Algerian Forest Fire Dataset - Temperature Prediction

- Data Collection
- Exploratory data analysis
- Data Cleaning
- Linear Regression Model Traning
- Ridge Regression Model Traning
- Lasso Regression Model Traning
- Elastincet Regression Model Traning

Importing the Libraries

In [63]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

Data Reading and Cleaning

In [64]:
```
df = pd.read_csv(r"C:\Users\hrush\Downloads\Algerian_forest_fires_dataset_UPDATE (1).csv",header=1)
df.head()
```

Out[64]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire

Drop an row

In [65]:
```
df.drop([122,123],inplace=True)
df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)
```

In [66]:
```
df.loc[:122, 'region'] = 'bejaia'
df.loc[122:, 'region'] = 'Sidi-Bel Abbes'
```

Stripping the names of the columns

In [67]:
```
df.columns = [i.strip() for i in df.columns]
df.columns
```

Out[67]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'region'],
      dtype='object')
```

Dropping the Classes Feature

In [68]:
```
df.drop('Classes',axis=1,inplace=True)
```

In [69]:

```
df.head()
```

Out[69]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	bejaia
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	bejaia
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	bejaia
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	bejaia
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	bejaia

Replacing the day,month,year feature with date feature

In [70]:

```
df['date']=pd.to_datetime(df[['day','month','year']])
df.drop(['day','month','year'],axis=1,inplace=True)
```

In [71]:

```
df.head()
```

Out[71]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	date
0	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	bejaia	2012-06-01
1	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	bejaia	2012-06-02
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	bejaia	2012-06-03
3	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	bejaia	2012-06-04
4	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	bejaia	2012-06-05

Checking the datatypes of feature

In [72]:

```
df.dtypes
```

Out[72]:

```
Temperature             object
RH                      object
Ws                      object
Rain                    object
FFMC                    object
DMC                     object
DC                      object
ISI                     object
BUI                     object
FWI                     object
region                  object
date            datetime64[ns]
dtype: object
```

Checking the datatypes of features

In [73]:

```
df['Temperature']=df['Temperature'].astype(int)
df['RH']=df['RH'].astype(int)
df['Ws']=df['Ws'].astype(int)
df['Rain']=df['Rain'].astype(float)
df['FFMC']=df['FFMC'].astype(float)
df['DMC']=df['DMC'].astype(float)
df['ISI']=df['ISI'].astype(float)
df['BUI']=df['BUI'].astype(float)
```

In [74]:

```
df.dtypes
```

Out[74]:

```
Temperature              int32
RH                       int32
Ws                       int32
Rain                   float64
FFMC                   float64
DMC                    float64
```

```
DC                        object
ISI                      float64
BUI                      float64
FWI                       object
region                    object
date              datetime64[ns]
dtype: object
```

Applying Label encoding in DC,FWI,region features

In [75]:

```
from sklearn.preprocessing import LabelEncoder
LabelEncoder=LabelEncoder()
```

In [76]:

```
df['DC']=LabelEncoder.fit_transform(df['DC'])
df['FWI']=LabelEncoder.fit_transform(df['FWI'])
df['region']=LabelEncoder.fit_transform(df['region'])
```

In [77]:

```
df.dtypes
```

Out[77]:

```
Temperature                int32
RH                         int32
Ws                         int32
Rain                     float64
FFMC                     float64
DMC                      float64
DC                         int32
ISI                      float64
BUI                      float64
FWI                        int32
region                     int32
date              datetime64[ns]
dtype: object
```

In [78]:

```
df.head()
```

Out[78]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	date
0	29	57	18	0.0	65.7	3.4	150	1.3	3.4	5	1	2012-06-01
1	29	61	13	1.3	64.4	4.1	150	1.0	3.9	4	1	2012-06-02
2	26	82	22	13.1	47.1	2.5	146	0.3	2.7	1	1	2012-06-03
3	25	89	13	2.5	28.6	1.3	136	0.0	1.7	0	1	2012-06-04
4	27	77	16	0.0	64.8	3.0	18	1.2	3.9	5	1	2012-06-05

Checking the null values

In [79]:

```
df.isnull().sum()
```

Out[79]:

```
Temperature    0
RH             0
Ws             0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
region         0
date           0
dtype: int64
```

Observation

Zero null value in the dataset

Univariate Analysis

In [80]:
```python
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O']
```

In [81]:
```python
numeric_features
```

Out[81]:
```
['Temperature',
 'RH',
 'Ws',
 'Rain',
 'FFMC',
 'DMC',
 'DC',
 'ISI',
 'BUI',
 'FWI',
 'region',
 'date']
```

Features Information

*Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations

*Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42

*RH : Relative Humidity in %: 21 to 90

*Ws :Wind speed in km/h: 6 to 29

*Rain: total day in mm: 0 to 16.8 FWI Components

*Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5

*Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9

*Drought Code (DC) index from the FWI system: 7 to 220.4

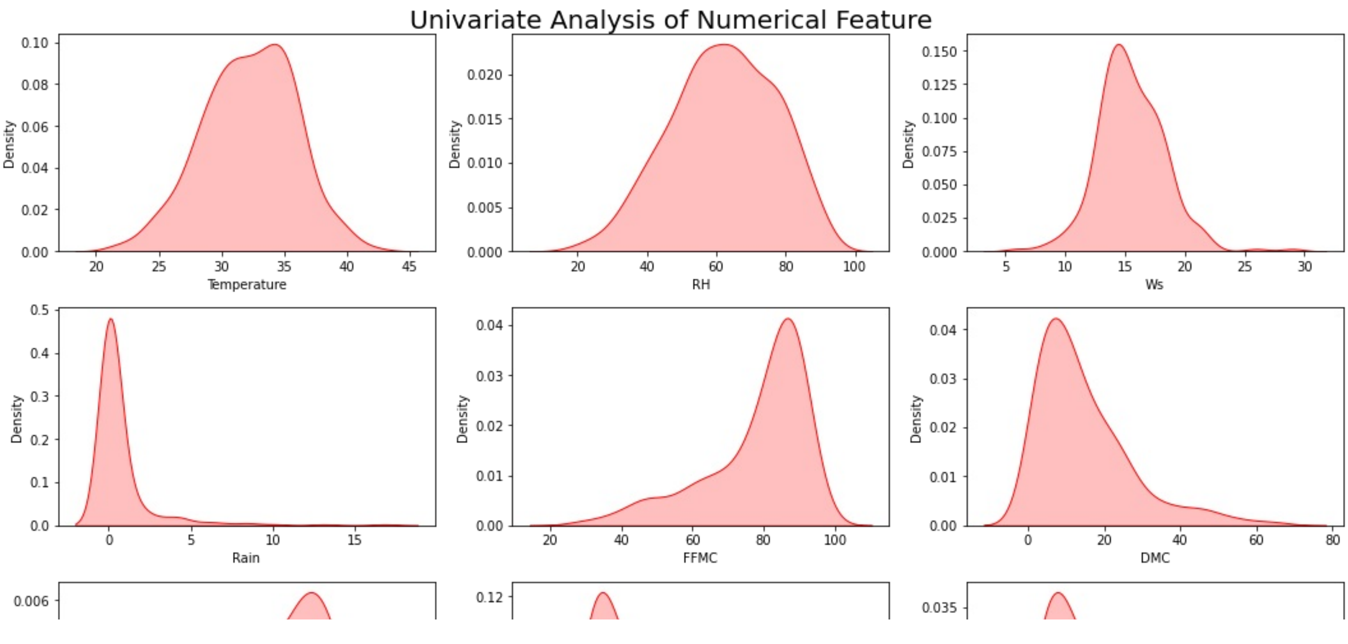
*Initial Spread Index (ISI) index from the FWI system: 0 to 18.5

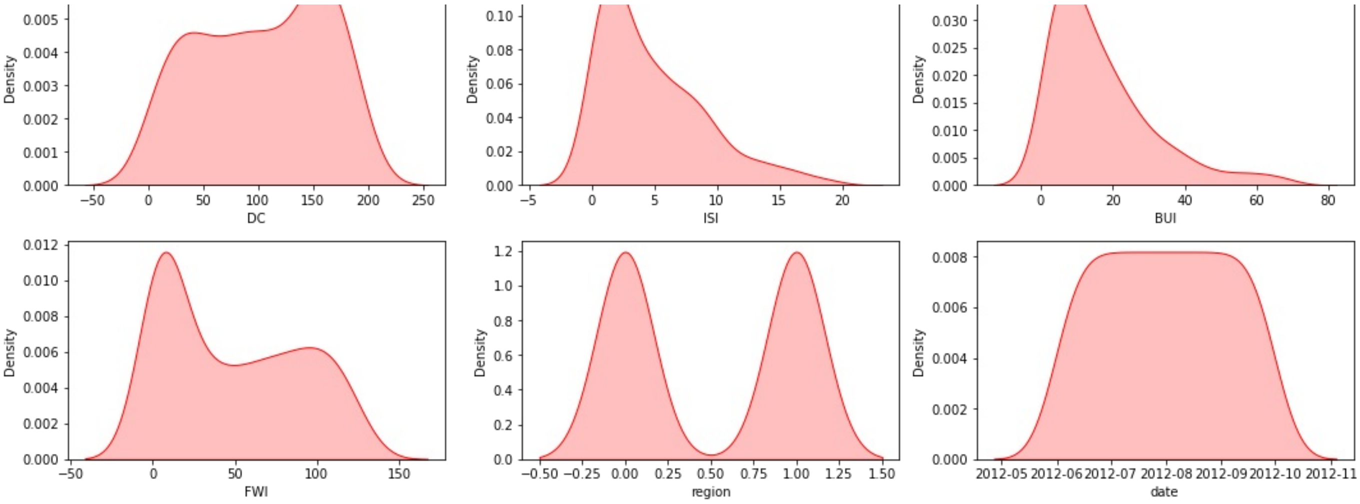
*Buildup Index (BUI) index from the FWI system: 1.1 to 68

*Fire Weather Index (FWI) Index: 0 to 31.1

In [82]:
```python
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Numerical Feature', fontsize=20, fontweight=20)

for i in range(0, len(numeric_features)):
    plt.subplot(5, 3, i+1)
    sns.kdeplot(x=df[numeric_features[i]],shade=True, color='r')
    plt.xlabel(numeric_features[i])
    plt.tight_layout()
```

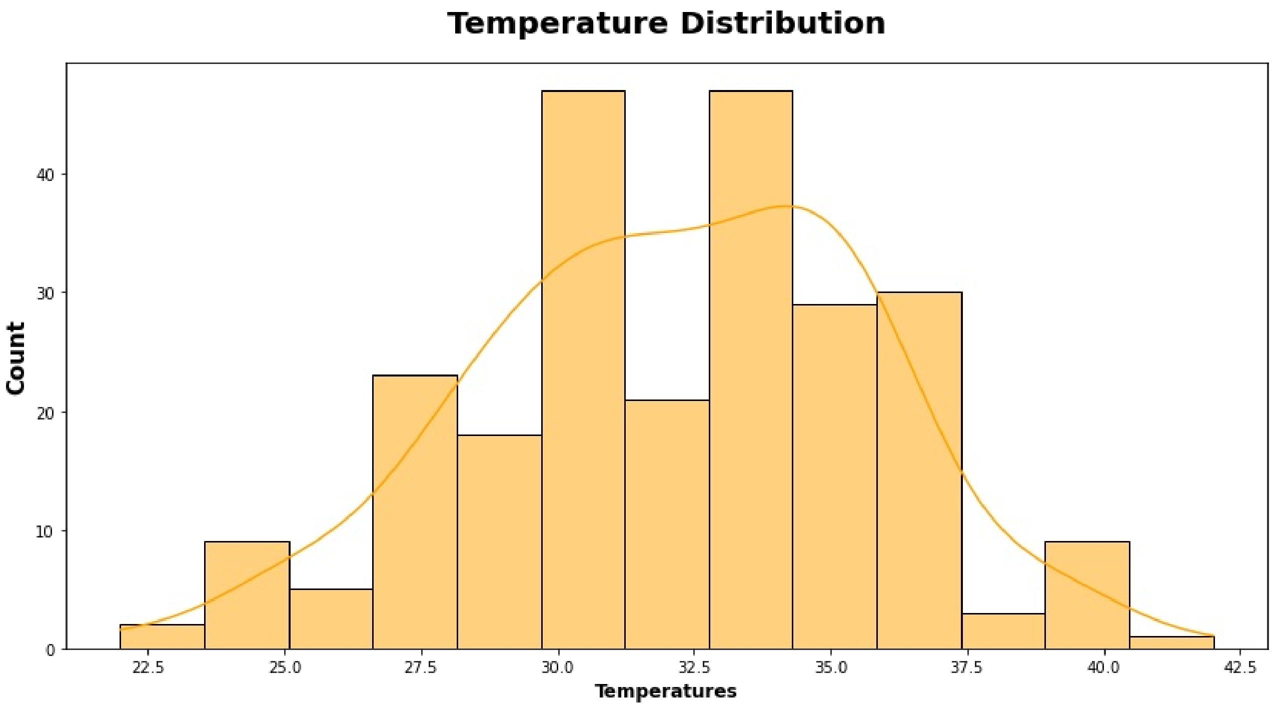




Visualization of Target Feature

In [83]:

```
plt.subplots(figsize=(14,7))
sns.histplot(x=df.Temperature, ec = "black", color='orange', kde=True)
plt.title("Temperature Distribution", weight="bold",fontsize=20, pad=20)
plt.ylabel("Count", weight="bold", fontsize=15)
plt.xlabel("Temperatures", weight="bold", fontsize=12)
plt.show()
```



Observation

Temperature occur most of the time in range 32.5 to 35.0

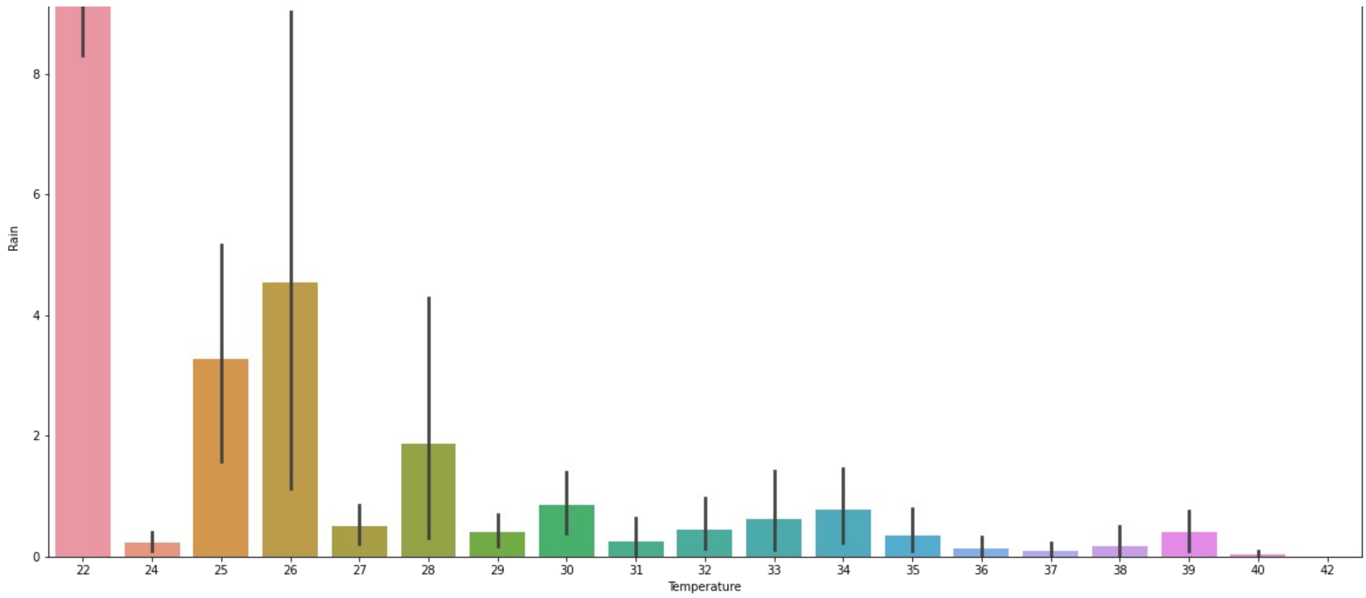
Temperature Vs Rain

In [84]:

```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

sns.barplot(x="Temperature",y="Rain",data=df)
```

Out[84]: <AxesSubplot:xlabel='Temperature', ylabel='Rain'>



Observation

When the temperature is around 22, most of the rain occur

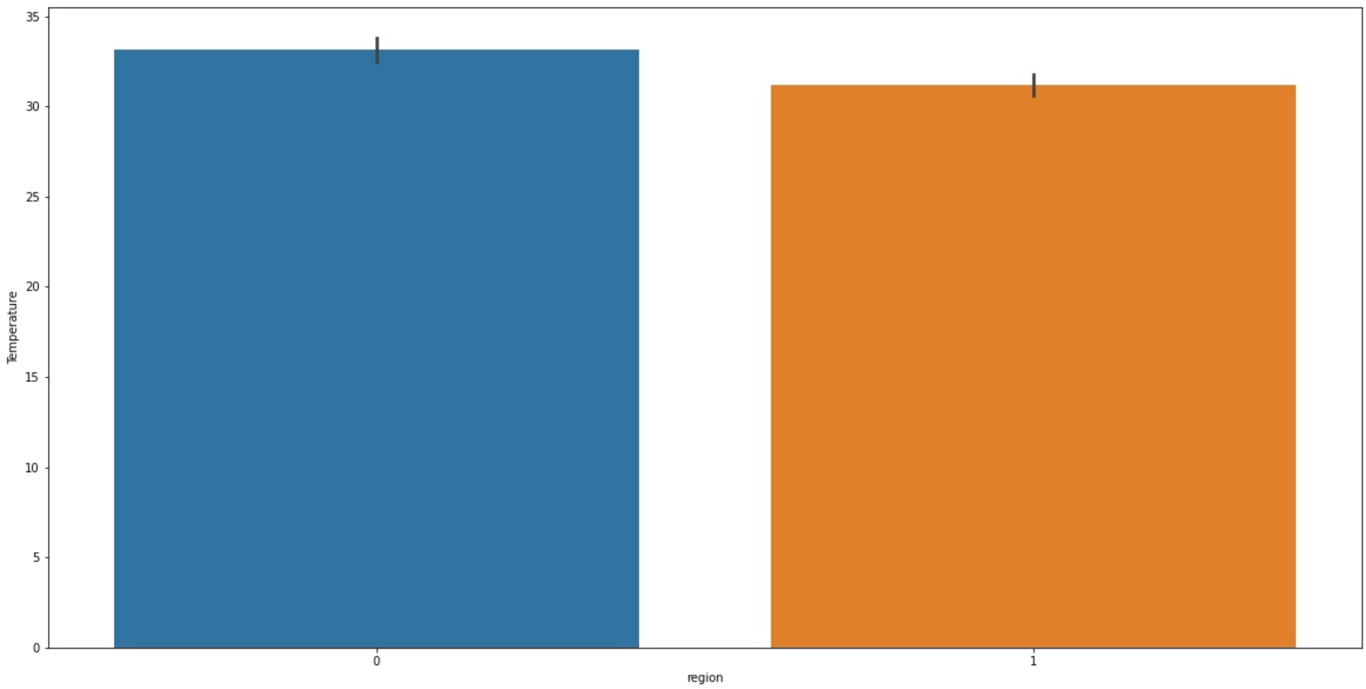
Which region has most temperature?

In [86]:

```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

sns.barplot(x="region",y="Temperature",data=df)
```

Out[86]: <AxesSubplot:xlabel='region', ylabel='Temperature'>



Observation

Region represented by 0 i.e. 'Sidi-Bel Abbes' has highest temperature

Correlation of the features

In [87]:

```
df.corr()
```

Out[87]:

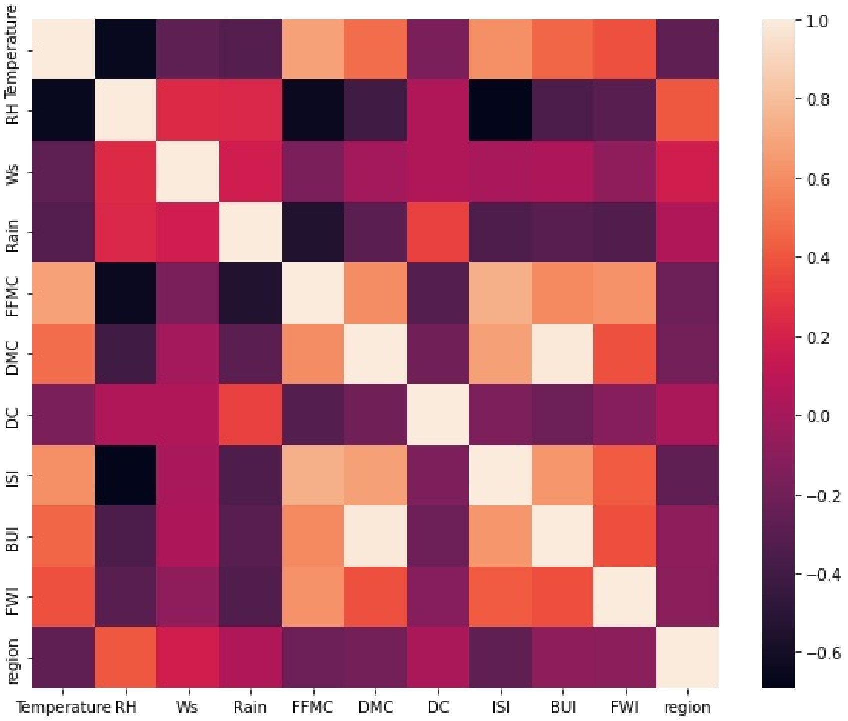
	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
Temperature	1.000000	-0.654443	-0.278132	-0.326786	0.677491	0.483105	-0.165840	0.607551	0.455504	0.380581	-0.273496
RH	-0.654443	1.000000	0.236084	0.222968	-0.645658	-0.405133	0.041651	-0.690637	-0.348587	-0.295093	0.406424
Ws	-0.278132	0.236084	1.000000	0.170169	-0.163255	-0.001246	0.040958	0.015248	0.029756	-0.081447	0.176829
Rain	-0.326786	0.222968	0.170169	1.000000	-0.544045	-0.288548	0.324748	-0.347105	-0.299171	-0.340412	0.041080
FFMC	0.677491	-0.645658	-0.163255	-0.544045	1.000000	0.602391	-0.319086	0.739730	0.589652	0.617445	-0.224680
DMC	0.483105	-0.405133	-0.001246	-0.288548	0.602391	1.000000	-0.200609	0.674499	0.982073	0.384628	-0.191094
DC	-0.165840	0.041651	0.040958	0.324748	-0.319086	-0.200609	1.000000	-0.152717	-0.226445	-0.118684	0.016293
ISI	0.607551	-0.690637	0.015248	-0.347105	0.739730	0.674499	-0.152717	1.000000	0.635891	0.412512	-0.268421
BUI	0.455504	-0.348587	0.029756	-0.299171	0.589652	0.982073	-0.226445	0.635891	1.000000	0.375234	-0.087370
FWI	0.380581	-0.295093	-0.081447	-0.340412	0.617445	0.384628	-0.118684	0.412512	0.375234	1.000000	-0.108099
region	-0.273496	0.406424	0.176829	0.041080	-0.224680	-0.191094	0.016293	-0.268421	-0.087370	-0.108099	1.000000

Multivariate analysis

In [88]:

```python
import seaborn as sns
plt.figure(figsize=(10,8))
sns.heatmap(df.corr())
```

Out[88]: <AxesSubplot:>



Observation

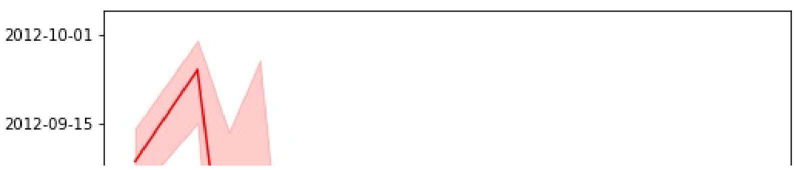
The target feature Temperature is highly positively correlated with FFMC,ISI

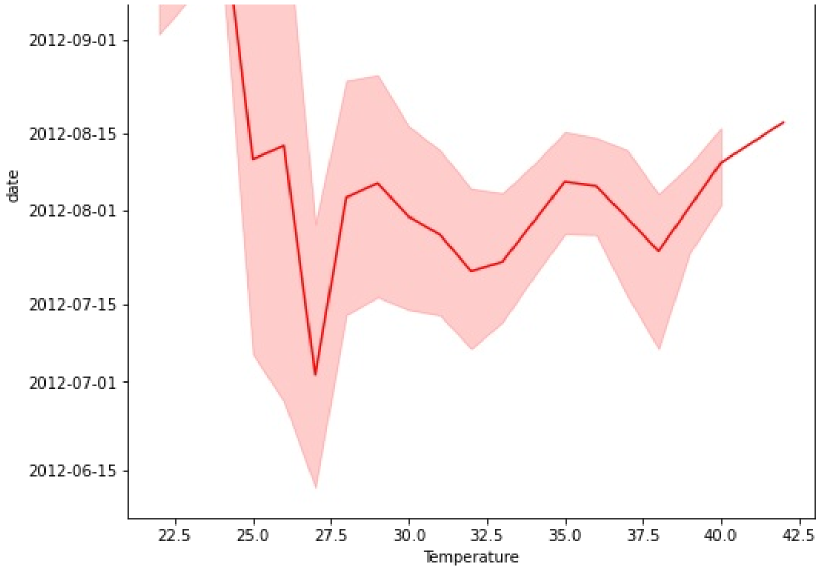
Temperature Vs date feature

In [89]:

```python
plt.figure(figsize=(8,8))
sns.lineplot(x='Temperature',y='date',data=df,color='r')
```

Out[89]: <AxesSubplot:xlabel='Temperature', ylabel='date'>



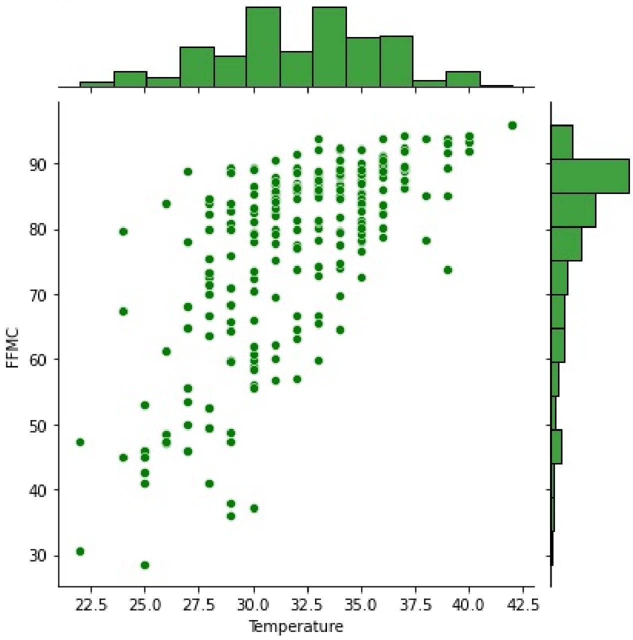


Temperature Vs FFMC

In [90]:
```
plt.figure(figsize=(10,10))
sns.jointplot(x='Temperature',y='FFMC',data=df,color='g')
```

Out[90]: <seaborn.axisgrid.JointGrid at 0x1c390143d90>

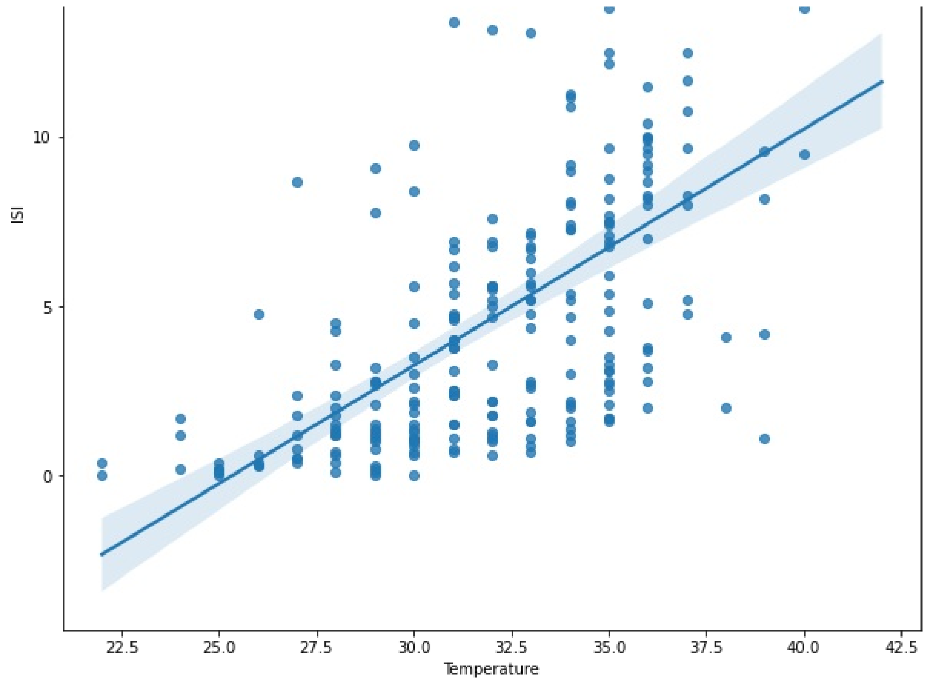
<Figure size 720x720 with 0 Axes>



Temperature Vs ISI

In [91]:
```
plt.figure(figsize=(10,10))
sns.regplot(x='Temperature',y='ISI',data=df)
```

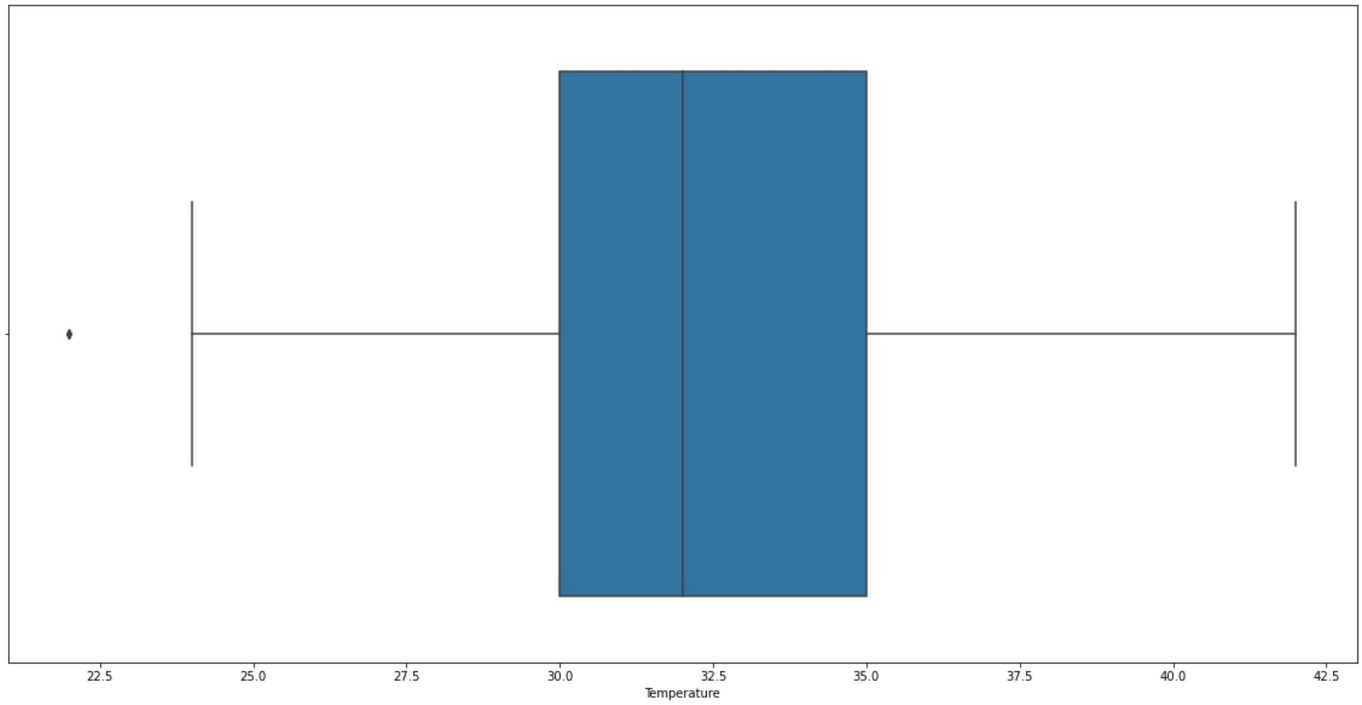
Out[91]: <AxesSubplot:xlabel='Temperature', ylabel='ISI'>



Checking the outliers of the target 'Temperature' feature

In [92]:
```
sns.boxplot(df['Temperature'])
```

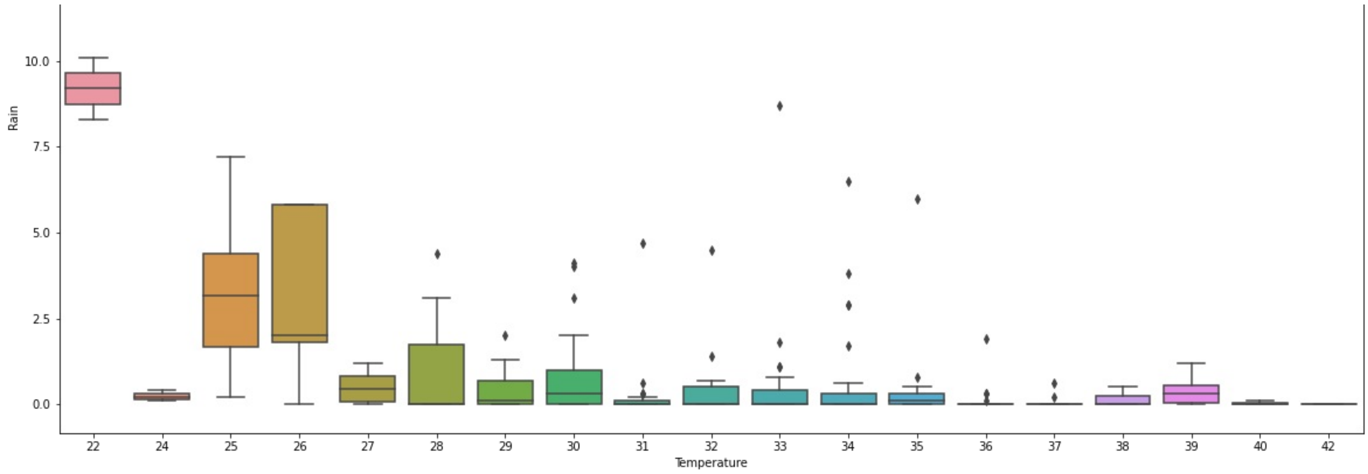
Out[92]: <AxesSubplot:xlabel='Temperature'>



Boxplot of Rain Vs Temperature

In [93]:
```
sns.boxplot(x ='Temperature', y ='Rain', data = df)
```

Out[93]: <AxesSubplot:xlabel='Temperature', ylabel='Rain'>



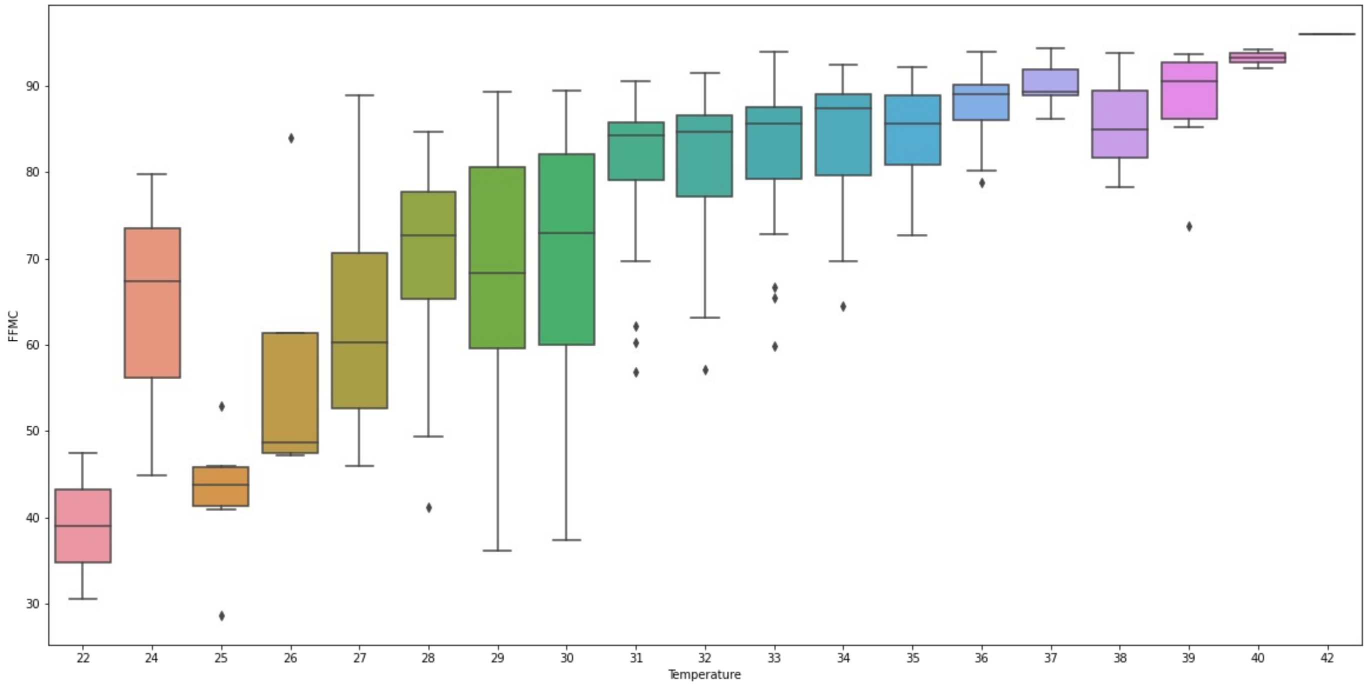
Boxplot of 'FFMC' Vs Temperature

In [94]:

```python
sns.boxplot(x ='Temperature', y ='FFMC', data = df)
```

Out[94]:

```
<AxesSubplot:xlabel='Temperature', ylabel='FFMC'>
```



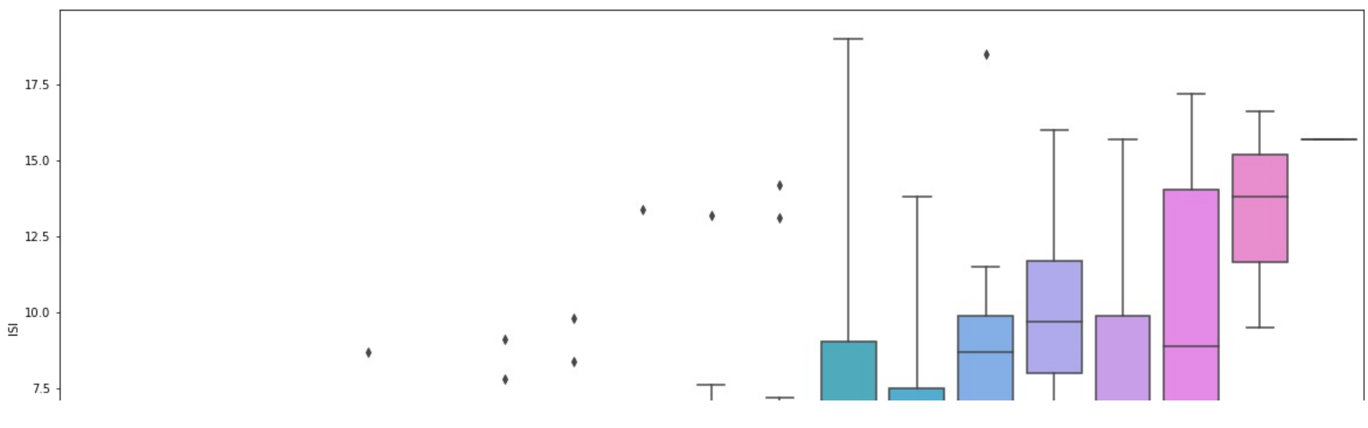
Boxplot of ISI Vs Temperature

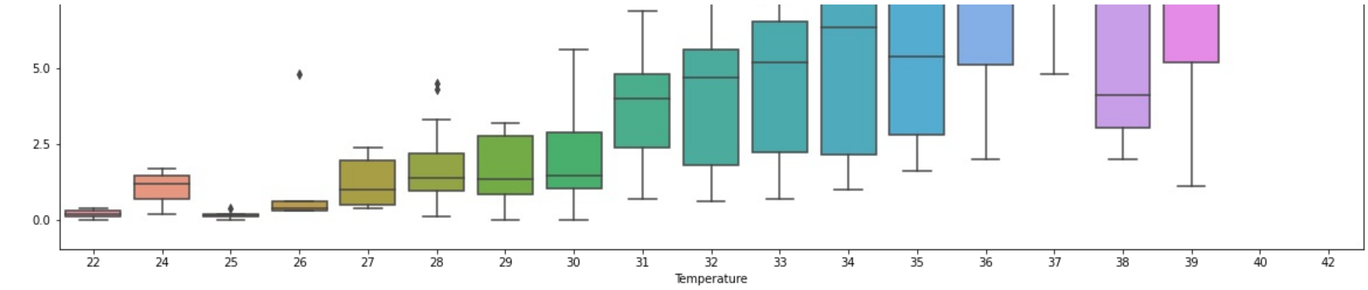
In [95]:

```python
sns.boxplot(x ='Temperature', y ='ISI', data = df)
```

Out[95]:

```
<AxesSubplot:xlabel='Temperature', ylabel='ISI'>
```

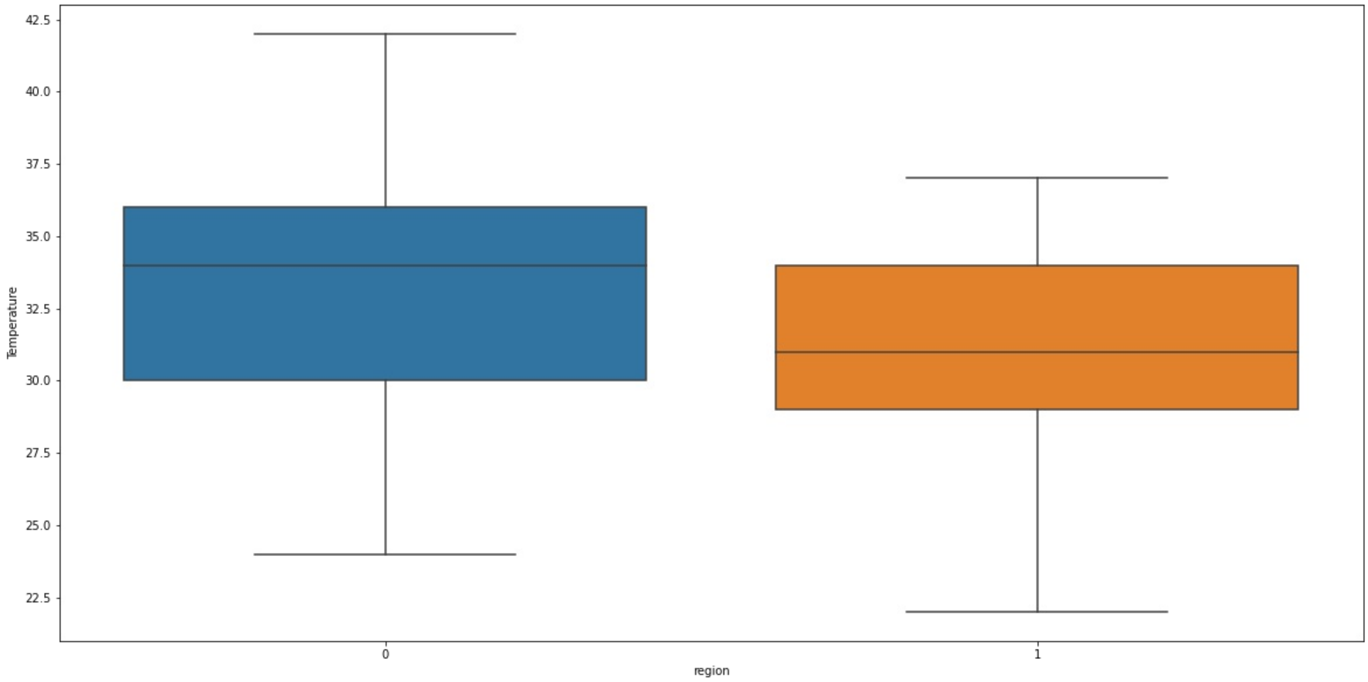




Boxplot of region Vs Temperature

In [96]:
```
sns.boxplot(x ='region', y ='Temperature', data = df)
```

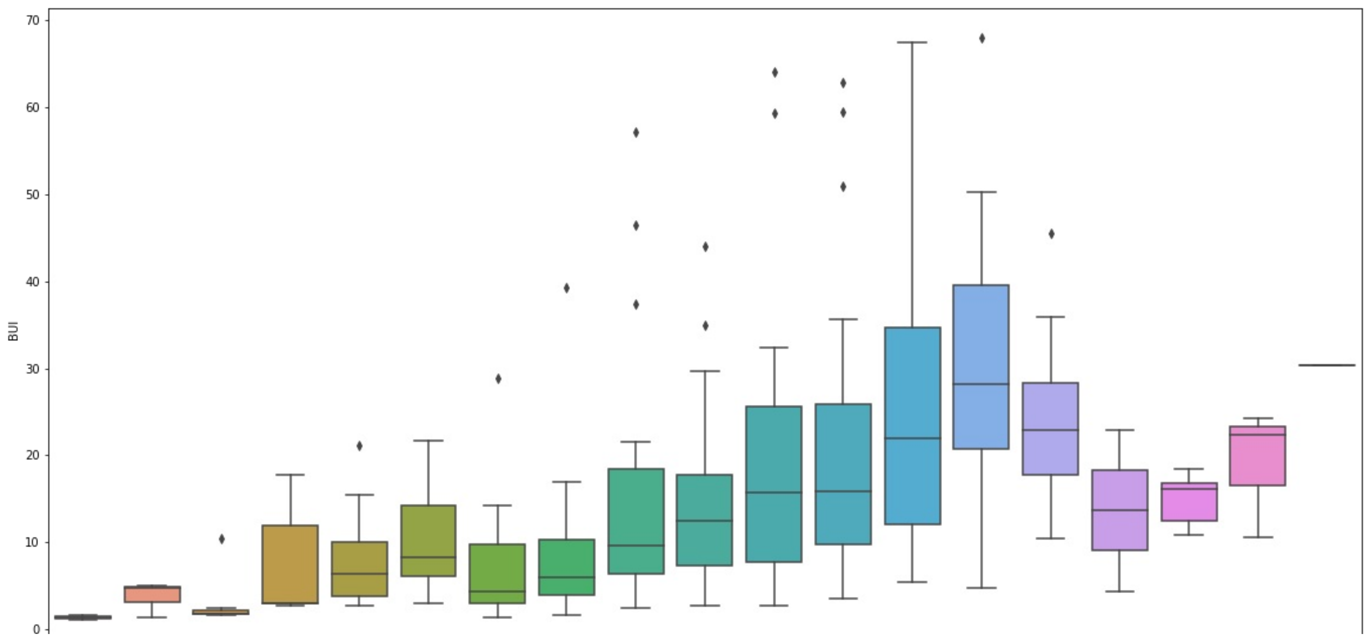
Out[96]: <AxesSubplot:xlabel='region', ylabel='Temperature'>

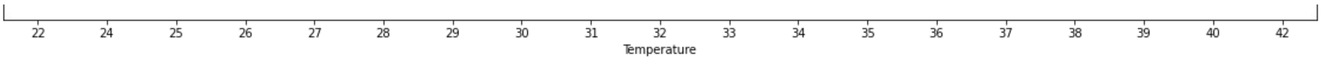


Boxplot of BUI Vs Temperature

In [97]:
```
sns.boxplot(x ='Temperature', y ='BUI', data = df)
```

Out[97]: <AxesSubplot:xlabel='Temperature', ylabel='BUI'>





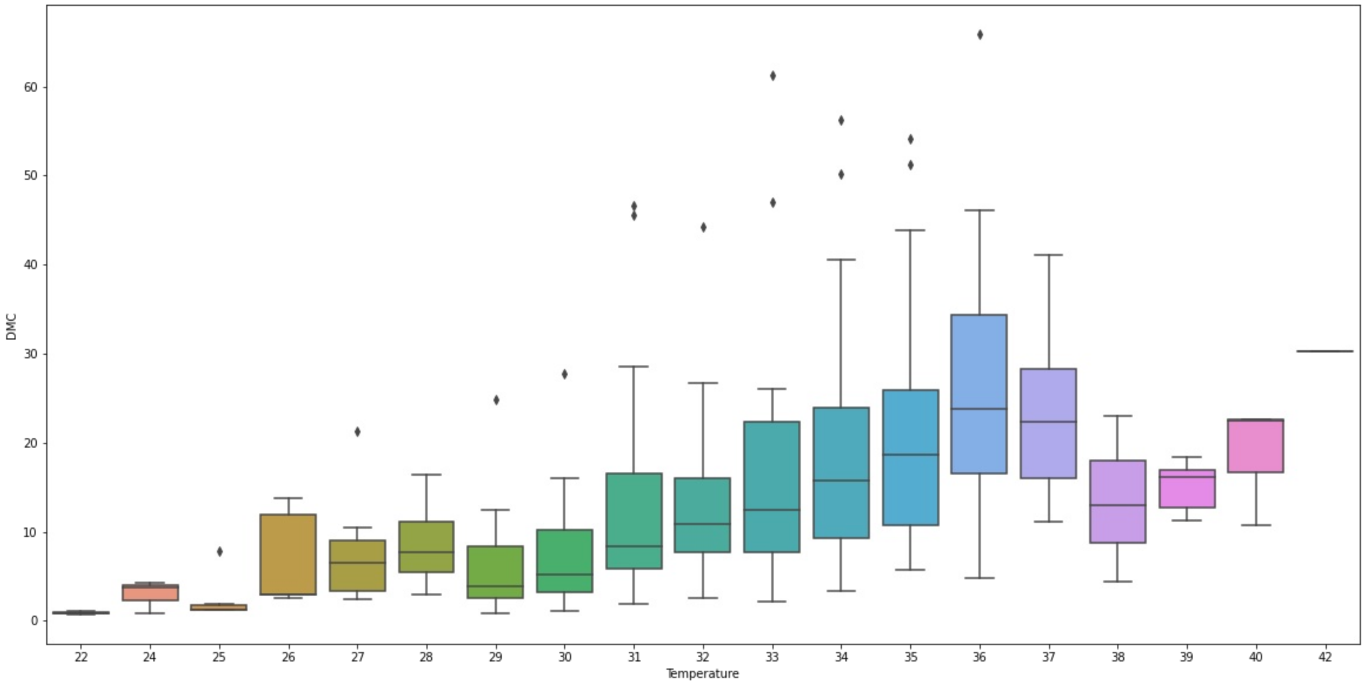
Boxplot DMC Vs Temperature

In [98]:

```
sns.boxplot(x ='Temperature', y ='DMC', data = df)
```

Out[98]:

```
<AxesSubplot:xlabel='Temperature', ylabel='DMC'>
```



Creating Dependent and Independent features

In [99]:

```
df.columns
```

Out[99]:

```
Index(['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
       'FWI', 'region', 'date'],
      dtype='object')
```

In [128…

```
## Independent Features
x=pd.DataFrame(df, columns=['RH','Ws','Rain','FFMC','DMC','DC','ISI','BUI','FWI','region'])
## Dependent Features
y=pd.DataFrame(df,columns=['Temperature'])
```

Independent Features

In [129…

```
x
```

Out[129…

	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
0	57	18	0.0	65.7	3.4	150	1.3	3.4	5	1
1	61	13	1.3	64.4	4.1	150	1.0	3.9	4	1
2	82	22	13.1	47.1	2.5	146	0.3	2.7	1	1
3	89	13	2.5	28.6	1.3	136	0.0	1.7	0	1
4	77	16	0.0	64.8	3.0	18	1.2	3.9	5	1
...	...	...	...	...	...	...	...	...	...	...
239	65	14	0.0	85.4	16.0	112	4.5	16.9	106	0

240	87	15	4.4	41.1	6.5	164	0.1	6.2	0	0
241	87	29	0.5	45.9	3.5	153	0.4	3.4	2	0
242	54	18	0.1	79.7	4.3	25	1.7	5.1	7	0
243	64	15	0.2	67.3	3.8	34	1.2	4.8	5	0

244 rows × 10 columns

Dependent Features

In [130…
```
y
```

Out[130…

	Temperature
0	29
1	29
2	26
3	25
4	27
...	...
239	30
240	28
241	27
242	24
243	24

244 rows × 1 columns

TrainTest Split

In [131…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=10)
```

In [132…
```
x_train.shape
```

Out[132…
```
(163, 10)
```

In [133…
```
x_test.shape
```

Out[133…
```
(81, 10)
```

In [134…
```
y_train.shape
```

Out[134…
```
(163, 1)
```

In [135…
```
y_test.shape
```

Out[135…
```
(81, 1)
```

Independent training dataset

In [136…
```
x_train
```

Out[136…

	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
237	49	6	2.0	61.3	11.9	77	0.6	11.9	4	0
78	54	18	0.0	89.4	20.0	8	9.7	27.5	47	1
25	64	18	0.0	86.8	17.8	157	6.7	21.6	20	1
124	80	14	2.0	48.7	2.2	150	0.3	2.6	1	0
176	64	9	1.2	73.8	11.7	28	1.1	11.4	7	0
...	...	...	...	...	...	...	...	...	...	...
64	69	13	0.0	85.0	8.2	53	4.0	8.2	86	1
15	89	13	0.7	36.1	1.7	150	0.0	2.2	0	1
228	51	13	0.0	88.7	16.0	122	6.9	17.8	124	0
125	64	14	0.0	79.4	5.2	26	2.2	5.6	10	0
9	79	12	0.0	73.2	9.5	114	1.3	12.6	9	1

163 rows × 10 columns

Independent Test Dataset

In [137…

```
x_test
```

Out[137…

	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
162	56	15	2.9	74.8	7.1	185	1.6	6.8	8	0
60	64	17	0.0	87.2	31.9	22	6.8	41.2	45	1
61	45	14	0.0	78.8	4.8	1	2.0	4.7	9	1
63	63	14	0.3	76.6	5.7	0	1.7	5.5	8	1
69	59	17	0.0	87.4	14.8	132	6.9	17.9	125	1
...	...	...	...	...	...	...	...	...	...	...
169	68	15	0.0	86.1	23.9	123	5.2	23.9	120	0
232	41	8	0.1	83.9	24.9	177	2.7	28.9	99	0
144	59	16	0.8	74.2	7.0	166	1.6	6.7	8	0
208	37	16	0.0	92.2	61.3	38	13.1	64.0	89	0
105	76	26	8.3	47.4	1.1	145	0.4	1.6	1	1

81 rows × 10 columns

Dependent Training Dataset

In [138…

```
y_train
```

Out[138…

	Temperature
237	26
78	36
25	31
124	29
176	39
...	...
64	34
15	29
228	32
125	30
9	28

163 rows × 1 columns

In [139…
```
y_test
```

Out[139…

	Temperature
162	34
60	35
61	36
63	35
69	35
...	...
169	33
232	29
144	33
208	33
105	22

81 rows × 1 columns

Standardizing or Feature Scaling

In [140…
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()  ## Initialising
```

In [141…
```
scaler
```

Out[141…
```
StandardScaler()
```

In [142…
```
x_train=scaler.fit_transform(x_train)
```

In [143…
```
x_test=scaler.transform(x_test)
```

In [144…
```
x_train
```

Out[144…
```
array([[-0.85631108, -3.36419461,  0.88853946, ..., -0.32535487,
        -1.03738328, -0.98176139],
       [-0.52508491,  0.99944243, -0.441414  , ...,  0.76565444,
        -0.01141751,  1.01857744],
       [ 0.13736742,  0.99944243, -0.441414  , ...,  0.35302912,
        -0.65562857,  1.01857744],
       ...,
       [-0.72382061, -0.81873967, -0.441414  , ...,  0.08727045,
         1.825777  , -0.98176139],
       [ 0.13736742, -0.45510325, -0.441414  , ..., -0.76595478,
        -0.89422526, -0.98176139],
       [ 1.13104591, -1.18237609, -0.441414  , ..., -0.27639932,
        -0.91808493,  1.01857744]])
```

In [145…
```
x_test
```

Out[145…
```
array([[-3.92594448e-01, -9.14668296e-02,  1.48701853e+00,
        -1.82411230e-01, -6.02677495e-01,  1.41938909e+00,
        -7.33442383e-01, -6.82030988e-01, -9.41944600e-01,
        -9.81761387e-01],
       [ 1.37367416e-01,  6.35806011e-01, -4.41414004e-01,
         6.64566895e-01,  1.37979749e+00, -1.44016117e+00,
         4.94418103e-01,  1.72378441e+00, -5.91368483e-02,
         1.01857744e+00],
       [-1.12129201e+00, -4.55103250e-01, -4.41414004e-01,
         9.08075201e-02, -7.86536062e-01, -1.80856948e+00,
        -6.38991577e-01, -8.28897625e-01, -9.18084931e-01,
         1.01857744e+00],
       [ 7.11221826e-02, -4.55103250e-01, -2.41920984e-01,
        -5.94627923e-02, -7.14591405e-01, -1.82611273e+00,
        -7.09829682e-01, -7.72948430e-01, -9.41944600e-01,
         1.01857744e+00],
       [-1.93858749e-01,  6.35806011e-01, -4.41414004e-01,
```

```
         6.78227832e-01,  1.28490116e-02,  4.89596678e-01,
         5.18030804e-01,  9.42640966e-02,  1.84963667e+00,
         1.01857744e+00],
       [-5.91330147e-01,  2.72169591e-01, -4.41414004e-01,
         8.21667676e-01,  1.83544698e+00, -1.19455562e+00,
         1.34447536e+00,  2.16438432e+00,  5.85074214e-01,
         1.01857744e+00],
       [-5.91330147e-01,  1.36307885e+00, -4.41414004e-01,
         8.01176270e-01,  1.96707579e-01,  1.61236488e+00,
         1.25002456e+00,  5.13883061e-01, -1.06856186e-01,
         1.01857744e+00],
       [-5.91330147e-01,  6.35806011e-01, -1.08925637e-01,
         1.86434082e-01,  4.84486205e-01, -1.40507466e+00,
        -4.73702665e-01,  9.82457572e-01,  1.27700461e+00,
        -9.81761387e-01],
       [ 5.34838813e-01, -4.55103250e-01, -4.41414004e-01,
         3.64026270e-01, -4.18818928e-01, -2.47219954e-01,
        -3.55639157e-01, -3.81304063e-01,  8.47530573e-01,
         1.01857744e+00],
       [ 7.11221826e-02, -4.55103250e-01, -4.41414004e-01,
         6.50905957e-01, -2.98911167e-01, -1.41960435e-01,
         2.11065683e-01, -2.83392971e-01,  1.44402230e+00,
         1.01857744e+00],
       [-3.92594448e-01,  2.72169591e-01, -4.41414004e-01,
         7.80684863e-01,  7.32295578e-01,  5.07139931e-01,
         8.24995926e-01,  5.06889412e-01, -3.45452876e-01,
        -9.81761387e-01],
       [ 9.98555444e-01,  2.09035169e+00,  7.55544118e-01,
        -1.29577764e+00, -1.01835773e+00,  1.10361054e+00,
        -8.51505892e-01, -9.89751562e-01, -1.06124295e+00,
         1.01857744e+00],
       [-1.84998957e+00,  2.72169591e-01, -3.08418657e-01,
         7.39702051e-01,  1.80719877e-01,  1.56274868e-01,
         6.59707014e-01,  6.62894990e-02, -6.79488242e-01,
        -9.81761387e-01],
       [ 6.01084046e-01, -4.55103250e-01,  3.88093477e+00,
        -8.85949510e-01, -9.06443823e-01,  1.36675933e+00,
        -8.75118593e-01, -9.12821418e-01, -1.03738328e+00,
        -9.81761387e-01],
       [-9.88801544e-01,  9.99442431e-01,  3.54844640e+00,
         2.27416895e-01, -3.86843525e-01,  1.43693235e+00,
        -3.79251858e-01, -5.00196103e-01,  7.52091897e-01,
        -9.81761387e-01],
       [ 5.34838813e-01,  2.72169591e-01, -4.41414004e-01,
         5.82601270e-01, -1.47028003e-01, -6.15628269e-01,
         1.63840280e-01, -2.69405673e-01,  1.42016263e+00,
        -9.81761387e-01],
       [-5.25084914e-01, -1.54601251e+00, -3.74916331e-01,
         4.25500489e-01, -4.98757435e-01, -5.80541763e-01,
        -3.79251858e-01, -5.07189753e-01,  7.75951566e-01,
        -9.81761387e-01],
       [ 1.06480068e+00, -4.55103250e-01,  4.89553424e-01,
        -2.21789092e+00, -1.01835773e+00,  7.87831981e-01,
        -1.06402021e+00, -9.89751562e-01, -1.10896228e+00,
         1.01857744e+00],
       [-1.78374434e+00, -9.14668296e-02, -4.41414004e-01,
         1.11537783e+00,  6.68344772e-01,  8.61018558e-02,
         2.59594855e+00,  4.43946567e-01,  5.37354876e-01,
        -9.81761387e-01],
       [-7.90065846e-01,  2.72169591e-01, -4.41414004e-01,
         8.48989551e-01,  1.44374829e+00,  8.75548246e-01,
         1.13196105e+00,  1.12233056e+00,  1.07880835e-01,
        -9.81761387e-01],
       [ 2.03612648e-01, -4.55103250e-01, -4.41414004e-01,
         4.80144239e-01, -1.71009555e-01,  4.19423665e-01,
        -1.66737544e-01, -5.26025410e-02,  1.22928528e+00,
         1.01857744e+00],
       [-1.93858749e-01,  1.36307885e+00, -4.41414004e-01,
         7.94345801e-01,  1.05204961e+00, -1.14192586e+00,
         1.20279915e+00,  1.59090507e+00,  2.51038848e-01,
        -9.81761387e-01],
       [-5.91330147e-01,  9.99442431e-01, -4.41414004e-01,
         7.87515332e-01,  5.56430862e-01,  1.20887006e+00,
         1.06112294e+00,  6.46762400e-01, -1.54575524e-01,
         1.01857744e+00],
       [ 4.87694966e-03,  2.72169591e-01, -4.41414004e-01,
         7.05549707e-01,  1.48744474e-01, -2.29676700e-01,
         5.41643506e-01, -1.06406445e-02,  1.75419799e+00,
        -9.81761387e-01],
       [-1.91623481e+00,  6.35806011e-01, -4.41414004e-01,
         1.10854736e+00,  1.96707579e-01, -3.17392966e-01,
         2.95013907e+00,  2.43276025e-02,  2.74898517e-01,
        -9.81761387e-01],
       [ 1.85974347e+00, -9.14668296e-02, -4.41414004e-01,
        -7.28848729e-01, -5.94683644e-01, -1.47524767e+00,
        -8.27893190e-01, -6.61050040e-01, -9.89663938e-01,
        -9.81761387e-01],
       [-6.13682833e-02,  9.99442431e-01, -2.41920984e-01,
         1.86434082e-01, -2.34960361e-01,  1.50710536e+00,
```

```
 -4.50089963e-01,  7.32831484e-02,  1.08612726e+00,
  1.01857744e+00],
[-9.88801544e-01, -4.55103250e-01,  2.40697100e-02,
  2.01142638e-03, -6.02677495e-01,  1.17378355e+00,
 -6.86216980e-01, -6.82030988e-01, -9.18084931e-01,
  1.01857744e+00],
[ 1.19729114e+00,  2.72169591e-01,  7.55544118e-01,
 -2.05395967e+00, -9.38419226e-01,  8.22918487e-01,
 -1.04040750e+00, -9.47789665e-01, -1.10896228e+00,
  1.01857744e+00],
[ 1.13104591e+00, -9.14668296e-02, -4.41414004e-01,
  5.41618457e-01,  1.10800656e+00, -1.54542068e+00,
 -1.44863200e-03,  1.45802573e+00, -6.31768904e-01,
  1.01857744e+00],
[ 1.26353638e+00,  2.09035169e+00,  3.41545106e+00,
 -1.97199404e+00, -9.30425375e-01,  8.22918487e-01,
 -1.01679480e+00, -9.47789665e-01, -1.10896228e+00,
  1.01857744e+00],
[ 1.59476254e+00,  2.72169591e-01, -4.41414004e-01,
 -2.70285420e+00, -1.09829624e+00,  1.06852403e+00,
 -1.08763291e+00, -1.05968806e+00, -1.13282195e+00,
 -9.81761387e-01],
[-5.25084914e-01, -8.18739670e-01, -4.41414004e-01,
  7.32871582e-01, -3.78849674e-01, -3.70022725e-01,
  3.99967296e-01, -3.95291362e-01,  1.51560130e+00,
  1.01857744e+00],
[-7.23820613e-01,  2.72169591e-01,  2.08549759e+00,
  2.01142638e-03, -5.30732838e-01,  1.41938909e+00,
 -6.38991577e-01, -6.19088143e-01, -8.46505925e-01,
 -9.81761387e-01],
[ 1.37367416e-01, -9.14668296e-02, -4.41414004e-01,
  6.30414551e-01, -3.51140928e-02,  6.47485956e-01,
  2.34678385e-01,  1.22238694e-01,  1.65875932e+00,
  1.01857744e+00],
[ 3.36103114e-01,  1.36307885e+00, -4.41414004e-01,
 -1.41428417e-01, -9.38419226e-01, -1.24718538e+00,
 -6.38991577e-01, -8.77853171e-01, -9.41944600e-01,
 -9.81761387e-01],
[ 1.19729114e+00,  2.72169591e-01, -1.75423310e-01,
 -1.20698154e+00, -8.98449972e-01, -5.45455257e-01,
 -8.98731295e-01, -8.00923028e-01, -1.03738328e+00,
  1.01857744e+00],
[ 4.87694966e-03, -9.14668296e-02, -4.41414004e-01,
  3.98178614e-01, -7.06597555e-01, -1.08929610e+00,
 -2.13962947e-01, -7.10005585e-01,  7.99811235e-01,
 -9.81761387e-01],
[ 1.32978161e+00, -9.14668296e-02, -1.75423310e-01,
 -2.22472138e+00, -1.09829624e+00,  7.52745475e-01,
 -1.06402021e+00, -1.05968806e+00, -1.13282195e+00,
  1.01857744e+00],
[-9.22556312e-01, -8.18739670e-01, -4.41414004e-01,
  8.76311426e-01,  6.04393966e-01, -1.70330996e+00,
  9.43059434e-01,  8.98533779e-01, -1.06856186e-01,
  1.01857744e+00],
[ 4.02348347e-01,  1.36307885e+00, -4.41414004e-01,
  7.39702051e-01,  9.00166443e-01, -1.56296394e+00,
  9.66672136e-01,  1.26919720e+00,  6.01614966e-02,
  1.01857744e+00],
[-4.58839681e-01, -9.14668296e-02, -4.41414004e-01,
  8.08006738e-01,  1.09201886e+00,  7.00115715e-01,
  8.48608628e-01,  8.21603635e-01, -2.26154531e-01,
 -9.81761387e-01],
[-1.32002771e+00,  2.09035169e+00, -4.41414004e-01,
  8.96802832e-01,  2.84639937e-01, -3.70022725e-01,
  2.05285641e+00,  1.01257746e-01,  3.63018276e-02,
 -9.81761387e-01],
[ 8.66064978e-01,  2.72169591e-01, -4.41414004e-01,
  3.23043457e-01, -8.18511465e-01, -6.85801282e-01,
 -3.32026455e-01, -7.37980183e-01,  3.94196862e-01,
 -9.81761387e-01],
[-1.38627294e+00, -1.90964893e+00, -3.74916331e-01,
  9.92429394e-01,  6.36369369e-01,  6.65029209e-01,
  1.13196105e+00,  5.34864009e-01, -1.78435193e-01,
 -9.81761387e-01],
[-4.58839681e-01, -4.55103250e-01, -4.41414004e-01,
  7.80684863e-01,  3.16615340e-01,  6.82572462e-01,
  6.36094313e-01,  3.74010073e-01, -5.60189897e-01,
  1.01857744e+00],
[ 1.19729114e+00,  2.09035169e+00,  1.07301951e+01,
 -1.70560576e+00, -4.74775883e-01,  1.15624030e+00,
 -9.69569400e-01, -5.77126247e-01, -1.06124295e+00,
  1.01857744e+00],
[-1.18753724e+00, -8.18739670e-01, -4.41414004e-01,
  8.62650488e-01, -1.63015705e-01, -9.31406825e-01,
  8.48608628e-01, -2.83392971e-01,  1.77805766e+00,
 -9.81761387e-01],
[-4.58839681e-01, -9.14668296e-02, -4.41414004e-01,
  7.94345801e-01,  5.00473906e-01,  1.21188362e-01,
  7.77770523e-01,  2.97079929e-01, -5.12470559e-01,
```

```
 -9.81761387e-01],
[ 1.72725301e+00, -8.18739670e-01, -4.41414004e-01,
 -4.41969042e-01, -9.62400778e-01, -1.21209888e+00,
 -8.27893190e-01, -8.98834119e-01, -1.01352361e+00,
  1.01857744e+00],
[ 1.06480068e+00, -4.55103250e-01, -4.41414004e-01,
  2.41077832e-01, -6.66628301e-01, -3.34936219e-01,
 -4.97315367e-01, -5.70132597e-01,  3.46477524e-01,
  1.01857744e+00],
[ 4.02348347e-01, -4.55103250e-01, -4.41414004e-01,
  5.41618457e-01, -2.02984958e-01,  1.03645109e-01,
 -2.50613337e-02, -1.64500931e-01,  1.30086428e+00,
  1.01857744e+00],
[-1.25378248e+00,  2.72169591e-01, -4.41414004e-01,
  9.37785644e-01,  2.51492429e+00, -1.52787743e+00,
  1.60421508e+00,  2.35321286e+00,  6.32793552e-01,
 -9.81761387e-01],
[ 3.36103114e-01, -4.55103250e-01, -4.41414004e-01,
  3.50365332e-01, -7.06597555e-01, -7.91060800e-01,
 -3.79251858e-01, -6.68043689e-01,  3.94196862e-01,
  1.01857744e+00],
[ 1.46227207e+00,  9.99442431e-01, -4.41414004e-01,
  4.32330957e-01, -9.10710479e-02,  2.79077640e-01,
 -4.86740353e-02, -3.86152421e-02,  1.34858362e+00,
  1.01857744e+00],
[ 6.01084046e-01,  6.35806011e-01, -2.41920984e-01,
 -5.37595604e-01, -9.14437674e-01, -4.05109232e-01,
 -7.57055085e-01, -8.00923028e-01, -9.89663938e-01,
  1.01857744e+00],
[ 2.69857881e-01, -9.14668296e-02, -3.74916331e-01,
  3.57195801e-01,  1.44374829e+00,  1.57727837e+00,
 -3.32026455e-01,  1.32514639e+00,  1.58718031e+00,
 -9.81761387e-01],
[-1.65125387e+00, -8.18739670e-01, -4.41414004e-01,
  1.02658174e+00,  1.00408650e+00,  3.49250653e-01,
  1.65144048e+00,  7.37679842e-01,  1.79459842e-01,
 -9.81761387e-01],
[ 7.11221826e-02,  6.35806011e-01,  2.90060404e-01,
 -3.19020605e-01,  5.00473906e-01,  4.72053425e-01,
 -7.33442383e-01,  3.60022774e-01,  3.94196862e-01,
 -9.81761387e-01],
[-8.56311079e-01,  1.36307885e+00, -4.41414004e-01,
  7.60193457e-01, -2.50948063e-01, -2.64763207e-01,
  1.03751024e+00, -2.90386621e-01, -7.03347911e-01,
 -9.81761387e-01],
[ 2.03612648e-01, -9.14668296e-02, -3.74916331e-01,
  2.68399707e-01, -1.86997257e-01,  6.12399450e-01,
 -4.50089963e-01, -3.64699509e-03,  1.03840792e+00,
  1.01857744e+00],
[ 7.33574512e-01, -9.14668296e-02, -4.41414004e-01,
  6.23584082e-01,  9.64117249e-01, -1.59805044e+00,
  2.11065683e-01,  1.29017815e+00, -5.36330228e-01,
  1.01857744e+00],
[-6.13682833e-02, -8.18739670e-01,  4.23055751e-01,
 -8.92779979e-01, -8.42493017e-01,  8.05375234e-01,
 -8.75118593e-01, -8.84846821e-01, -1.03738328e+00,
  1.01857744e+00],
[ 1.32978161e+00,  2.45398811e+00,  8.26978122e+00,
 -2.07445107e+00, -9.70394629e-01,  7.35202221e-01,
 -1.04040750e+00, -9.68770614e-01, -1.10896228e+00,
  1.01857744e+00],
[-1.27613516e-01,  9.99442431e-01, -2.41920984e-01,
 -2.53104486e-02, -2.66935764e-01,  2.08904628e-01,
 -5.91766173e-01, -1.71494581e-01,  4.18056531e-01,
  1.01857744e+00],
[-1.84998957e+00, -4.55103250e-01, -4.41414004e-01,
  1.08122549e+00, -3.06905018e-01, -8.26147307e-01,
  2.14730722e+00, -4.16272310e-01, -3.21593207e-01,
 -9.81761387e-01],
[ 1.66100777e+00,  9.99442431e-01, -4.41414004e-01,
  1.72773145e-01, -2.26966510e-01, -4.57738991e-01,
 -4.50089963e-01, -3.32348517e-01,  7.99811235e-01,
 -9.81761387e-01],
[ 2.69857881e-01,  2.45398811e+00, -1.75423310e-01,
 -6.33222167e-01, -3.30886570e-01,  8.93091499e-01,
 -6.86216980e-01, -8.05771386e-02,  3.22617855e-01,
  1.01857744e+00],
[-5.91330147e-01, -4.55103250e-01, -4.41414004e-01,
  8.21667676e-01,  2.11523175e+00, -1.58050719e+00,
  7.77770523e-01,  2.02451133e+00,  1.55600173e-01,
 -9.81761387e-01],
[-4.58839681e-01, -4.55103250e-01, -4.41414004e-01,
  5.96262207e-01, -5.06751286e-01, -1.01912309e+00,
  6.93894730e-02, -5.84119896e-01,  1.13384660e+00,
 -9.81761387e-01],
[ 9.32310211e-01,  1.72671527e+00,  2.40697100e-02,
 -9.81576073e-01, -9.62400778e-01,  1.38430259e+00,
 -8.04280488e-01, -9.47789665e-01, -1.01352361e+00,
  1.01857744e+00],
```

```
       [-5.25084914e-01,  9.99442431e-01, -3.74916331e-01,
         1.52281739e-01, -8.26505316e-01, -1.38753141e+00,
        -7.09829682e-01, -8.00923028e-01, -9.65804269e-01,
        -9.81761387e-01],
       [ 7.33574512e-01, -9.14668296e-02, -4.41414004e-01,
         6.23584082e-01, -2.02984958e-01, -7.17874223e-02,
         2.11065683e-01, -2.13456477e-01,  1.46788197e+00,
         1.01857744e+00],
       [ 1.37367416e-01, -9.14668296e-02, -3.08418657e-01,
        -6.94696386e-01, -8.66474569e-01, -1.22964213e+00,
        -8.27893190e-01, -8.21903976e-01, -1.01352361e+00,
        -9.81761387e-01],
       [ 4.68593580e-01, -8.18739670e-01,  2.40697100e-02,
        -7.42509667e-01, -6.90609853e-01,  1.40184584e+00,
        -8.51505892e-01, -7.51967482e-01, -1.01352361e+00,
         1.01857744e+00],
       [ 1.32978161e+00,  2.09035169e+00, -4.41414004e-01,
         5.07466114e-01,  2.58686895e+00, -8.43690560e-01,
        -7.22867370e-02,  2.98963495e+00, -3.45452876e-01,
         1.01857744e+00],
       [ 4.02348347e-01, -9.14668296e-02, -4.41414004e-01,
         5.89431738e-01,  7.40289429e-01,  3.31707400e-01,
         1.16614876e-01,  5.13883061e-01,  1.73033833e+00,
        -9.81761387e-01],
       [-1.38627294e+00, -2.63692177e+00, -3.74916331e-01,
         4.39161426e-01,  8.20227936e-01,  1.27904307e+00,
        -4.73702665e-01,  8.63565532e-01,  1.22928528e+00,
        -9.81761387e-01],
       [-1.93858749e-01,  2.72169591e-01,  9.05673835e-02,
        -2.23394042e-01, -6.10671346e-01,  1.08606728e+00,
        -7.33442383e-01, -6.89024637e-01, -9.41944600e-01,
        -9.81761387e-01],
       [-1.65125387e+00,  2.72169591e-01, -4.41414004e-01,
         1.00609033e+00,  3.72998960e+00, -1.15946912e+00,
         1.98201831e+00,  3.31833647e+00,  9.90688587e-01,
        -9.81761387e-01],
       [ 9.32310211e-01,  3.90853379e+00,  5.07789289e+00,
        -2.05395967e+00, -1.08230854e+00,  7.17658968e-01,
        -1.01679480e+00, -1.04570076e+00, -1.10896228e+00,
         1.01857744e+00]])
```

Model Training

In [146…
```python
from sklearn.linear_model import LinearRegression
```

In [147…
```python
regression=LinearRegression()
```

In [148…
```python
regression
regression.fit(x_train,y_train)
```

Out[148…
```
LinearRegression()
```

Coefficient

In [149…
```python
print(regression.coef_)
```

```
[[-1.27500995 -0.53842199 -0.21205266  0.70886534 -1.02729123 -0.32455869
   0.2501139   1.35400654  0.21687466 -0.23115864]]
```

Intercept

In [150…
```python
print(regression.intercept_)
```

```
[32.17791411]
```

Precdiction for Test Data

In [151…
```
reg_pred=regression.predict(x_test)
```

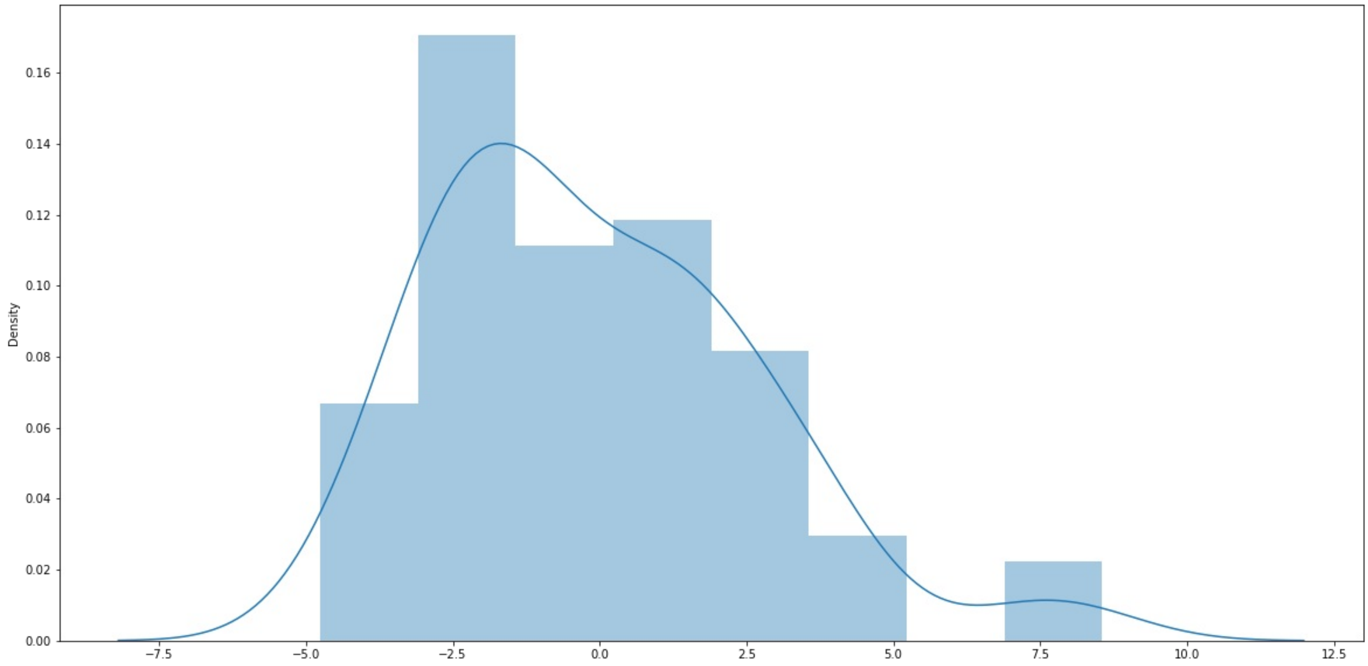
In [152…
```
reg_pred
```

Out[152…
```
array([[31.35728286],
       [33.48448971],
       [33.68885621],
       [32.00434093],
       [32.90791395],
       [35.12184072],
       [32.88392257],
       [34.41877632],
       [31.94627835],
       [32.98721461],
       [33.30675698],
       [27.36975802],
       [35.07784211],
       [29.24028529],
       [31.85823402],
       [32.41802576],
       [34.37535576],
       [28.09756027],
       [36.26505018],
       [34.01992072],
       [32.55507349],
       [34.37404142],
       [32.95376928],
       [33.26908507],
       [36.11652279],
       [29.43281151],
       [31.64047533],
       [32.38849979],
       [27.56606986],
       [32.22728534],
       [25.99441341],
       [27.23155419],
       [34.06867619],
       [31.64327002],
       [32.76692249],
       [31.05185077],
       [29.01675218],
       [33.06175783],
       [27.69372403],
       [35.63560078],
       [32.8693709 ],
       [33.63210892],
       [34.17783984],
       [31.5433198 ],
       [36.08261913],
       [33.41675348],
       [24.66437356],
       [35.74882134],
       [33.62798919],
       [29.69949092],
       [31.06668332],
       [32.38004487],
       [36.25233174],
       [32.16965552],
       [30.17098904],
       [30.08639562],
       [32.50310102],
       [36.07831078],
       [31.40637145],
       [33.45272801],
       [32.10289562],
       [32.80364988],
       [30.70110717],
       [24.64737332],
       [31.51727723],
       [36.35580039],
       [29.95761627],
       [29.70472774],
       [35.38938777],
       [34.07489424],
       [27.95824128],
       [32.55161796],
       [31.90597354],
       [31.60138869],
       [30.05790994],
       [31.14615789],
       [32.6846203 ],
       [36.0426106 ],
       [31.28209795],
       [36.91863039],
       [25.08477191]])
```

In [154…

```
import seaborn as sns
sns.distplot(reg_pred-y_test)
```

Out[154… <AxesSubplot:ylabel='Density'>

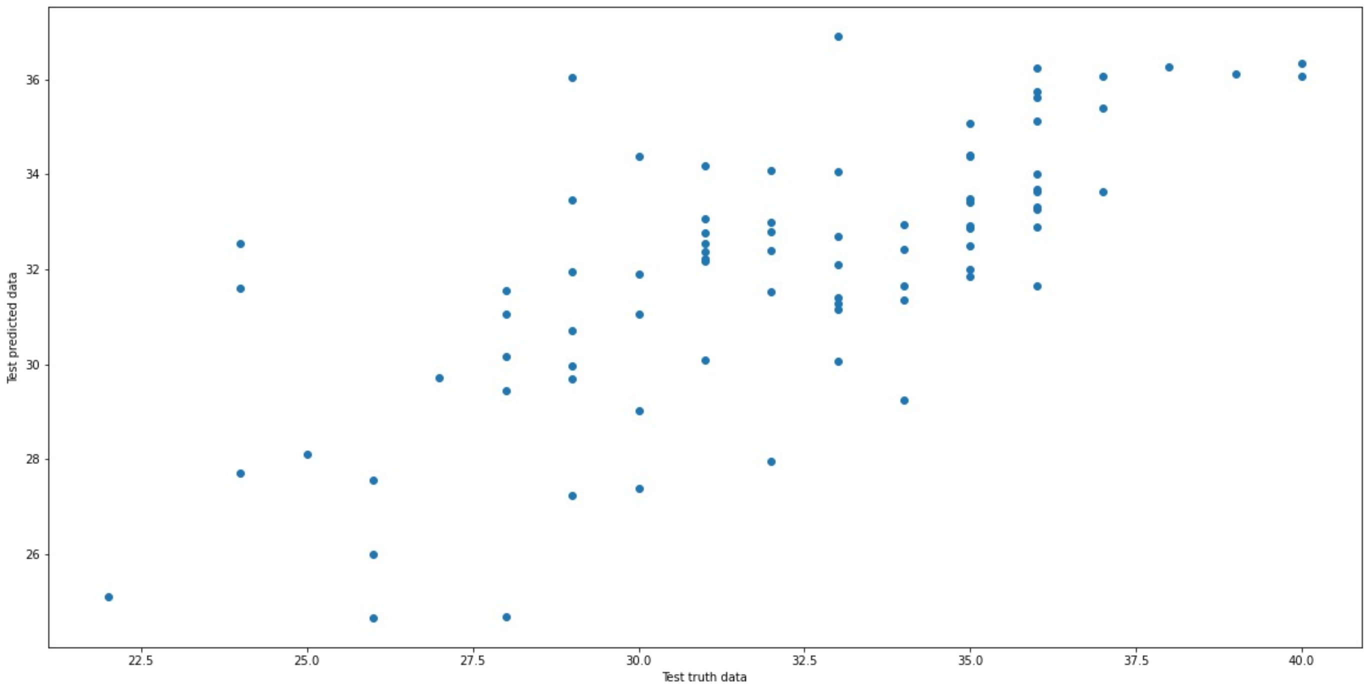


Assumption of Linear Regression

In [155…

```
plt.scatter(y_test,reg_pred)
plt.xlabel("Test truth data")
plt.ylabel('Test predicted data')
```

Out[155… Text(0, 0.5, 'Test predicted data')



Residuals

In [156…

```
residual=y_test-reg_pred
```

In [157…

```
residual
```

Out[157…

	Temperature
162	2.642717
60	1.515510
61	2.311144
63	2.995659
69	2.092086
...	...
169	0.315380
232	-7.042611
144	1.717902
208	-3.918630
105	-3.084772

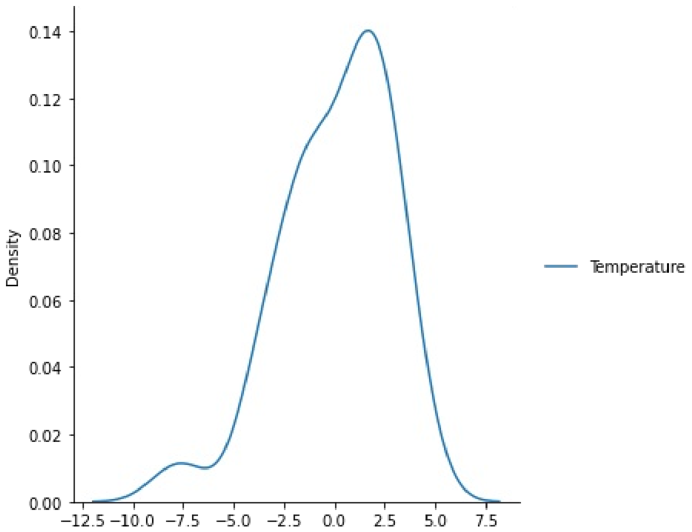
81 rows × 1 columns

In [158…

```
sns.displot(residual,kind='kde')
```

Out[158…

```
<seaborn.axisgrid.FacetGrid at 0x1c39909b550>
```



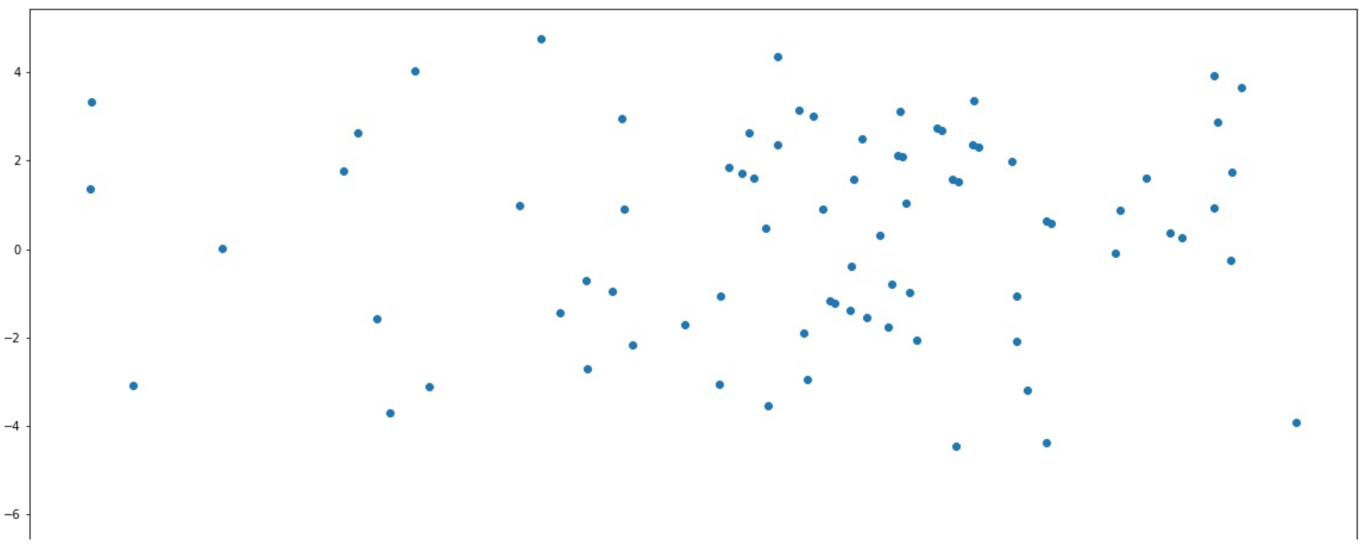
Scatterplot with prediction and residual

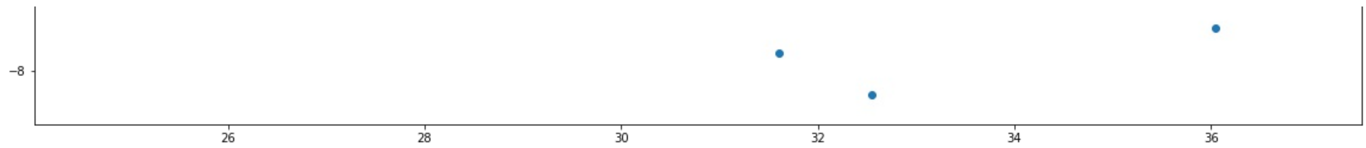
In [159…

```
plt.scatter(reg_pred,residual)
```

Out[159…

```
<matplotlib.collections.PathCollection at 0x1c39909a1c0>
```





Performance Metrics

In [160…

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,reg_pred))
print(mean_absolute_error(y_test,reg_pred))
print(np.sqrt(mean_squared_error(y_test,reg_pred)))
```

```
7.504766973062432
2.2354993752323624
2.739482975501478
```

R square and adjusted R square

In [161…

```
from sklearn.metrics import r2_score
score=r2_score(y_test,reg_pred)
print(score)
```

```
0.5037314185907535
```

Adjusted R square

In [162…

```
1-(1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[162…

```
0.43283590696086116
```

Ridege Regression Algorithm

In [163…

```
from sklearn.linear_model import Ridge
```

In [164…

```
ridge=Ridge()
```

In [165…

```
ridge
```

Out[165…

```
Ridge()
```

In [166…

```
ridge.fit(x_train,y_train)
```

Out[166…

```
Ridge()
```

In [167…

```
## Coefficient

print(ridge.coef_)
```

```
[[-1.25483718 -0.53296814 -0.20702885  0.72895886 -0.60290935 -0.32591751
   0.24045074  0.93900788  0.21118476 -0.1916003 ]]
```

In [168…

```
## Intercept
```

```
print(ridge.intercept_)
```

```
[32.17791411]
```

In [169…

```
ridge_pred=ridge.predict(x_test)
```

In [170…

```
ridge_pred
```

Out[170…

```
array([[31.35162743],
       [33.40986612],
       [33.72771064],
       [32.07338971],
       [32.90931259],
       [35.0319983 ],
       [32.79015818],
       [34.17178646],
       [31.97942736],
       [33.01801414],
       [33.36867288],
       [27.41069432],
       [35.06294721],
       [29.21943029],
       [31.86814167],
       [32.4415311 ],
       [34.32301905],
       [28.10883976],
       [36.28106232],
       [34.1156651 ],
       [32.5477257 ],
       [34.12721932],
       [32.95790766],
       [33.29642758],
       [36.10629748],
       [29.45172695],
       [31.55350828],
       [32.44406184],
       [27.6048532 ],
       [32.16940933],
       [26.0626233 ],
       [27.15902083],
       [34.09770279],
       [31.64258349],
       [32.74172122],
       [31.00145744],
       [29.02397313],
       [33.02208479],
       [27.70413671],
       [35.54562191],
       [32.78559985],
       [33.71213911],
       [34.19881056],
       [31.48850848],
       [36.06072934],
       [33.43404309],
       [24.81171141],
       [35.7280546 ],
       [33.67748084],
       [29.73906751],
       [31.08558962],
       [32.40946204],
       [36.27999474],
       [32.1989361 ],
       [30.22249483],
       [30.08807274],
       [32.52930384],
       [36.1226345 ],
       [31.43466997],
       [33.42545678],
       [32.07007099],
       [32.74547017],
       [30.74281953],
       [24.73332347],
       [31.51966611],
       [36.32138028],
       [30.00101555],
       [29.64684244],
       [35.40167579],
       [34.05569841],
       [28.00372035],
       [32.50468393],
       [31.96302548],
       [31.53812647],
       [30.11946542],
       [31.09359251],
```

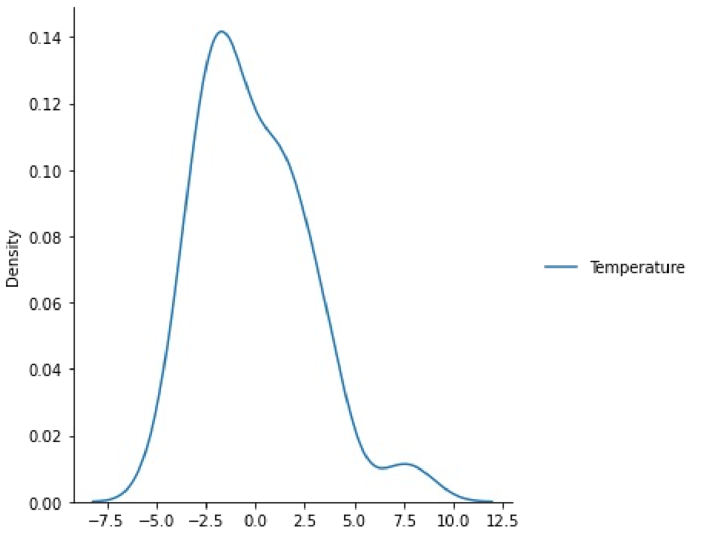
```
       [32.75250885],
       [35.95392448],
       [31.27455865],
       [37.0485867 ],
       [25.1392405 ]])
```

In [171…

```python
import seaborn as sns
sns.displot(ridge_pred-y_test,kind='kde')
```

Out[171…

```
<seaborn.axisgrid.FacetGrid at 0x1c38911c880>
```



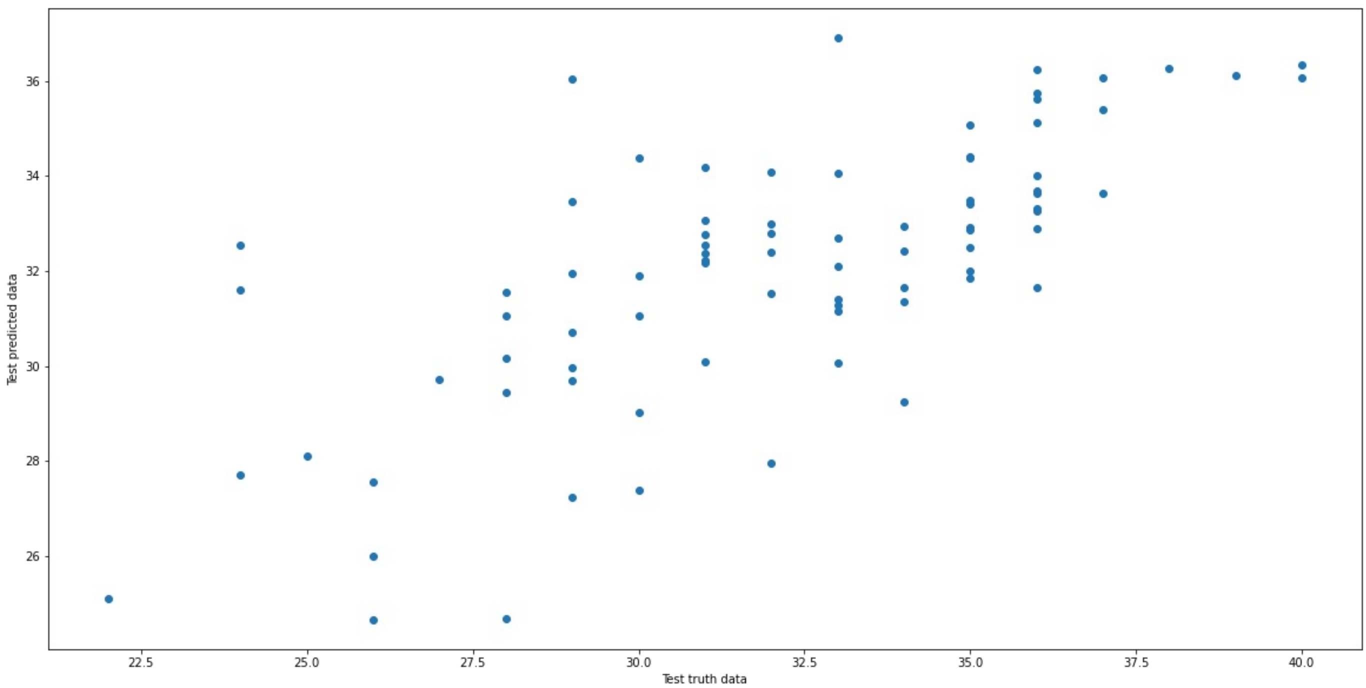
Assumption on Ridge Regression

In [172…

```python
plt.scatter(y_test,reg_pred)
plt.xlabel("Test truth data")
plt.ylabel('Test predicted data')
```

Out[172…

```
Text(0, 0.5, 'Test predicted data')
```



In [173…

```python
# Residual

residual=y_test-ridge_pred
```

In [174…

```python
residual
```

```
residual
```

Out[174...

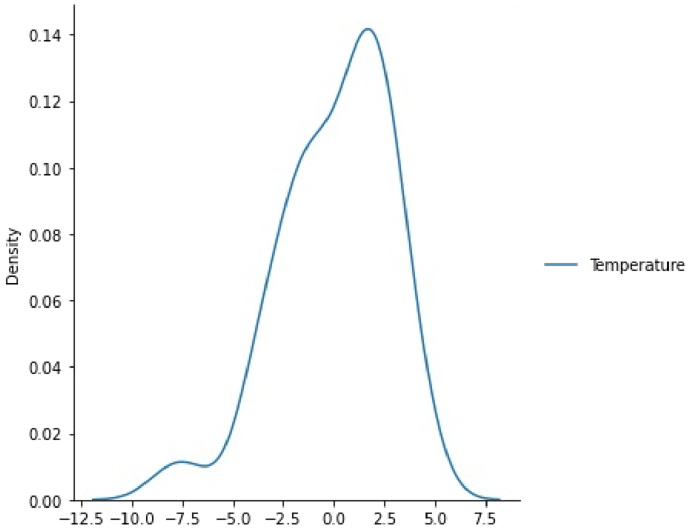
	Temperature
162	2.648373
60	1.590134
61	2.272289
63	2.926610
69	2.090687
...	...
169	0.247491
232	-6.953924
144	1.725441
208	-4.048587
105	-3.139240

81 rows × 1 columns

In [175...

```python
sns.displot(residual,kind='kde')
```

Out[175... <seaborn.axisgrid.FacetGrid at 0x1c39956b520>

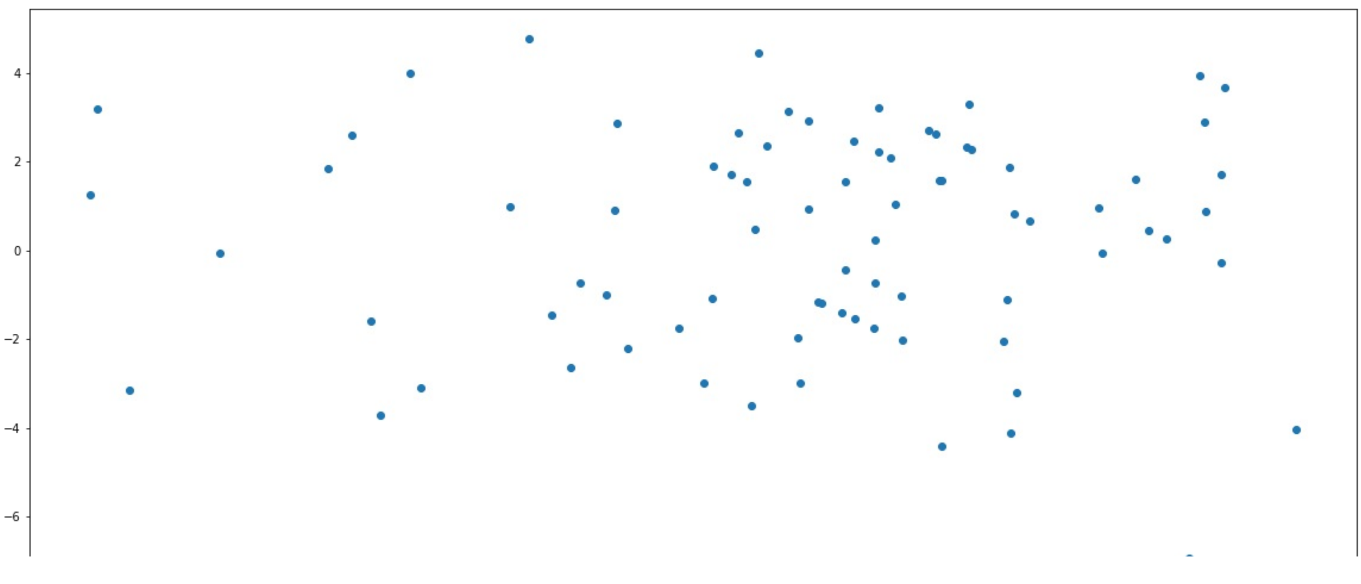


Scatter plot with residual and prediction

In [176...

```python
plt.scatter(ridge_pred,residual)
```

Out[176... <matplotlib.collections.PathCollection at 0x1c39c15e370>



Performance Matrics

In [177…

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,ridge_pred))
print(mean_absolute_error(y_test,ridge_pred))
print(np.sqrt(mean_squared_error(y_test,ridge_pred)))
```

```
7.444528719288363
2.2355295943587614
2.728466367630058
```

R square

In [178…

```
from sklearn.metrics import r2_score
ridge_score=r2_score(y_test,ridge_pred)
print(ridge_score)
```

```
0.5077148004671435
```

Adjusted R square

In [179…

```
1-(1-ridge_score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[179…

```
0.43738834339102106
```

Lasso Regression

In [180…

```
from sklearn.linear_model import Lasso
```

In [181…

```
lasso=Lasso()
```

In [182…

```
lasso
```

Out[182…

```
Lasso()
```

In [183…

```
lasso.fit(x_train,y_train)
```

Out[183…

```
Lasso()
```

Coefficients and Intercepts

In [184…

```
print(lasso.coef_)
```

```
[-0.71955751 -0.         -0.          0.89582004  0.         -0.
  0.          0.          0.         -0.        ]
```

In [185…

```
print(lasso.intercept_)
```

```
[32.17791411]
```

In [186…
```
## Prediction for test data

lasso_pred = lasso.predict(x_test)
```

In [187…
```
lasso_pred
```

Out[187…
```
array([32.29700076, 32.6744027 , 33.06609539, 32.07346965, 32.92497671,
       33.33947653, 33.32111992, 32.77042154, 32.11916885, 32.70983221,
       33.15976154, 30.29861247, 34.17172792, 30.95174825, 33.0931383 ,
       32.31497272, 32.93691477, 29.42489766, 34.46059856, 33.50695377,
       32.46152593, 33.02899752, 33.30888217, 32.80645043, 34.5498142 ,
       30.18680443, 32.38908351, 32.89121556, 29.47641605, 31.8492542 ,
       29.50217524, 28.6091198 , 33.21226395, 32.70054654, 32.64380834,
       31.80937418, 30.23515603, 32.53110125, 29.22810977, 33.62676377,
       32.55104126, 33.23190428, 33.93112391, 31.84411936, 34.06445535,
       33.20742879, 29.78847846, 33.80519505, 33.21966653, 30.53913152,
       31.62769114, 32.373594  , 33.92016988, 32.24993288, 31.51301599,
       31.26381066, 32.303719  , 34.28571873, 31.84095256, 33.47507571,
       32.27184094, 32.20868418, 31.42230192, 29.36272493, 32.24706577,
       34.47767146, 31.13749714, 31.41648274, 33.33947653, 33.04221928,
       30.62774778, 32.69215994, 32.20868418, 31.45674741, 31.17557904,
       31.67565808, 32.4164261 , 33.56882682, 32.11728577, 34.26736212,
       29.66708507])
```

Performance Matrics

In [188…
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,lasso_pred))
print(mean_absolute_error(y_test,lasso_pred))
print(np.sqrt(mean_squared_error(y_test,lasso_pred)))
```

```
9.10609532182792
2.4978660766652734
3.0176307464346794
```

R square

In [189…
```
from sklearn.metrics import r2_score
lasso_score=r2_score(y_test,lasso_pred)
print(lasso_score)
```

```
0.39784019626969913
```

Adjusted R square

In [190…
```
1-(1-lasso_score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[190…
```
0.31181736716537045
```

Elastic - Net regression

In [191…
```
from sklearn.linear_model import ElasticNet
```

In [192…
```
elastic=ElasticNet()
```

In [193…
```
elastic
```

Out[193… ElasticNet()

In [194…
```
elastic.fit(x_train,y_train)
```

Out[194… ElasticNet()

Coefficients and Intercepts

In [195…
```
print(elastic.coef_)
```

```
[-0.69396083 -0.10315403 -0.01507374  0.6926462   0.10752205 -0.
  0.28392506  0.07544656  0.05920494 -0.        ]
```

In [197…
```
print(elastic.intercept_)
```

```
[32.17791411]
```

In [198…
```
## Prediction for test data
elastic_pred = elastic.predict(x_test)
```

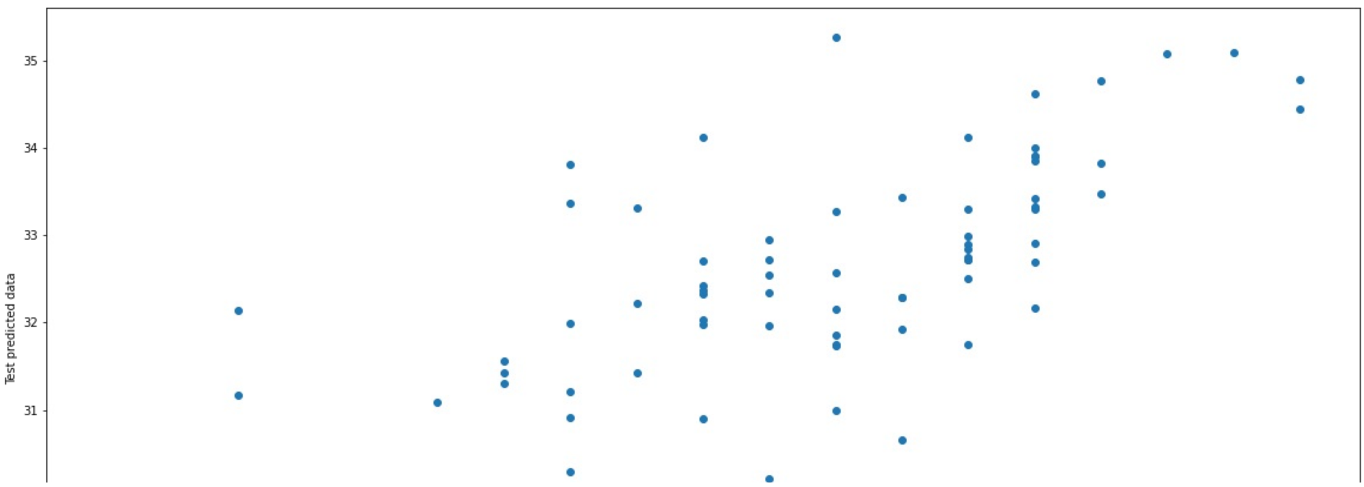
In [199…
```
elastic_pred
```

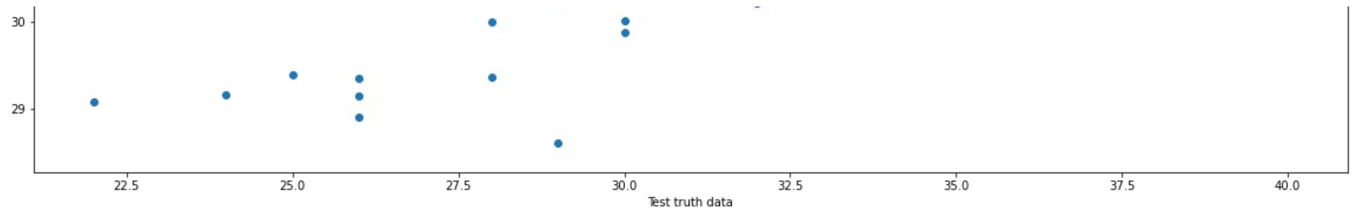
Out[199…
```
array([31.93076461, 32.89925279, 32.68965549, 31.74550697, 32.98836724,
       33.91299333, 33.41776043, 32.72078781, 31.98790062, 32.72490467,
       33.30044153, 29.87165926, 34.12217082, 30.65936304, 32.72255437,
       32.28333638, 32.84852051, 29.39040946, 35.0786389 , 33.86050655,
       32.42586731, 33.31820617, 33.43805315, 32.91463826, 35.09348452,
       29.99111145, 32.16696212, 32.5466324 , 29.35144729, 31.97557447,
       29.14216505, 28.60375228, 33.27376701, 32.28678221, 32.70561446,
       31.30843328, 30.01197805, 32.32347658, 29.15902087, 33.91042464,
       32.74766721, 33.47903767, 34.12940549, 31.56461381, 34.44958031,
       33.30036678, 29.35558024, 33.99794233, 33.3293338 , 30.2978916 ,
       31.4241864 , 32.36311525, 34.61701506, 32.03023331, 31.41948172,
       30.89419347, 32.508051  , 34.7691147 , 31.73370163, 33.3687946 ,
       32.1509131 , 32.34603101, 31.21296836, 28.90195873, 31.96458545,
       34.79046967, 30.91855019, 31.0841594 , 33.82121587, 32.95120309,
       30.20921418, 32.14232146, 32.22575701, 31.16525171, 30.98976689,
       31.86034438, 32.57697932, 33.81340658, 31.74661525, 35.27207415,
       29.07891997])
```

Assumption of Elastic Net Regression

In [200…
```
plt.scatter(y_test,elastic_pred)
plt.xlabel("Test truth data")
plt.ylabel('Test predicted data')
```

Out[200… Text(0, 0.5, 'Test predicted data')





Performance Matrix

In [201…

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,elastic_pred))
print(mean_absolute_error(y_test,elastic_pred))
print(np.sqrt(mean_squared_error(y_test,elastic_pred)))
```

```
8.34600759092681
2.3987645425349116
2.888945757698959
```

R square

In [202…

```
from sklearn.metrics import r2_score
elastic_score=r2_score(y_test,elastic_pred)
print(elastic_score)
```

```
0.4481026043251144
```

Adjusted R square

In [204…

```
1-(1-elastic_score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[204…

```
0.36926011922870217
```

Loading [MathJax]/extensions/Safe.js