " + str(true_predicted))
print("DBSCAN  | False Positive = " + str(mistakes))
```

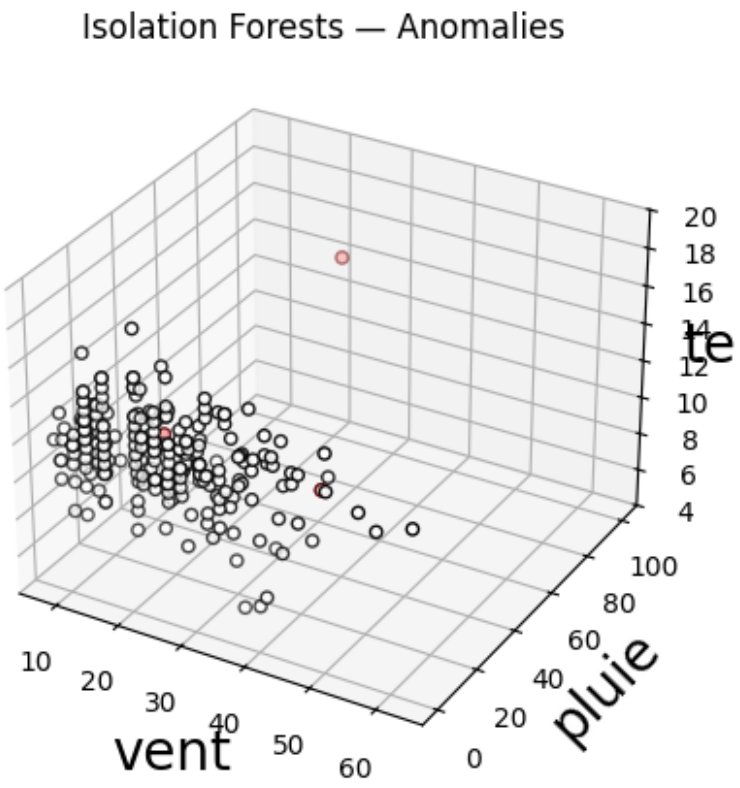
```
DBSCAN  | True Negative = 0
DBSCAN  | False Positive = 1
```



In [20]:

```python
# 2nd algorithm: Isolation Forests
from sklearn.ensemble import IsolationForest
rs=np.random.RandomState(0)
clf = IsolationForest(max_samples=100,random_state=rs, contamination=.01)
clf.fit(df_curr)
if_scores = clf.decision_function(df_curr)
if_anomalies=clf.predict(df_curr)
if_anomalies=pd.Series(if_anomalies).replace([-1,1],[1,0])
if_anomalies=df_curr[if_anomalies==1]

# 3D Plot
ax = plt.axes(projection='3d')
ax.scatter3D(xdata, ydata, zdata, c='white',s=20,edgecolor='k')
ax.scatter3D(if_anomalies['vent'], if_anomalies['pluie'], if_anomalies['temp'],
ax.set_xlabel('vent',fontsize=20)
ax.set_ylabel('pluie',fontsize=20)
ax.set_zlabel('temp',fontsize=20)
plt.title('Isolation Forests – Anomalies')

true_predicted = 0
mistakes = 0

# Get True Negative and False Positive
for i in range(len(clusters)):
    if i in if_anomalies.index and i in df_anomaly_reference.index:
        true_predicted += 1
    if i in if_anomalies.index and i not in df_anomaly_reference.index:
        mistakes += 1

print("Isolation Forests | True Negative = " + str(true_predicted))
print("Isolation Forests | False Positive = " + str(mistakes))
```

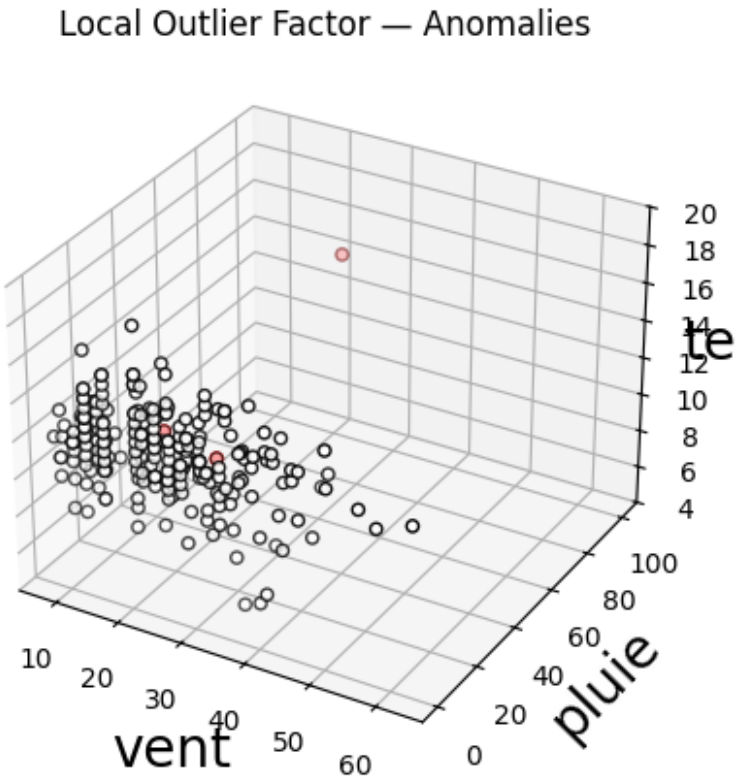
```
Isolation Forests | True Negative = 0
Isolation Forests | False Positive = 3
```


Isolation Forests — Anomalies
vent
pluie
te
10
20
30
40
50
60
0
20
40
60
80
100
4
6
8
10
12
14
16
18
20

In [21]:

```
# 3rd algorithm: Local Outlier Factor
from sklearn.neighbors import LocalOutlierFactor
clf = LocalOutlierFactor(n_neighbors=30, contamination=.01)
y_pred = clf.fit_predict(df_curr)
LOF_Scores = clf.negative_outlier_factor_
LOF_pred=pd.Series(y_pred).replace([-1,1],[1,0])
LOF_anomalies=df_curr[LOF_pred==1]

# 3D Plot
ax = plt.axes(projection='3d')
ax.scatter3D(xdata, ydata, zdata, c='white',s=20,edgecolor='k')
ax.scatter3D(LOF_anomalies['vent'], LOF_anomalies['pluie'], LOF_anomalies['temp
ax.set_xlabel('vent',fontsize=20)
ax.set_ylabel('pluie',fontsize=20)
ax.set_zlabel('temp',fontsize=20)
plt.title('Local Outlier Factor – Anomalies')

true_predicted = 0
mistakes = 0

# Get True Negative and False Positive
for i in range(len(clusters)):
    if i in LOF_anomalies.index and i in df_anomaly_reference.index:
        true_predicted += 1
    if i in LOF_anomalies.index and i not in df_anomaly_reference.index:
        mistakes += 1

print("Local Outlier Factor | True Negative = " + str(true_predicted))
print("Local Outlier Factor | False Positive = " + str(mistakes))
```

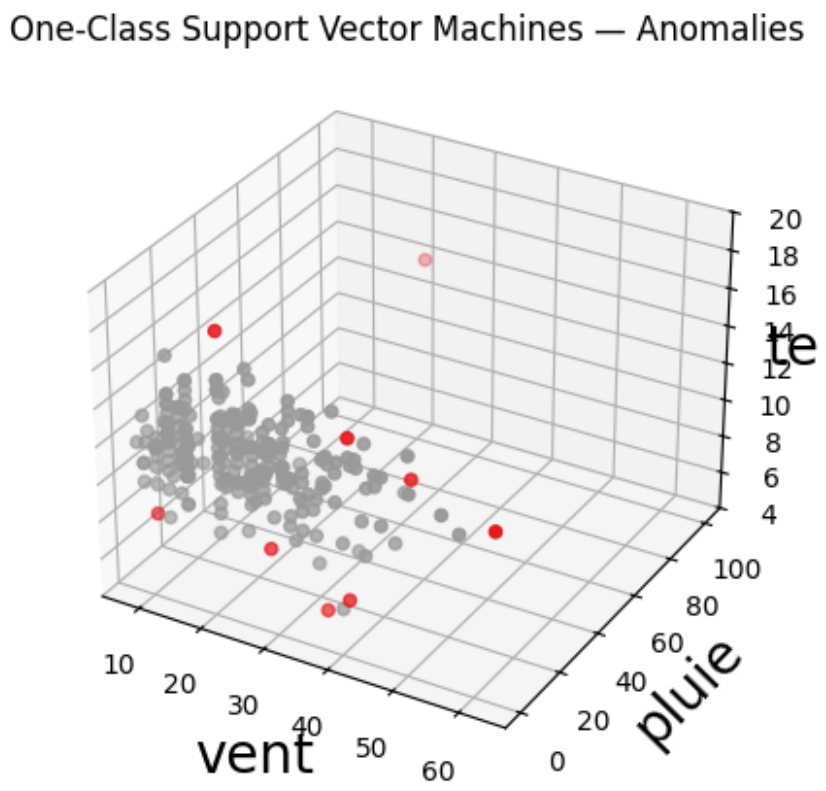
```
Local Outlier Factor | True Negative = 0
Local Outlier Factor | False Positive = 3
```



In [22]:

```
# 5th algorithm: One-Class Support Vector Machines
from sklearn import svm
clf=svm.OneClassSVM(nu=.02,kernel='rbf',gamma=.005)
clf.fit(df_curr)
y_pred=clf.predict(df_curr)

# 3D Plot
ax = plt.axes(projection='3d')
ax.scatter3D(xdata, ydata, zdata, c=y_pred, cmap=cmap)
ax.set_xlabel('vent',fontsize=20)
ax.set_ylabel('pluie',fontsize=20)
ax.set_zlabel('temp',fontsize=20)
plt.title('One-Class Support Vector Machines — Anomalies')

true_predicted = 0
mistakes = 0

# Get True Negative and False Positive
for i in range(len(clusters)):
    if y_pred[i] == -1 and i in df_anomaly_reference.index:
        true_predicted += 1
    if y_pred[i] == -1 and i not in df_anomaly_reference.index:
        mistakes += 1

print("One-Class Support Vector Machines | True Negative = " + str(true_predict
print("One-Class Support Vector Machines | False Positive = " + str(mistakes))
```

```
One-Class Support Vector Machines | True Negative = 0
One-Class Support Vector Machines | False Positive = 9
```



In [ ]:
```
1
```