Linear Regression on Algerian Fire Forest Dataset

```
from IPython import display
display.Image("D:\\220818074357-02-algeria-fires-0817.jpg",
width=1000)
```

Life Cycle of Machine Learning Project

- Understanding the Problem Statement
- Data Collection
- Exploratory data analysis
- Data Cleaning
- Data Pre-Processing
- Model Training
- Choose best model

Dataset Link : https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++

Data Set Information:

- The dataset includes 244 instances that regroup a data of two regions of Algeria,namely the Bejaia region located in the northeast of Algeria and the Sidi Bel-abbes region located in the northwest of Algeria.
- 122 instances for each region.
- The period from June 2012 to September 2012.
- The dataset includes 11 attribues and 1 output attribue (class)

- The 244 instances have been classified into â€˜fireâ€™ (138 classes) and â€˜not fireâ€™ (106 classes) classes.

Attribute Information:

-
    a. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations
-
    a. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42
-
    a. RH : Relative Humidity in %: 21 to 90
-
    a. Ws :Wind speed in km/h: 6 to 29
-
    a. Rain: total day in mm: 0 to 16.8 FWI Components
-
    a. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5
-
    a. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9
-
    a. Drought Code (DC) index from the FWI system: 7 to 220.4
-
    a. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5
-
    a. Buildup Index (BUI) index from the FWI system: 1.1 to 68
-
    a. Fire Weather Index (FWI) Index: 0 to 31.1
-
    a. Classes: two classes, namely Fire and not Fire

Problem Statement

- Predict the tempreture using Linear Regression Algorithm

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import cufflinks as cf
cf.go_offline
import plotly.express as px
%matplotlib inline
```

```python
import warnings
warnings.filterwarnings('ignore')

# Importing dataset
df=pd.read_csv("D:\Algerian_forest_fires_dataset_UPDATE (1).csv",
header=1)
df.head()
```

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

     Classes
0  not fire
1  not fire
2  not fire
3  not fire
4  not fire
```

```python
df.shape
```

```
(246, 14)
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          246 non-null    object
 1   month        245 non-null    object
 2   year         245 non-null    object
 3   Temperature  245 non-null    object
 4    RH          245 non-null    object
 5    Ws          245 non-null    object
 6   Rain         245 non-null    object
 7   FFMC         245 non-null    object
 8   DMC          245 non-null    object
 9   DC           245 non-null    object
 10  ISI          245 non-null    object
 11  BUI          245 non-null    object
```

```
 12  FWI          245 non-null    object
 13  Classes      244 non-null    object
dtypes: object(14)
memory usage: 27.0+ KB
```

```
df.columns
```

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

```
# We hav few extra space in column name
for feature in df.columns:
    df.rename(columns= {feature : feature.strip()}, inplace=True )
df.columns
```

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

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

```
day            0
month          1
year           1
Temperature    1
RH             1
Ws             1
Rain           1
FFMC           1
DMC            1
DC             1
ISI            1
BUI            1
FWI            1
Classes        2
dtype: int64
```

```
#Deleting uneccesary rows
df.drop([122,123], axis=0, inplace=True)
```

```
# There is till one missing value present in dataset
df.isnull().sum()
```

```
day            0
month          0
year           0
Temperature    0
RH             0
Ws             0
```

```
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        1
dtype: int64
```

```
#Checking which index have NaN value
df[df['Classes'].isna()]
```

```
    day month  year Temperature  RH  Ws Rain  FFMC   DMC      DC   ISI
BUI  \
167  14    07  2012          37  37  18  0.2  88.9  12.9  14.6 9  12.5
10.4

         FWI Classes
167  fire        NaN
```

Observation

- At index 167 few column values are shifted towards left

```
# WE need range that enitities correctly
df.at[167, 'DC']=14.6
df.at[167, 'ISI']=9
df.at[167, 'BUI']=12.5
df.at[167, 'FWI']=10.4
df.at[167, 'Classes']='fire'
```

```
df.head()
```

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

       Classes
0  not fire
1  not fire
2  not fire
3  not fire
4  not fire
```

```
df.isna().sum()
```

```
day            0
month          0
year           0
Temperature    0
RH             0
Ws             0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        0
dtype: int64
```

```
#Adding new column in dataset because we have two region present in 
dataset
# From index 0-122 we have Bajaija region
# From index 122 onwards we have Sidi-Bel Abbes region
df.loc[:122, 'Region']='Bajaia'
df.loc[122:, 'Region']='Sidi-Bel Abbes'
```

Observtaion

- We added region columns for better understanding

```
df.dtypes
```

```
day            object
month          object
year           object
Temperature    object
RH             object
Ws             object
Rain           object
FFMC           object
DMC            object
DC             object
ISI            object
BUI            object
FWI            object
Classes        object
Region         object
dtype: object
```

```
#here we combined day, month and year column together
df['date']=(df['day']+('/')+df['month']+('/')+df['year'])
df.date
```

```
0      01/06/2012
1      02/06/2012
2      03/06/2012
3      04/06/2012
4      05/06/2012
          ...
241    26/09/2012
242    27/09/2012
243    28/09/2012
244    29/09/2012
245    30/09/2012
Name: date, Length: 244, dtype: object
```

```
#Dropping day, month, year column
df.drop(['day', 'month', 'year'], axis=1, inplace=True)

df.head()
```

```
  Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  FWI
Classes  \
0          29  57  18     0  65.7  3.4   7.6  1.3  3.4  0.5  not fire

1          29  61  13   1.3  64.4  4.1   7.6    1  3.9  0.4  not fire

2          26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1  not fire

3          25  89  13   2.5  28.6  1.3   6.9    0  1.7    0  not fire

4          27  77  16     0  64.8    3  14.2  1.2  3.9  0.5  not fire


   Region        date
0  Bajaia  01/06/2012
1  Bajaia  02/06/2012
2  Bajaia  03/06/2012
3  Bajaia  04/06/2012
4  Bajaia  05/06/2012
```

```
df['Classes'].unique()
```

```
array(['not fire   ', 'fire   ', 'fire', 'fire ', 'not fire', 'not 
fire ',
       'not fire     ', 'not fire    '], dtype=object)
```

Observation

- Here we have few uneccesary spaces between classes entities

```
#striping uneccessary spaces
df['Classes']= [i.strip() for i in df['Classes']]

df.Classes.unique()
```

```
array(['not fire', 'fire'], dtype=object)
```

```
# Changing important columns datatypes
df.dtypes
```

```
Temperature    object
RH             object
Ws             object
Rain           object
FFMC           object
DMC            object
DC             object
ISI            object
BUI            object
FWI            object
Classes        object
Region         object
date           object
dtype: object
```

Observation

- Few numeric columns have dtype as object

```
df.columns
```

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

```
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['DC']=df['DC'].astype(float)
df['ISI']=df['ISI'].astype(float)
df['BUI']=df['BUI'].astype(float)
df['FWI']=df['FWI'].astype(float)
df['Temperature']=df['Temperature'].astype(int)
```

```
df.dtypes
```

```
Temperature      int32
RH               int32
Ws               int32
Rain           float64
FFMC           float64
DMC            float64
DC             float64
ISI            float64
BUI            float64
```

```
FWI             float64
Classes          object
Region           object
date             object
dtype: object
```

```
df.head()
```

```
   Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  FWI   Classes
\
0           29  57  18   0.0  65.7  3.4   7.6  1.3  3.4  0.5  not fire

1           29  61  13   1.3  64.4  4.1   7.6  1.0  3.9  0.4  not fire

2           26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1  not fire

3           25  89  13   2.5  28.6  1.3   6.9  0.0  1.7  0.0  not fire

4           27  77  16   0.0  64.8  3.0  14.2  1.2  3.9  0.5  not fire


   Region        date
0  Bajaia  01/06/2012
1  Bajaia  02/06/2012
2  Bajaia  03/06/2012
3  Bajaia  04/06/2012
4  Bajaia  05/06/2012
```

```
df.shape
```

```
(244, 13)
```

```
#Checking duplicate values
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

```
       Temperature          RH          Ws        Rain        FFMC  \
count   244.000000  244.000000  244.000000  244.000000  244.000000
mean     32.172131   61.938525   15.504098    0.760656   77.887705
std       3.633843   14.884200    2.810178    1.999406   14.337571
min      22.000000   21.000000    6.000000    0.000000   28.600000
25%      30.000000   52.000000   14.000000    0.000000   72.075000
50%      32.000000   63.000000   15.000000    0.000000   83.500000
75%      35.000000   73.250000   17.000000    0.500000   88.300000
max      42.000000   90.000000   29.000000   16.800000   96.000000

              DMC          DC         ISI         BUI         FWI
count  244.000000  244.000000  244.000000  244.000000  244.000000
```

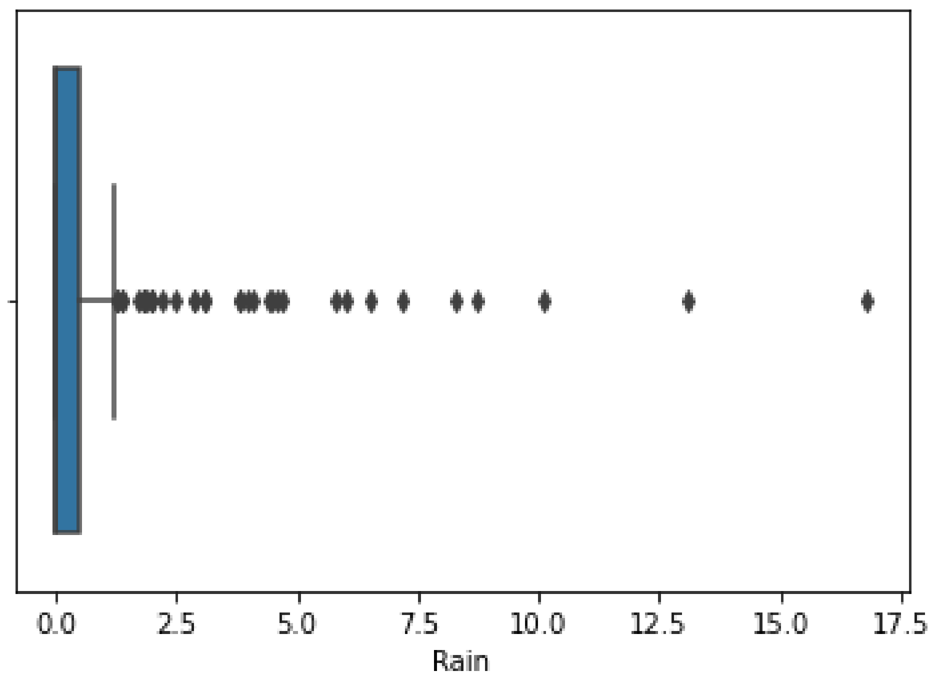
```
mean     14.673361   49.288115    4.759836   16.673361    7.049180
std      12.368039   47.619662    4.154628   14.201648    7.428366
min       0.700000    6.900000    0.000000    1.100000    0.000000
25%       5.800000   13.275000    1.400000    6.000000    0.700000
50%      11.300000   33.100000    3.500000   12.450000    4.450000
75%      20.750000   68.150000    7.300000   22.525000   11.375000
max      65.900000  220.400000   19.000000   68.000000   31.100000
```

Observation

- As we can see min, max, 24th, 50th and 75th percentile surely we have some outliers in few features

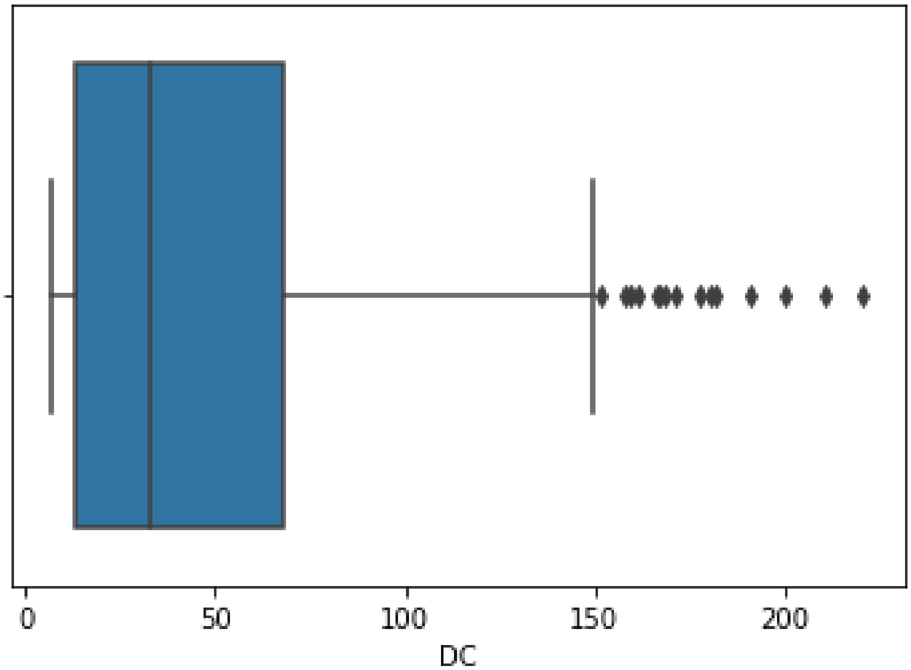
```
sns.boxplot(x=df['Rain'])
```

```
<AxesSubplot:xlabel='Rain'>
```




```
sns.boxplot(x=df['DC'])
```

```
<AxesSubplot:xlabel='DC'>
```

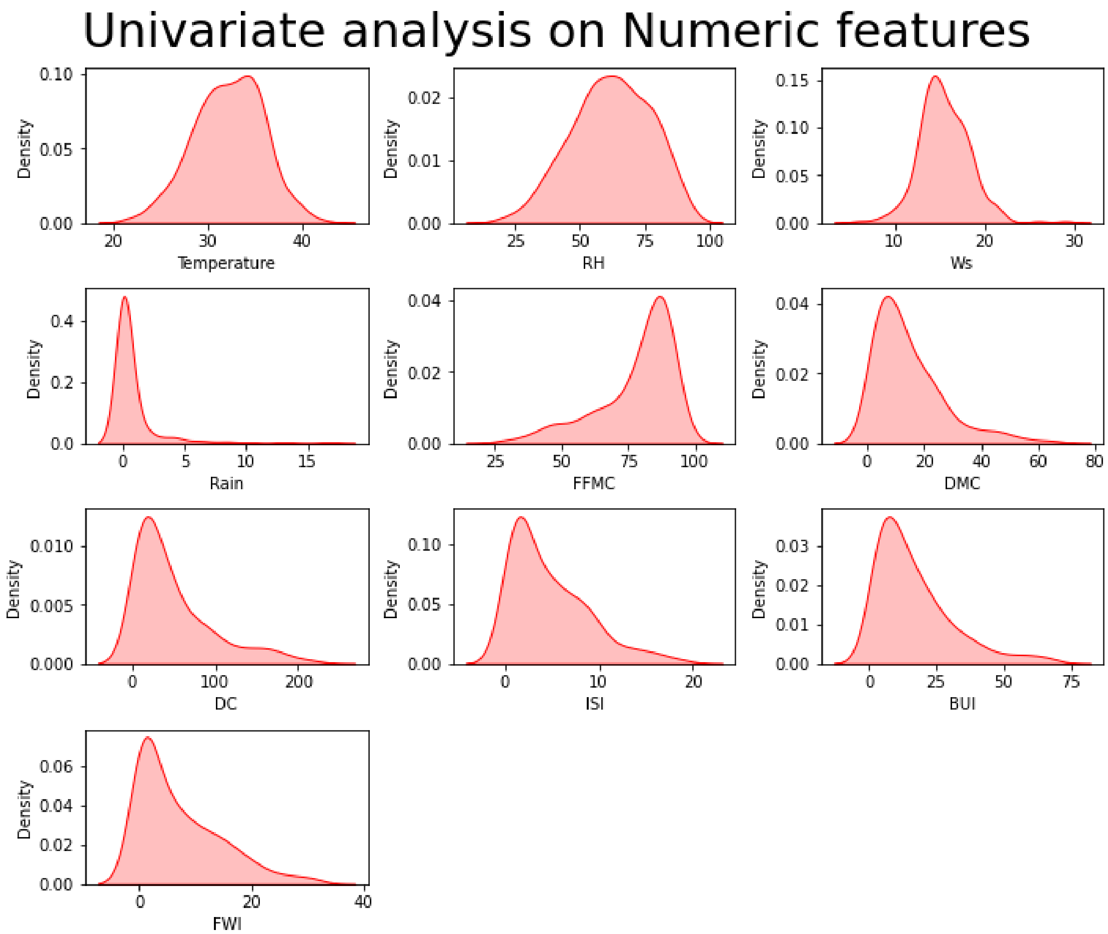


EDA and Feature Engineearning

```
Num_col=[feature for feature in df.columns if df[feature].dtypes!='O']
Num_col
```

```
['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
'FWI']
```

```
plt.figure(figsize= (10,10))
plt.suptitle('Univariate analysis on Numeric features', fontsize=30)

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



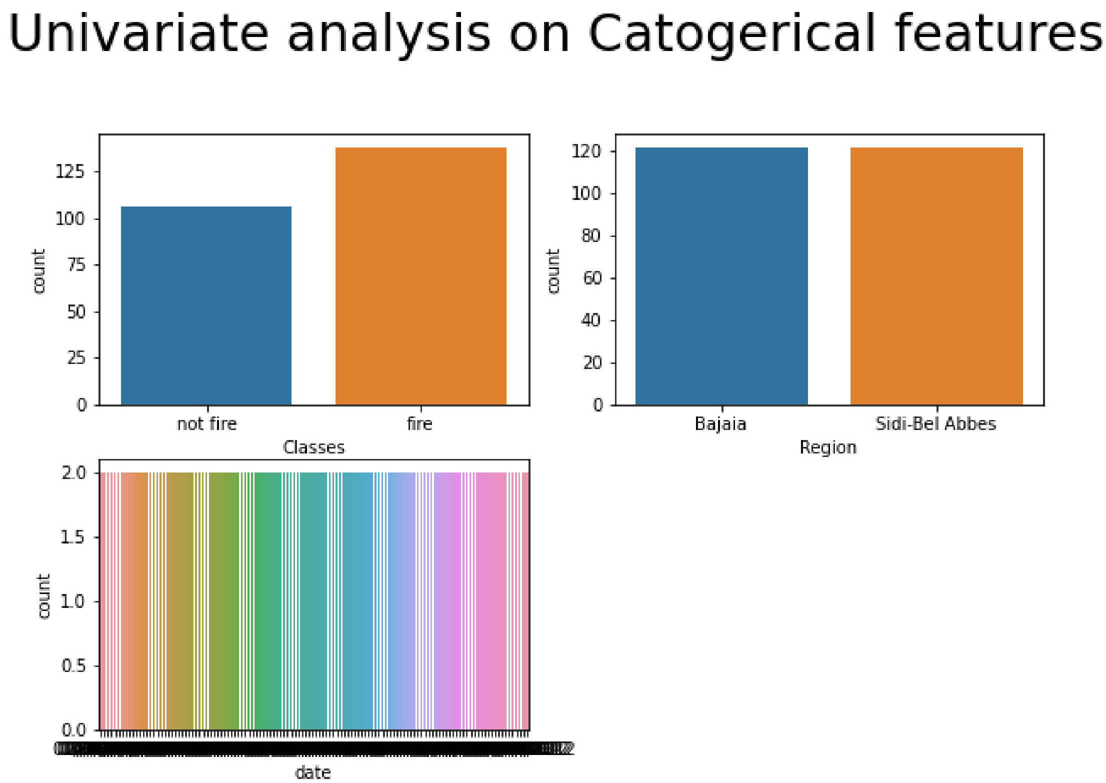
Observation

- Feature Tempreture, RH, WH are normally distrubuted
- Feature Rain, DC, ISI, BUI, FWI are right skewed and have outliers
- Feature FFMC, DMC are left skewed and have outliers

```
Cat_col=[feature for feature in df.columns if df[feature].dtypes=='O']
Cat_col
```

```
['Classes', 'Region', 'date']
```

```
plt.figure(figsize= (10,10))
plt.suptitle('Univariate analysis on Catogerical features',
fontsize=30)

for i in range(len(Cat_col)):
    plt.subplot(3, 2, i+1)
    sns.countplot(x=df[Cat_col[i]])
    plt.xlabel(Cat_col[i])
```



Observation

- In Classes feature there are more count of fire compared to no fire

```
df.corr()

             Temperature        RH        Ws      Rain      FFMC
DMC  \
Temperature     1.000000 -0.654443 -0.278132 -0.326786  0.677491
0.483105
RH             -0.654443  1.000000  0.236084  0.222968 -0.645658 -
0.405133
Ws             -0.278132  0.236084  1.000000  0.170169 -0.163255 -
0.001246
Rain           -0.326786  0.222968  0.170169  1.000000 -0.544045 -
0.288548
FFMC            0.677491 -0.645658 -0.163255 -0.544045  1.000000
0.602391
DMC             0.483105 -0.405133 -0.001246 -0.288548  0.602391
1.000000
DC              0.370498 -0.220330  0.076245 -0.296804  0.503910
0.875358
ISI             0.605971 -0.688268  0.012245 -0.347862  0.740751
0.678355
BUI             0.456415 -0.349685  0.030303 -0.299409  0.590251
0.982206
```

```
FWI              0.566839 -0.580457  0.033957 -0.324755  0.691430
0.875191

                   DC       ISI       BUI       FWI
Temperature  0.370498  0.605971  0.456415  0.566839
RH          -0.220330 -0.688268 -0.349685 -0.580457
Ws           0.076245  0.012245  0.030303  0.033957
Rain        -0.296804 -0.347862 -0.299409 -0.324755
FFMC         0.503910  0.740751  0.590251  0.691430
DMC          0.875358  0.678355  0.982206  0.875191
DC           1.000000  0.503919  0.941672  0.737041
ISI          0.503919  1.000000  0.641351  0.922422
BUI          0.941672  0.641351  1.000000  0.856912
FWI          0.737041  0.922422  0.856912  1.000000
```

```python
sns.set(rc={'figure.figsize':(10,10)})
sns.heatmap(df.corr(), annot=True)
```

```
<AxesSubplot:>
```

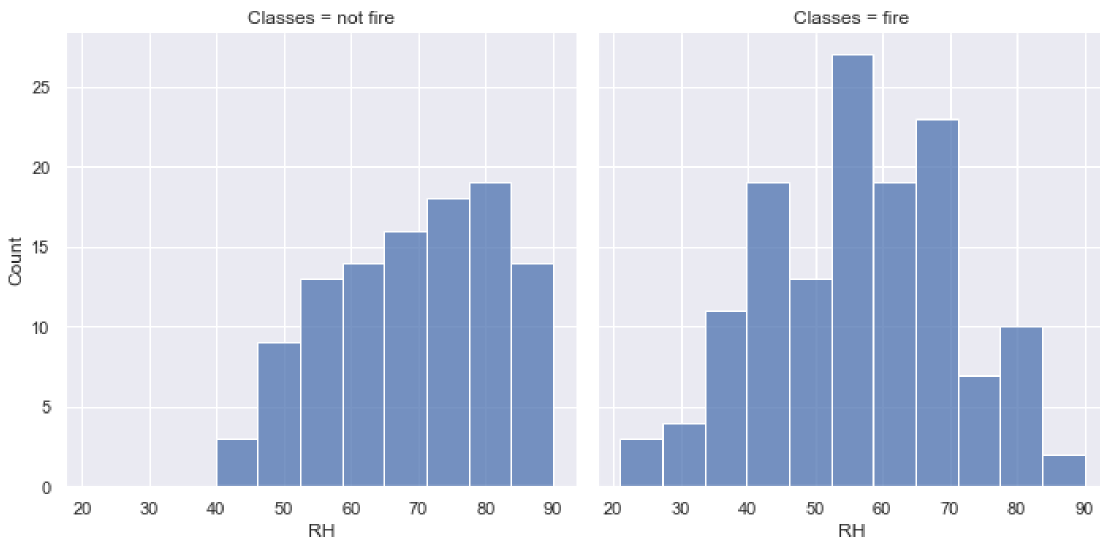
Bivariate Data Analysis

```
df.columns
```

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

```
sns.displot(df, x='RH', col="Classes")
```

```
<seaborn.axisgrid.FacetGrid at 0x213f6f2b610>
```

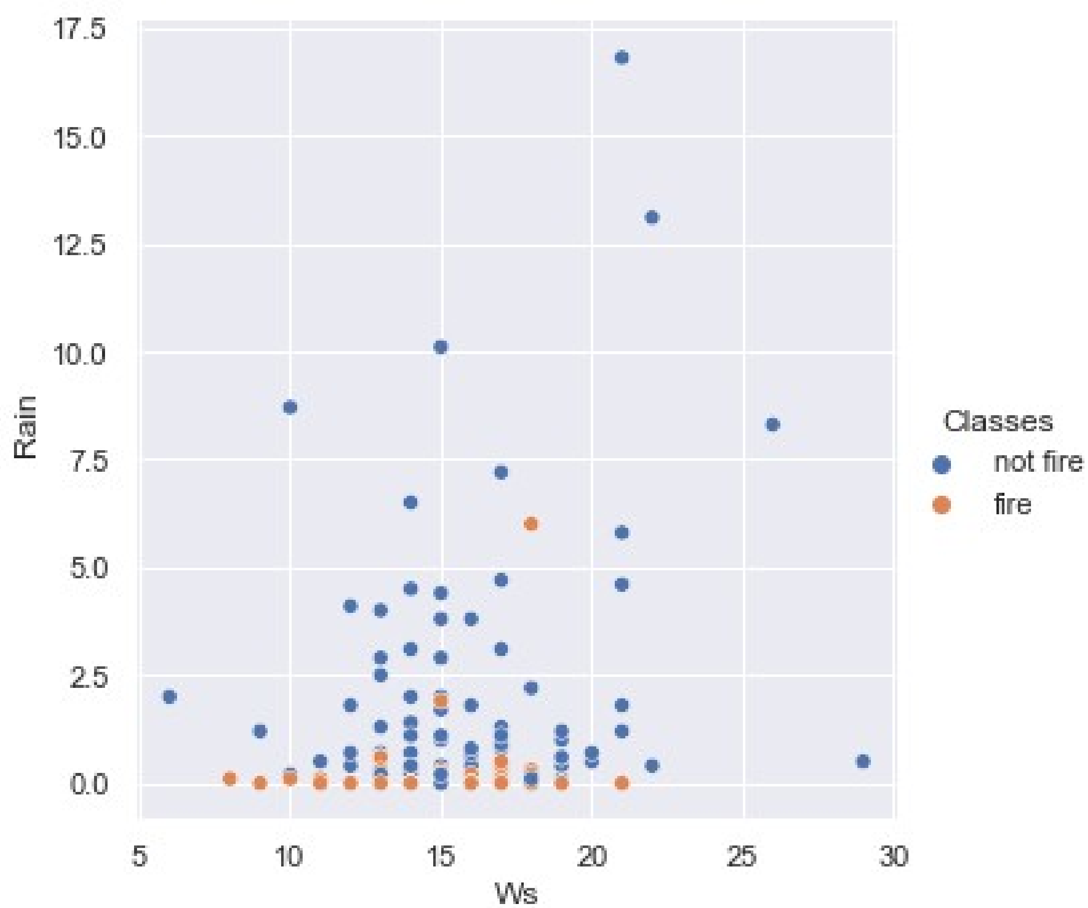


Observation

- RH(relative humidity) between range 40% - 70% rate of catching fire is high

```
sns.relplot(data=df, x="Ws", y="Rain", hue="Classes")
```

```
<seaborn.axisgrid.FacetGrid at 0x213f73d4a60>
```

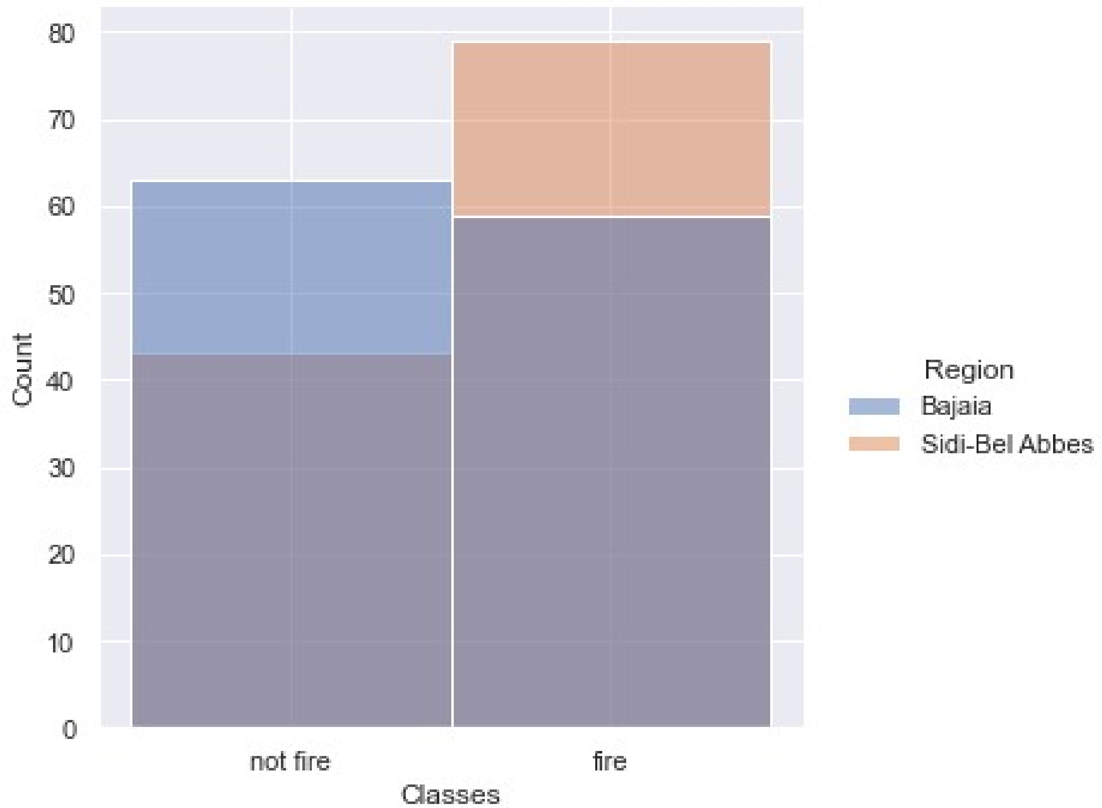


Observation

- Whenever there is less Ws(Wind speed) the chances of rain are less , so chances of catching fire is high (most of fire shown in this phase)
- Whenever there is high Ws(Wind speed) the chances of rain are high , so chances of catching fire is less

```
sns.displot(df, x='Classes', hue='Region')
```

```
<seaborn.axisgrid.FacetGrid at 0x213f6d697f0>
```

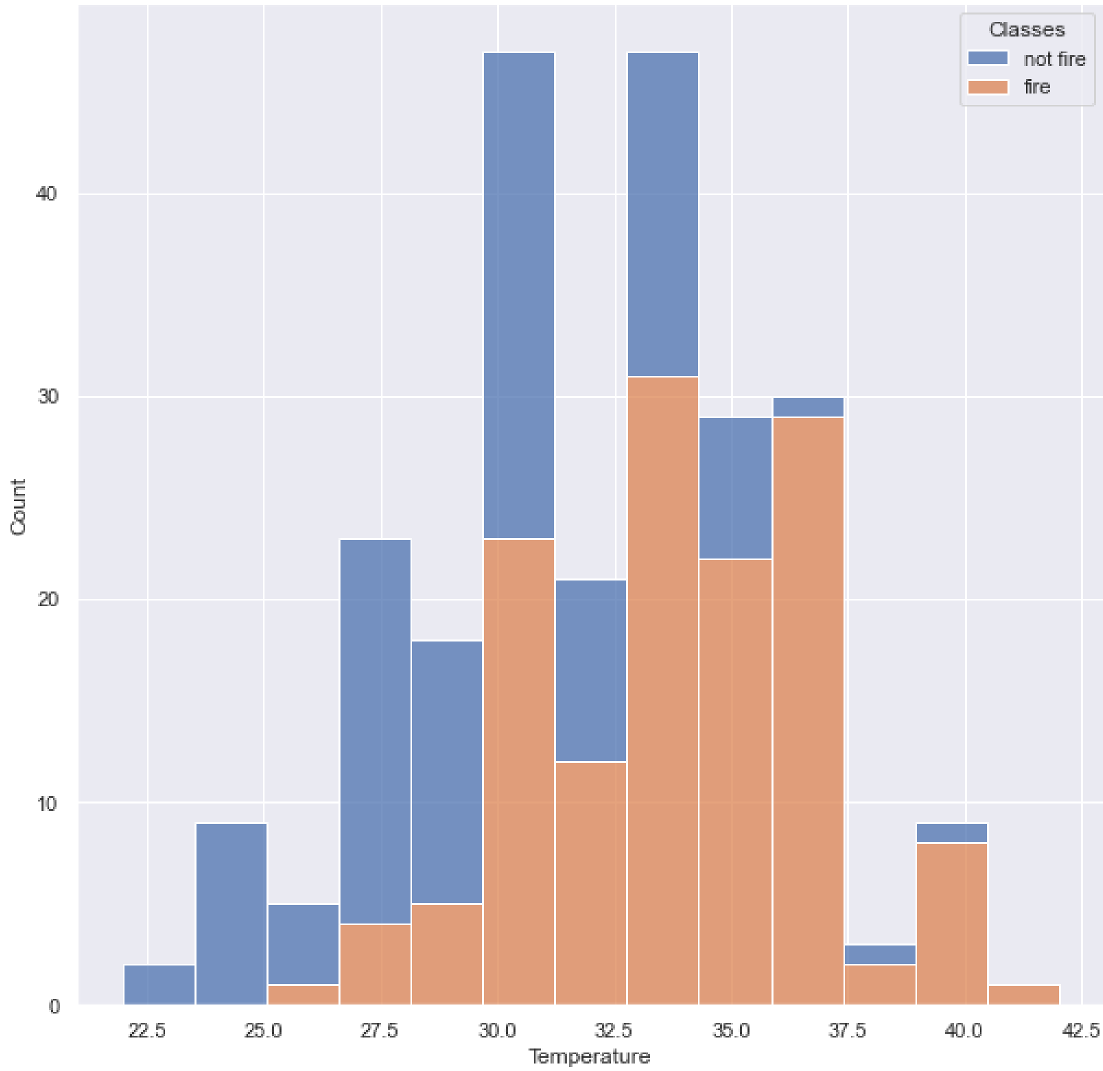


Observation

- As compared to Sidi-Bel Abbes region Bajaija region has more fire cases

```
sns.histplot(data=df, x="Temperature", hue="Classes",
multiple="stack")
```

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

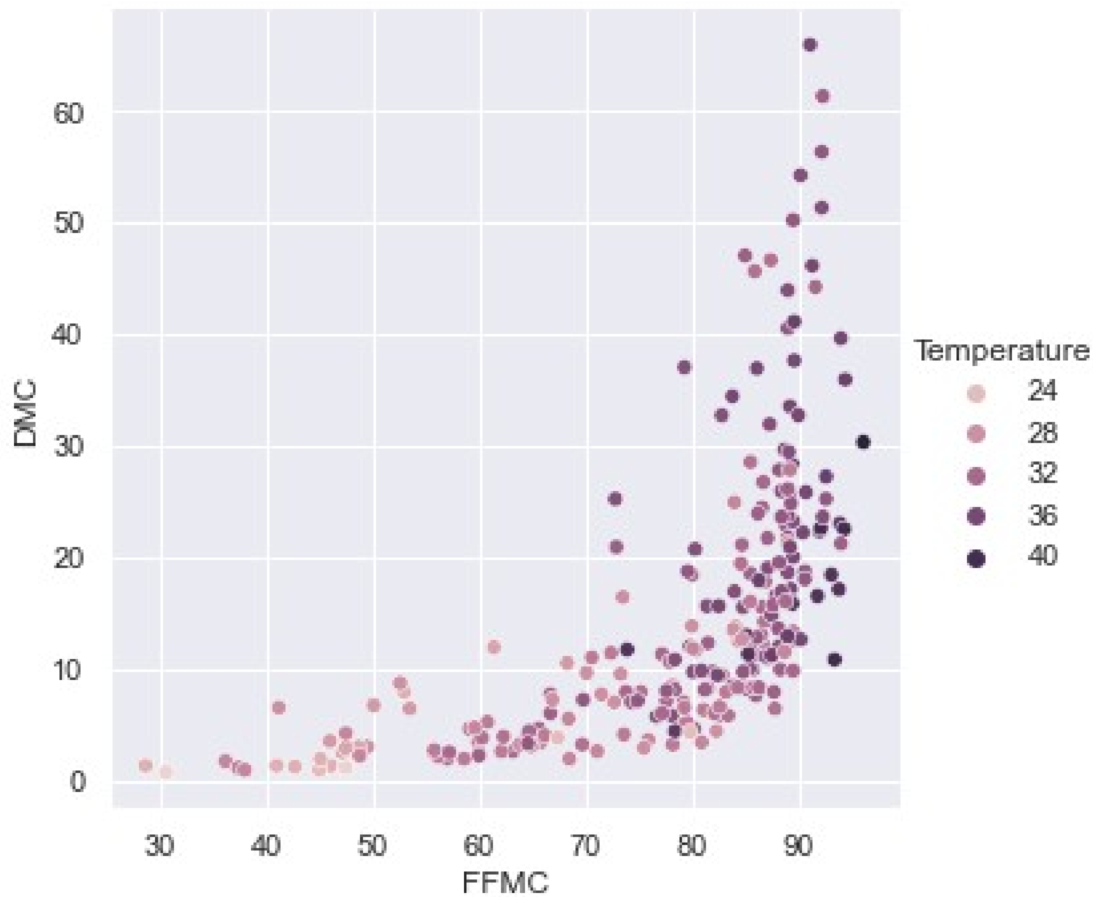


Observation

- The relationship between Temperature and Classes are directly propotional

```
sns.relplot(data=df, x="FFMC", y="DMC", hue="Temperature")
```

```
<seaborn.axisgrid.FacetGrid at 0x213f842a310>
```



Observation

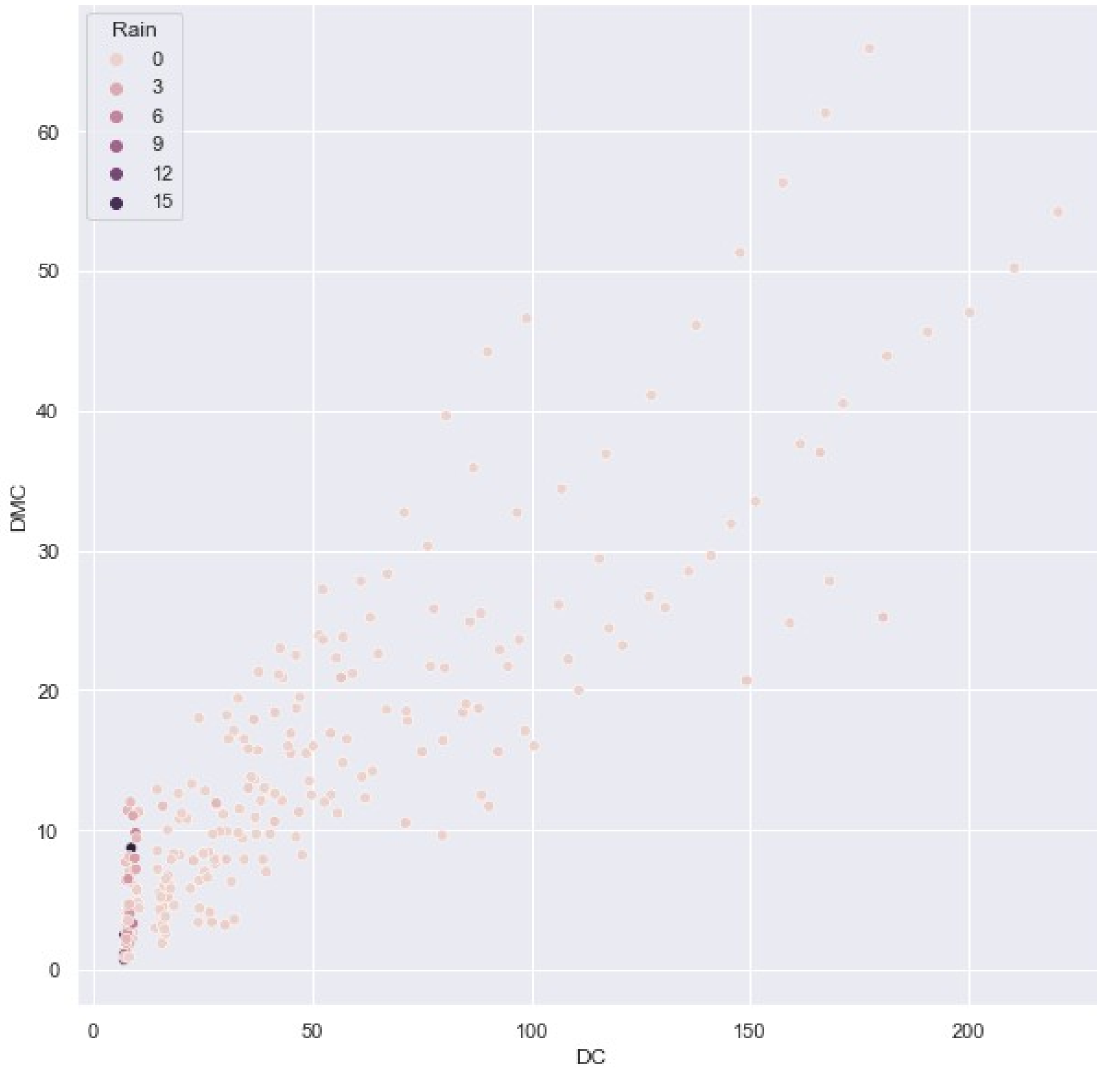
- Feature FFMC(Fine Fuel Moisture Code), DMC(Duff Moisture Code) increses exponentially with respect to Temperature

```
df.columns
```

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

```
sns.scatterplot(data=df, x='DC',y='DMC', hue='Rain')
```

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

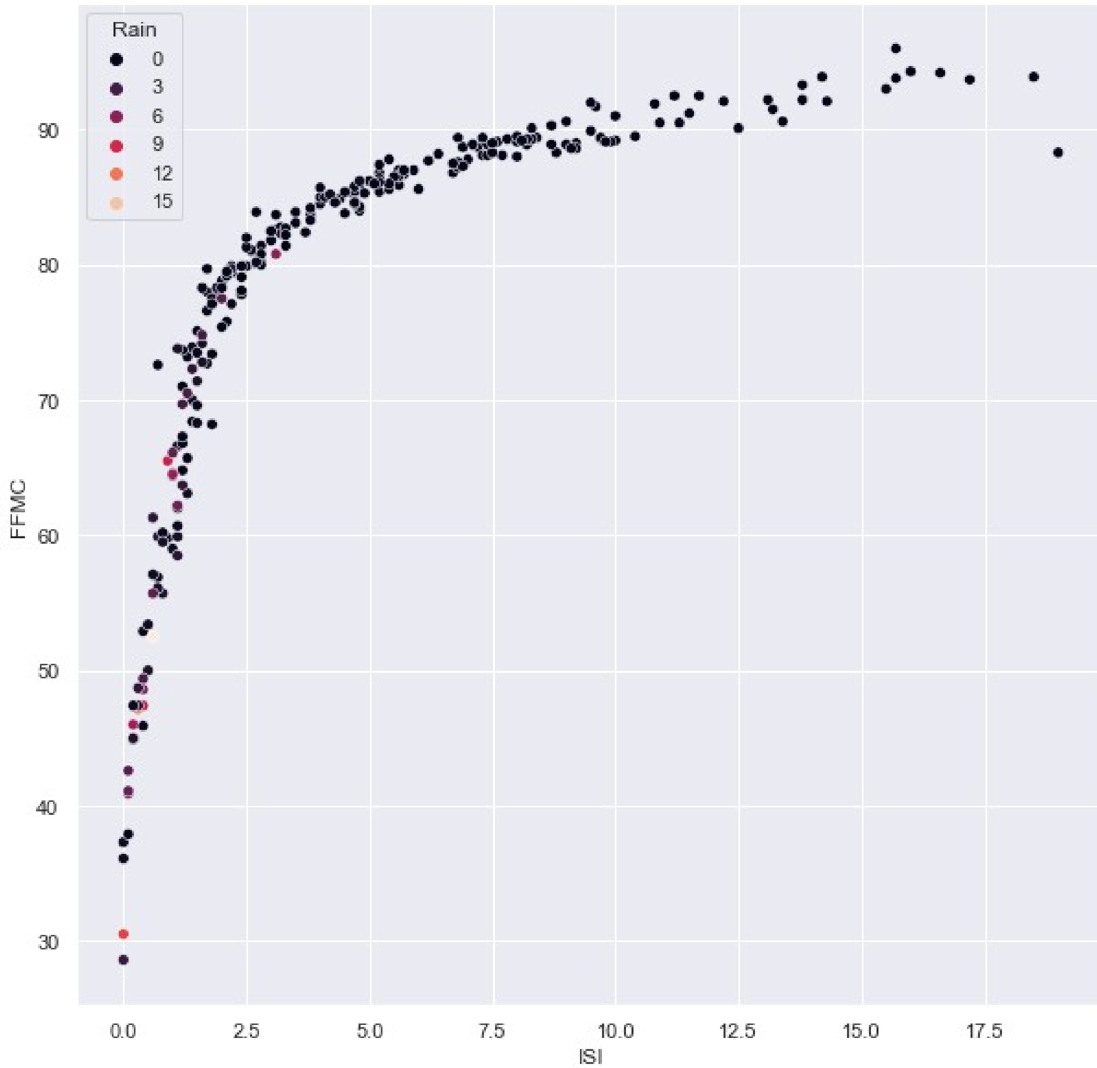


Observation

- As we can see when DC(Drought Code) increses along with DMC(Duff Moisture Code) chances of Rain are very less

```
sns.scatterplot(data=df,x='ISI', y='FFMC', hue='Rain',
palette='rocket')
```

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



Final Observation

- As compared to Sidi-Bel Abbes region Bajaia region have more fire cacthed cases
- Increase in feature Tempreture , Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC) showns more fire cases
- While feature like Increase in Ws (Wind speed), Rain showns less chances of fire catched

```
df.columns
```

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

```
df.drop(columns='date', inplace=True)
```

```
df.head(1)
```

```
   Temperature  RH  Ws  Rain  FFMC  DMC   DC  ISI  BUI  FWI   Classes
Region
0           29  57  18   0.0  65.7  3.4  7.6  1.3  3.4  0.5  not fire
Bajaia
```

```
df.Classes.unique()
```

```
array(['not fire', 'fire'], dtype=object)
```

```
df["Classes"]=df["Classes"].map({'not fire':0, 'fire':1})
```

```
df.Region.unique()
```

```
array(['Bajaia', 'Sidi-Bel Abbes'], dtype=object)
```

```
df['Region']=df["Region"].map({'Bajaia':0, 'Sidi-Bel Abbes':1})
```

```
df
```

```
     Temperature  RH  Ws  Rain  FFMC   DMC    DC  ISI   BUI  FWI
Classes  \
0             29  57  18   0.0  65.7   3.4   7.6  1.3   3.4  0.5
0
1             29  61  13   1.3  64.4   4.1   7.6  1.0   3.9  0.4
0
2             26  82  22  13.1  47.1   2.5   7.1  0.3   2.7  0.1
0
3             25  89  13   2.5  28.6   1.3   6.9  0.0   1.7  0.0
0
4             27  77  16   0.0  64.8   3.0  14.2  1.2   3.9  0.5
0
..           ...  ..  ..   ...   ...   ...   ...  ...   ...  ...
...
241           30  65  14   0.0  85.4  16.0  44.5  4.5  16.9  6.5
1
242           28  87  15   4.4  41.1   6.5   8.0  0.1   6.2  0.0
0
243           27  87  29   0.5  45.9   3.5   7.9  0.4   3.4  0.2
0
244           24  54  18   0.1  79.7   4.3  15.2  1.7   5.1  0.7
0
245           24  64  15   0.2  67.3   3.8  16.5  1.2   4.8  0.5
0

     Region
0         0
1         0
2         0
3         0
4         0
..      ...
241       1
```

```
242       1
243       1
244       1
245       1

[244 rows x 12 columns]
```

```
#### Independant and dependant feature
X=df.iloc[:, 1:]
```

```
X
```

```
     RH  Ws  Rain  FFMC   DMC    DC  ISI   BUI  FWI  Classes  Region
0    57  18   0.0  65.7   3.4   7.6  1.3   3.4  0.5        0       0
1    61  13   1.3  64.4   4.1   7.6  1.0   3.9  0.4        0       0
2    82  22  13.1  47.1   2.5   7.1  0.3   2.7  0.1        0       0
3    89  13   2.5  28.6   1.3   6.9  0.0   1.7  0.0        0       0
4    77  16   0.0  64.8   3.0  14.2  1.2   3.9  0.5        0       0
..   ..  ..   ...   ...   ...   ...  ...   ...  ...      ...     ...
241  65  14   0.0  85.4  16.0  44.5  4.5  16.9  6.5        1       1
242  87  15   4.4  41.1   6.5   8.0  0.1   6.2  0.0        0       1
243  87  29   0.5  45.9   3.5   7.9  0.4   3.4  0.2        0       1
244  54  18   0.1  79.7   4.3  15.2  1.7   5.1  0.7        0       1
245  64  15   0.2  67.3   3.8  16.5  1.2   4.8  0.5        0       1

[244 rows x 11 columns]
```

```
y=df.iloc[:, :1]
```

```
y
```

```
     Temperature
0             29
1             29
2             26
3             25
4             27
..           ...
241           30
242           28
243           27
244           24
245           24

[244 rows x 1 columns]
```

```
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=30)
```

```
X_train
```

```
     RH  Ws  Rain  FFMC   DMC     DC   ISI   BUI   FWI  Classes  
Region
231  26  13   0.0  93.9  21.2   59.2  14.2  22.4  19.3        1  
1
159  42  15   0.3  84.7  15.5   45.1   4.3  16.7   6.3        1  
1
227  72  14   0.0  84.2   8.3   25.2   3.8   9.1   3.9        1  
1
87   82  21   0.0  84.9  47.0  200.2   4.4  59.3  13.2        1  
0
6    54  13   0.0  88.2   9.9   30.5   6.4  10.9   7.2        1  
0
..   ..  ..   ...   ...   ...    ...   ...   ...   ...      ...     ..
.
142  67  14   4.5  64.6   4.4    8.2   1.0   4.2   0.4        0  
1
45   76  21   0.0  72.6   7.0   25.5   0.7   8.3   0.4        0  
0
175  48  18   0.0  91.5  44.2   90.1  13.2  44.0  25.4        1  
1
167  37  18   0.2  88.9  12.9   14.6   9.0  12.5  10.4        1  
1
37   68  19   0.0  85.6  12.5   49.8   6.0  15.4   8.0        1  
0

[163 rows x 11 columns]
```

y_train

```
     Temperature
231           33
159           35
227           31
87            33
6             33
..           ...
142           32
45            28
175           32
167           37
37            33

[163 rows x 1 columns]
```

X_test

```
     RH  Ws  Rain  FFMC   DMC    DC  ISI   BUI   FWI  Classes  Region
51   79  18   0.1  73.4  16.4  79.9  1.8  21.7   2.8        0       0
224  80  15   0.0  83.1   7.9  34.5  3.5  10.0   3.7        1       1
41   75  13   0.1  75.1   7.9  27.7  1.5   9.2   0.9        0       0
192  56  11   0.0  87.4  11.2  20.2  5.2  11.0   5.9        1       1
```

```
67   69  16   0.0  86.5  15.5  48.6  5.5  17.2   8.0        1       0
..   ..  ..   ...   ...   ...   ...  ...   ...   ...      ...     ...
245  64  15   0.2  67.3   3.8  16.5  1.2   4.8   0.5        0       1
172  58  16   0.0  88.1  27.8  61.1  7.3  27.7  13.0        1       1
183  56  16   0.0  88.9  23.8  57.1  8.2  23.8  13.2        1       1
199  46  13   0.3  83.9  16.9  54.2  3.5  19.0   5.5        1       1
125  73  13   4.0  55.7   2.7   7.8  0.6   2.9   0.2        0       1

[81 rows x 11 columns]
```

```
y_test
```

```
     Temperature
51            28
224           30
41            31
192           37
67            32
..           ...
245           24
172           34
183           36
199           35
125           30

[81 rows x 1 columns]
```

```
#### Model Training
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()

X_train=scaler.fit_transform(X_train)

X_train
```

```
array([[-2.27212014, -0.90068636, -0.36944043, ...,  1.66828808,
         0.90061707,  1.03116009],
       [-1.24886175, -0.22413393, -0.22001415, ..., -0.08645353,
         0.90061707,  1.03116009],
       [ 0.66974773, -0.56241015, -0.36944043, ..., -0.41040582,
         0.90061707,  1.03116009],
       ...,
       [-0.86513986,  0.79069471, -0.36944043, ...,  2.49166684,
         0.90061707,  1.03116009],
       [-1.56863   ,  0.79069471, -0.26982291, ...,  0.46696498,
         0.90061707,  1.03116009],
       [ 0.41393313,  1.12897092, -0.36944043, ...,  0.14301268,
         0.90061707, -0.96978152]])
```

```
X_test=scaler.transform(X_test)
```

```
X_test
```

```
array([[ 1.11742327,  0.79069471, -0.31963167, -0.28472323,
0.16888849,
         0.70985894, -0.6911055 ,  0.38668233, -0.55888396, -
1.11034982,
        -0.96978152],
       [ 1.18137692, -0.22413393, -0.36944043,  0.37189273, -
0.55043215,
        -0.27510311, -0.29616771, -0.45610838, -0.43740185,
0.90061707,
         1.03116009],
       [ 0.86160868, -0.90068636, -0.31963167, -0.16964621, -
0.55043215,
        -0.42263047, -0.76080041, -0.5137351 , -0.8153462 , -
1.11034982,
        -0.96978152],
       [-0.35351066, -1.57723878, -0.36944043,  0.6629699 , -
0.27116649,
        -0.58534446,  0.09877008, -0.38407499, -0.14044558,
0.90061707,
         1.03116009],
       [ 0.47788678,  0.11414228, -0.36944043,  0.60204677,
0.09272513,
         0.0307992 ,  0.16846498,  0.06253206,  0.14301268,
0.90061707,
        -0.96978152],
       [ 0.92556232,  0.45241849,  3.21679024, -2.13949408, -
1.10896348,
        -0.8608735 , -1.06281166, -1.04678222, -0.92333029, -
1.11034982,
        -0.96978152],
       [ 0.54184043,  0.45241849,  0.02902964, -0.33210789,
0.91359693,
         2.89022647, -0.71433714,  1.51760663, -0.36991179, -
1.11034982,
        -0.96978152],
       [-0.67327891,  0.45241849,  0.27807344, -0.42010796, -
0.56735734,
        -0.86304302, -0.76080041, -0.65059855, -0.82884421, -
1.11034982,
         1.03116009],
       [-1.37676905,  0.79069471, -0.36944043,  0.98112403,
3.54546423,
         2.39340641,  2.21284884,  3.10954465,  3.26105354,
0.90061707,
         1.03116009],
       [ 0.92556232, -0.56241015, -0.36944043,  0.23650799, -
0.53350696,
```

```
         -0.61788726, -0.50525242, -0.59297183, -0.63987203, -
1.11034982,
         -0.96978152],
        [ 0.47788678,  0.45241849,  1.97157126, -1.04287774, -
0.88893599,
         -0.8500259 , -0.85372694, -0.90271543, -0.88283626, -
1.11034982,
          1.03116009],
        [-0.54537161,  0.79069471, -0.36944043,  0.77127769,
0.60894348,
          0.71853702,  1.02803547,  0.68201925,  1.08787355,
0.90061707,
         -0.96978152],
        [ 0.79765503, -0.22413393,  0.17845592, -1.22564713, -
0.82123522,
         -0.84568686, -0.92342185, -0.84508872, -0.89633427, -
1.11034982,
          1.03116009],
        [ 0.22207218,  0.79069471, -0.36944043,  0.45312357, -
0.16115275,
          0.90077669,  0.00584354,  0.15617547,  0.0485266 ,
0.90061707,
         -0.96978152],
        [-1.56863   ,  0.11414228, -0.36944043,  0.98789326,
3.96859402,
          2.60384984,  1.93406922,  3.43369493,  3.15306944,
0.90061707,
          1.03116009],
        [ 0.09416488, -0.56241015, -0.22001415, -0.06810766, -
0.73660926,
         -0.8066355 , -0.71433714, -0.78025866, -0.82884421, -
1.11034982,
         -0.96978152],
        [-0.99304716, -0.56241015,  0.17845592,  0.04696937, -
0.53350696,
         -0.84351734, -0.66787387, -0.62178519, -0.77485216, -
1.11034982,
          1.03116009],
        [-0.60932526,  0.79069471, -0.36944043,  0.7915854 ,
0.13503811,
          1.16111908,  1.14419364,  0.47312241,  1.0338815 ,
0.90061707,
         -0.96978152],
        [ 1.69300612, -0.90068636, -0.26982291, -1.67241675, -
0.55043215,
         -0.18181376, -1.01634839, -0.42009169, -0.89633427, -
1.11034982,
         -0.96978152],
        [-0.80118621, -0.22413393, -0.36944043,  0.78481616,
0.87974654,
```

```
         2.42811873,  0.77248749,  1.39514986,  1.22285367,
0.90061707,
         1.03116009],
       [ 0.09416488, -0.56241015, -0.36944043,  0.63589296, -
0.29655428,
        -0.22086512,  0.19169662, -0.2760249 , -0.01896347,
0.90061707,
        -0.96978152],
       [-0.16164972,  0.45241849, -0.36944043,  0.71035456, -
0.20346572,
         0.12191903,  0.67956095,  0.13456545,  0.53445504,
0.90061707,
        -0.96978152],
       [-0.03374242, -0.90068636,  0.27807344, -0.89395453, -
0.87201079,
        -0.85870398, -0.87695858, -0.89551209, -0.88283626, -
1.11034982,
        -0.96978152],
       [ 1.69300612, -0.90068636, -0.36944043, -0.44718491, -
0.99894973,
        -0.66344718, -0.83049531, -0.90991877, -0.86933824, -
1.11034982,
        -0.96978152],
       [-0.22560336, -0.90068636, -0.26982291,  0.12820021,  0.3635282
,
         0.88559005, -0.6214106 ,  0.5811725 , -0.42390384, -
1.11034982,
         1.03116009],
       [ 0.34997948,  1.12897092, -0.36944043, -0.1493385 , -
0.97356194,
        -0.66995574, -0.64464223, -0.88830875, -0.82884421, -
1.11034982,
         1.03116009],
       [-0.41746431, -1.57723878, -0.36944043,  0.69004685,
0.17735109,
         0.23256455,  0.14523335,  0.20659885,  0.18350672,
0.90061707,
        -0.96978152],
       [ 0.15811853,  0.45241849, -0.36944043,  0.64943143,
1.48059085,
         2.13740306,  0.47047623,  1.79133353,  1.18235964,
0.90061707,
        -0.96978152],
       [ 0.60579408, -0.22413393, -0.36944043,  0.56820059,
1.90372064,
         1.51692035,  0.07553844,  1.79853687,  0.7099292 ,
0.90061707,
         1.03116009],
       [ 0.28602583, -1.57723878, -0.36944043,  0.54789288, -
0.51658177,
```

```
        -0.48337703, -0.18000954, -0.52814178, -0.3834098 ,
0.90061707,
        -0.96978152],
       [ 1.05346962, -0.56241015, -0.36944043,  0.22973876, -
0.68583369,
        -0.33801919, -0.50525242, -0.57136182, -0.63987203,
0.90061707,
        -0.96978152],
       [-0.73723256,  0.11414228, -0.36944043,  0.83220082,
1.54829161,
         0.51677166,  1.09773037,  1.17184634,  1.39832783,
0.90061707,
         1.03116009],
       [-0.80118621,  1.12897092, -0.36944043,  0.74420074, -0.2457787
,
        -0.29896783,  1.00480383, -0.28322824,  0.45346697,
0.90061707,
         1.03116009],
       [ 0.86160868,  0.11414228, -0.36944043,  0.3109696 , -
0.84662301,
        -0.49639415, -0.34263098, -0.74424196, -0.599378  ,
0.90061707,
         1.03116009],
       [-0.22560336,  0.45241849, -0.36944043,  0.76450845,
0.58355569,
        -0.20350896,  0.91187729,  0.35066564,  0.80441529,
0.90061707,
         1.03116009],
       [ 0.73370138,  0.45241849,  0.0788384 , -1.05641621, -
0.99894973,
        -0.84134782, -0.85372694, -0.96034215, -0.88283626, -
1.11034982,
        -0.96978152],
       [ 1.82091342, -0.22413393, -0.36944043, -0.73149285, -
0.60967032,
        -0.70466806, -0.83049531, -0.66500523, -0.85584023, -
1.11034982,
         1.03116009],
       [ 0.98951597, -0.22413393,  0.12864716, -1.45580117, -
1.04126271,
        -0.84134782, -0.94665348, -0.98915551, -0.90983228, -
1.11034982,
        -0.96978152],
       [-0.16164972,  1.12897092, -0.36944043,  0.77804693,
1.13362442,
         2.62554504,  1.16742528,  1.65447008,  1.68178609,
0.90061707,
         1.03116009],
       [ 0.60579408, -1.23896257, -0.02077911, -1.38810881, -
1.00741233,
```

```
        -0.84568686, -0.96988512, -0.97474883, -0.90983228, -
1.11034982,
         1.03116009],
       [-0.09769607, -0.22413393, -0.36944043,  0.76450845,
2.49610234,
         2.90975215,  0.79571912,  2.76378436,  1.80326821,
0.90061707,
        -0.96978152],
       [-1.24886175,  1.80552335, -0.36944043,  0.87958548,
0.32121522,
        -0.36188391,  2.00376413,  0.12015877,  1.31733976,
0.90061707,
         1.03116009],
       [-1.3128154 , -2.59206742, -0.31963167,  0.42604662,
0.88820914,
         0.84219965, -0.48202079,  0.90532278, -0.18093961,
0.90061707,
         1.03116009],
       [-0.03374242,  0.79069471, -0.22001415,  0.17558486, -
0.22885351,
         0.93765853, -0.45878915,  0.09134542, -0.36991179,
0.90061707,
        -0.96978152],
       [ 0.60579408, -0.56241015,  2.86812892, -0.88718529, -
0.93971156,
        -0.82616118, -0.87695858, -0.92432545, -0.88283626, -
1.11034982,
         1.03116009],
       [ 0.03021123,  0.11414228, -0.36944043,  0.69004685,
0.17735109,
        -0.27510311,  0.5169395 ,  0.00490534,  0.34548287,
0.90061707,
         1.03116009],
       [-0.28955701, -0.56241015, -0.36944043,  0.7915854 , -
0.16115275,
        -0.12757576,  0.70279258, -0.15356813,  0.37247889,
0.90061707,
        -0.96978152],
       [-0.48141796, -0.56241015, -0.36944043,  0.90666242,
4.35787343,
         2.82297136,  1.21388855,  3.72182851,  2.58615292,
0.90061707,
         1.03116009],
       [-0.22560336,  0.79069471, -0.36944043,  0.70358532, -
0.06806419,
        -0.22520416,  0.74925585, -0.16077147,  0.39947492,
0.90061707,
         1.03116009],
       [ 0.15811853, -0.22413393, -0.36944043,  0.61558525, -
0.01728862,
```

```
         0.36056623,  0.21492825,  0.14176879,  0.19700473,
0.90061707,
        -0.96978152],
       [-1.56863   , -0.56241015, -0.36944043,  1.13004723,
1.81909468,
         0.85955581,  2.60778663,  1.40955654,  2.61314895,
0.90061707,
         1.03116009],
       [ 0.34997948,  0.45241849, -0.31963167,  0.10112326, -
0.62659552,
        -0.16662712, -0.55171569, -0.4777184 , -0.62637402, -
1.11034982,
        -0.96978152],
       [ 0.03021123, -1.915515  ,  3.96392163, -0.81949293, -
0.82969782,
        -0.84351734, -0.90019021, -0.85949539, -0.88283626, -
1.11034982,
         1.03116009],
       [ 1.11742327,  0.11414228, -0.02077911, -1.63857056, -
0.67737109,
        -0.86521254, -0.99311675, -0.73703862, -0.89633427, -
1.11034982,
         1.03116009],
       [ 1.05346962, -0.56241015,  0.3278822 , -2.20718644, -1.0581879
,
        -0.8608735 , -1.06281166, -1.00356218, -0.92333029, -
1.11034982,
        -0.96978152],
       [ 0.54184043,  0.45241849, -0.36944043,  0.52758517,  0.346603
,
         0.52761926,  0.09877008,  0.43710571,  0.25099678,
0.90061707,
        -0.96978152],
       [ 0.92556232, -0.22413393, -0.36944043,  0.60204677,
0.84589616,
         1.53210699,  0.19169662,  1.13582964,  0.58844709,
0.90061707,
        -0.96978152],
       [ 1.11742327, -0.22413393, -0.36944043,  0.52758517,
1.19286259,
         1.92695962, -0.01738809,  1.51760663,  0.50745902,
0.90061707,
        -0.96978152],
       [-1.05700081,  0.45241849, -0.36944043,  0.87281624,
0.30429003,
        -0.50073319,  1.42297326,  0.09854875,  0.96639144,
0.90061707,
         1.03116009],
       [-1.3128154 , -1.915515  , -0.31963167,  0.97435479,
0.69356943,
```

```
         0.38876999,  1.09773037,  0.56676582,  1.06087752,
0.90061707,
         1.03116009],
       [ 0.73370138, -1.23896257,  0.52711724, -1.19857018, -
1.03280012,
        -0.83050022, -0.94665348, -0.98195217, -0.89633427, -
1.11034982,
        -0.96978152],
       [ 1.56509882, -0.22413393,  4.66124426, -3.18872576, -
1.15973905,
        -0.8717211 , -1.10927493, -1.0972056 , -0.93682831, -
1.11034982,
        -0.96978152],
       [-1.18490811, -1.23896257, -0.36944043,  0.95404708,
0.17735109,
        -0.35320583,  1.12096201,  0.00490534,  0.77741926,
0.90061707,
         1.03116009],
       [ 1.43719152,  0.79069471, -0.36944043,  0.41927738, -
0.07652679,
         0.04598584, -0.06385136, -0.02390802, -0.08645353,
0.90061707,
        -0.96978152],
       [ 0.86160868,  0.11414228, -0.36944043,  0.21620028, -
0.93124897,
        -0.50290271, -0.45878915, -0.80907202, -0.7073621 ,
0.90061707,
        -0.96978152],
       [ 0.15811853,  0.79069471, -0.26982291,  0.16204639, -
0.39810543,
        -0.14710144, -0.45878915, -0.30483826, -0.50489191, -
1.11034982,
        -0.96978152],
       [-1.37676905, -0.90068636, -0.36944043,  0.96758555,
0.66818165,
         0.18049607,  1.39974162,  0.42990237,  1.18235964,
0.90061707,
         1.03116009],
       [ 0.15811853,  0.79069471, -0.36944043,  0.62235448,
0.28736483,
         0.53412782,  0.4472446 ,  0.37947899,  0.49396101,
0.90061707,
        -0.96978152],
       [-0.22560336,  0.79069471,  0.72635228, -0.94133919, -
0.94817416,
        -0.8391783 , -0.83049531, -0.93873213, -0.86933824, -
1.11034982,
         1.03116009],
       [-0.80118621, -1.57723878, -0.36944043,  0.79835464, -
0.38964283,
```

```
        -0.30547639,  0.47047623, -0.36246497,  0.10251865,
0.90061707,
        -0.96978152],
       [-0.09769607, -0.56241015, -0.26982291, -0.03426147, -
0.71122147,
        -0.64175198, -0.6911055 , -0.70822527, -0.8153462 , -
1.11034982,
         1.03116009],
       [-0.92909351,  0.79069471,  2.61908513,  0.21620028, -
0.38964283,
        -0.81314406, -0.38909425, -0.49932842, -0.53188794,
0.90061707,
         1.03116009],
       [ 0.41393313, -0.56241015, -0.36944043,  0.52758517, -
0.19500313,
        -0.0885244 , -0.04061973, -0.15356813, -0.12694756,
0.90061707,
        -0.96978152],
       [ 0.66974773, -0.56241015, -0.26982291, -1.17826247, -
0.89739858,
        -0.8500259 , -0.92342185, -0.90991877, -0.89633427, -
1.11034982,
         1.03116009],
       [-0.35351066, -0.56241015, -0.17020539,  0.1078925 ,
1.91218323,
         2.5778156 , -0.6214106 ,  1.02777955, -0.11344955, -
1.11034982,
         1.03116009],
       [-1.12095446,  0.45241849, -0.26982291,  0.54112364, -
0.38118024,
        -0.39659623,  0.14523335, -0.40568501, -0.12694756,
0.90061707,
         1.03116009],
       [ 0.15811853, -0.22413393, -0.26982291, -0.69764667, -
0.89739858,
        -0.6656167 , -0.83049531, -0.83068204, -0.86933824, -
1.11034982,
         1.03116009],
       [-0.22560336,  0.11414228, -0.36944043,  0.71035456,
1.13362442,
         0.30198919,  0.58663441,  0.8188827 ,  0.8179133 ,
0.90061707,
         1.03116009],
       [-0.35351066,  0.11414228, -0.36944043,  0.76450845,
0.79512058,
         0.21520839,  0.79571912,  0.53795246,  0.84490933,
0.90061707,
         1.03116009],
       [-0.99304716, -0.90068636, -0.22001415,  0.42604662,
0.21120147,
```

```
         0.15229231, -0.29616771,  0.19219217, -0.19443763,
0.90061707,
         1.03116009],
       [ 0.73370138, -0.90068636,  1.62290994, -1.48287812, -
0.99048714,
        -0.85436494, -0.96988512, -0.96754549, -0.90983228, -
1.11034982,
         1.03116009]])
```

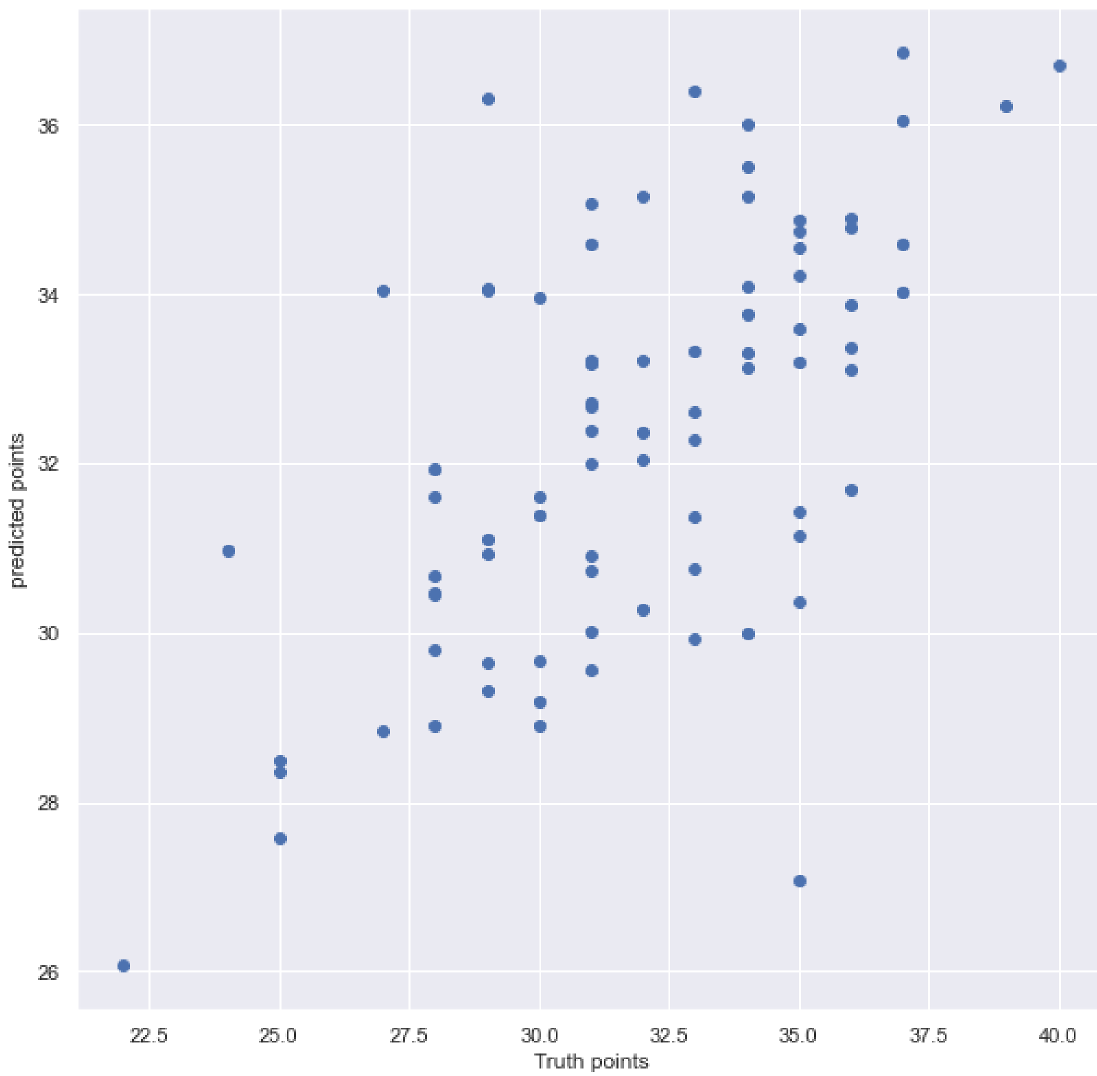
```
#### Model Training
from sklearn.linear_model import LinearRegression

regression=LinearRegression()

regression

LinearRegression()

regression.fit(X_train, y_train)

LinearRegression()

print(regression.coef_)

[[-1.21117536 -0.56706065 -0.12035252  0.79369256 -3.55532765 -
1.72320298
   0.49361804  5.46481362 -0.04530436  0.31041538  0.1591916 ]]

print(regression.intercept_)

[32.3190184]

#### Prediction for the test data
reg_pred=regression.predict(X_test)

reg_pred

array([[29.80432572],
       [31.61071065],
       [30.73014592],
       [34.58491894],
       [32.35863837],
       [27.58007776],
       [30.36907285],
       [31.94145267],
       [36.01307871],
       [30.75069173],
       [29.54877507],
       [34.09509279],
       [29.64882182],
       [32.00585198],
       [36.40617546],
       [31.42711725],
```

```
[33.33381035],
[34.02910159],
[28.49873916],
[35.16579465],
[33.22024225],
[34.55160794],
[30.92189997],
[29.32146855],
[33.12666656],
[30.4810564 ],
[34.59746657],
[33.57686519],
[33.1030309 ],
[33.18337889],
[31.39203256],
[34.78314735],
[34.04470933],
[31.60202042],
[34.03663356],
[29.19871698],
[28.90059558],
[28.89787311],
[33.96700741],
[30.28273097],
[34.8669716 ],
[35.06170097],
[36.3077914 ],
[31.68988943],
[29.99483302],
[33.3634101 ],
[34.07017095],
[34.89172421],
[33.29372785],
[33.2248847 ],
[36.84607614],
[30.91854501],
[31.36269269],
[28.844181  ],
[28.35025506],
[32.27919608],
[32.60007873],
[32.38141252],
[35.50179954],
[36.70607581],
[29.92935775],
[26.08386059],
[36.22513487],
[30.66696089],
[31.11362841],
[31.14057926],
```

```
       [36.05348884],
       [32.67282739],
       [30.44540749],
       [35.14557424],
       [32.04100112],
       [33.18585853],
       [32.71730626],
       [30.01110205],
       [27.06618946],
       [34.22490375],
       [30.98029946],
       [33.7562864 ],
       [33.87397263],
       [34.74045167],
       [29.65709425]])
```

```
#### Assumption of linear Regression
plt.scatter(y_test, reg_pred)
plt.xlabel("Truth points")
plt.ylabel("predicted points")
```

```
Text(0, 0.5, 'predicted points')
```



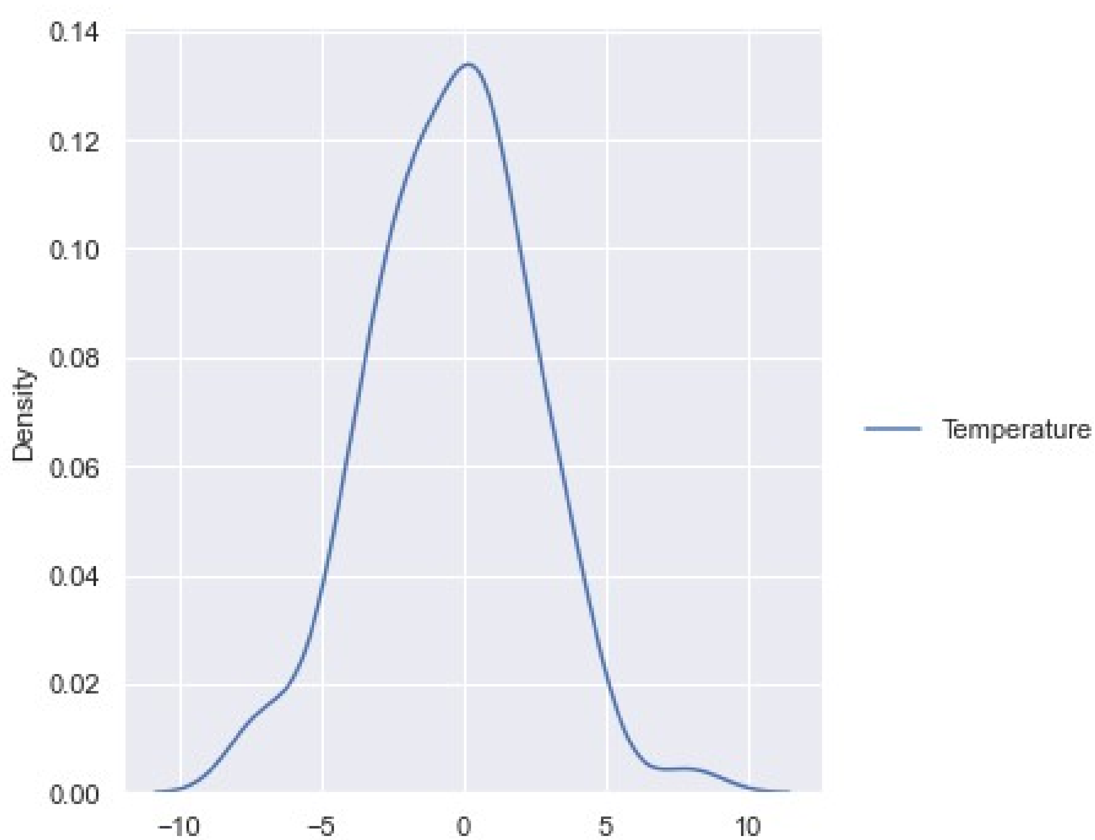
With respect to test data and predicted data our scatter plot should be linearlly disrubuted

```
residuals=y_test-reg_pred

residuals

     Temperature
51     -1.804326
224    -1.610711
41      0.269854
192     2.415081
67     -0.358638
..           ...
245    -6.980299
172     0.243714
183     2.126027
199     0.259548
125     0.342906
```

```
[81 rows x 1 columns]
```

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

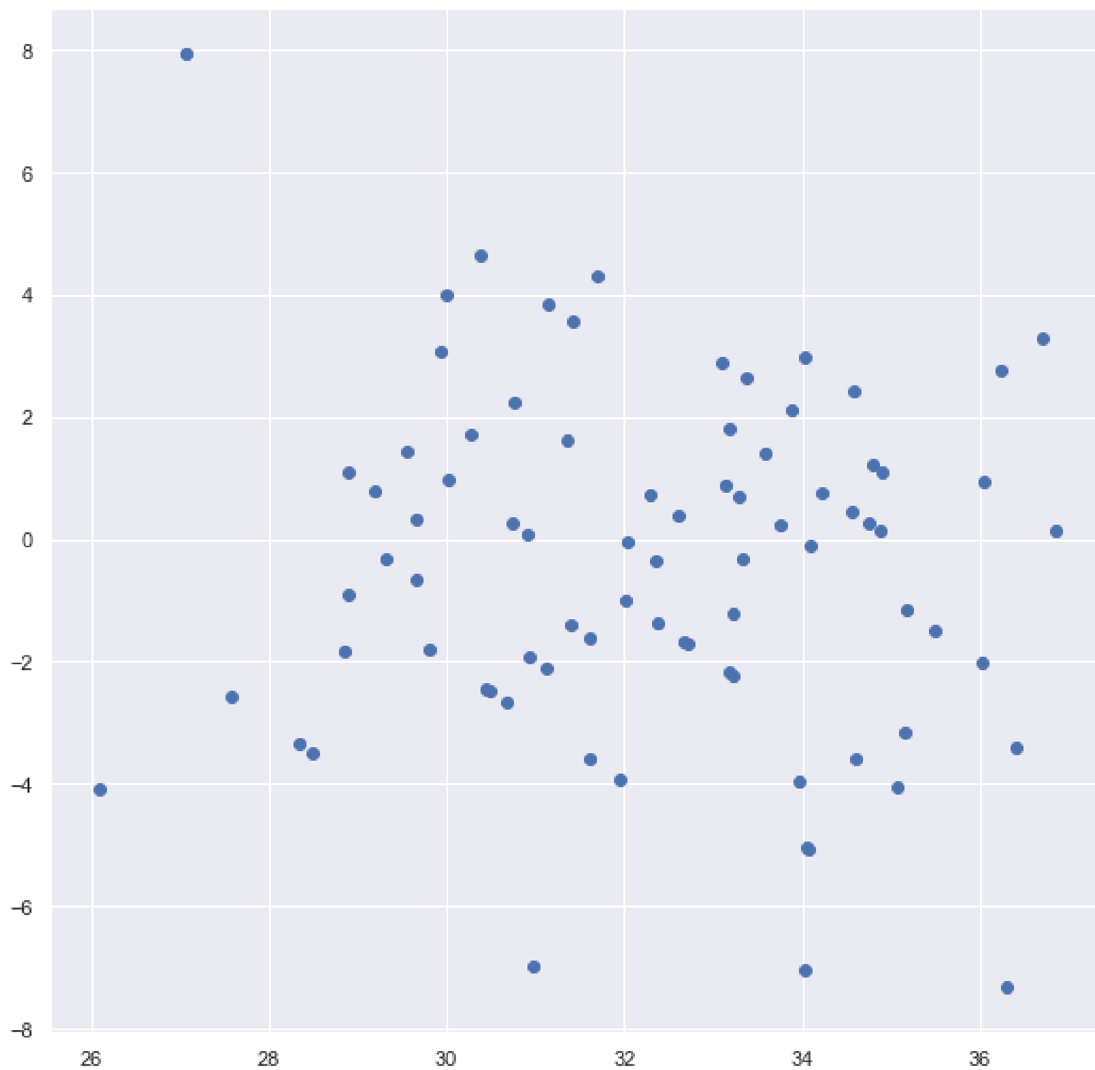
```
<seaborn.axisgrid.FacetGrid at 0x213f72e4c70>
```



With respect to residuels our graph should be normally distrubuted and if there is outliers in dataset it will be skewed

```python
## Scatter plot of residuels and predicted points
## Uniform distrubution
plt.scatter(reg_pred, residuals)
```

```
<matplotlib.collections.PathCollection at 0x213f926f880>
```



Graph of reg_pred and residuals should be uniformally distrubuted

Performance Metrics

```
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)))
```

```
8.016503622543366
2.2234819896738385
2.831343077506392
```

```
from sklearn.metrics import r2_score
```

R-squared score

```
score=r2_score(y_test, reg_pred)
```

```
score
```

```
0.356305467292779
```

Adjusted R squared

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

```
0.25368749831046844
```

conclusion: Adjusted R squared will be always less than R-squared

Ridge Regression

```
##Ridge
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
ridge_regressor=Ridge()
```

```
parameters={'alpha':[1,2,3,5,6,47,8,9,89,5,2,10]}
ridgecv=GridSearchCV(ridge_regressor,parameters,scoring='neg_mean_squa
red_error', cv=5)
ridgecv.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1, 2, 3, 5, 6, 47, 8, 9, 89, 5, 2,
10]},
             scoring='neg_mean_squared_error')
```

```
print(ridgecv.best_params_)
```

```
{'alpha': 47}
```

```
print(ridgecv.best_score_)
```

```
-5.696939972304345
```

```
ridge_pred=ridgecv.predict(X_test)
```

```
ridge_pred
```

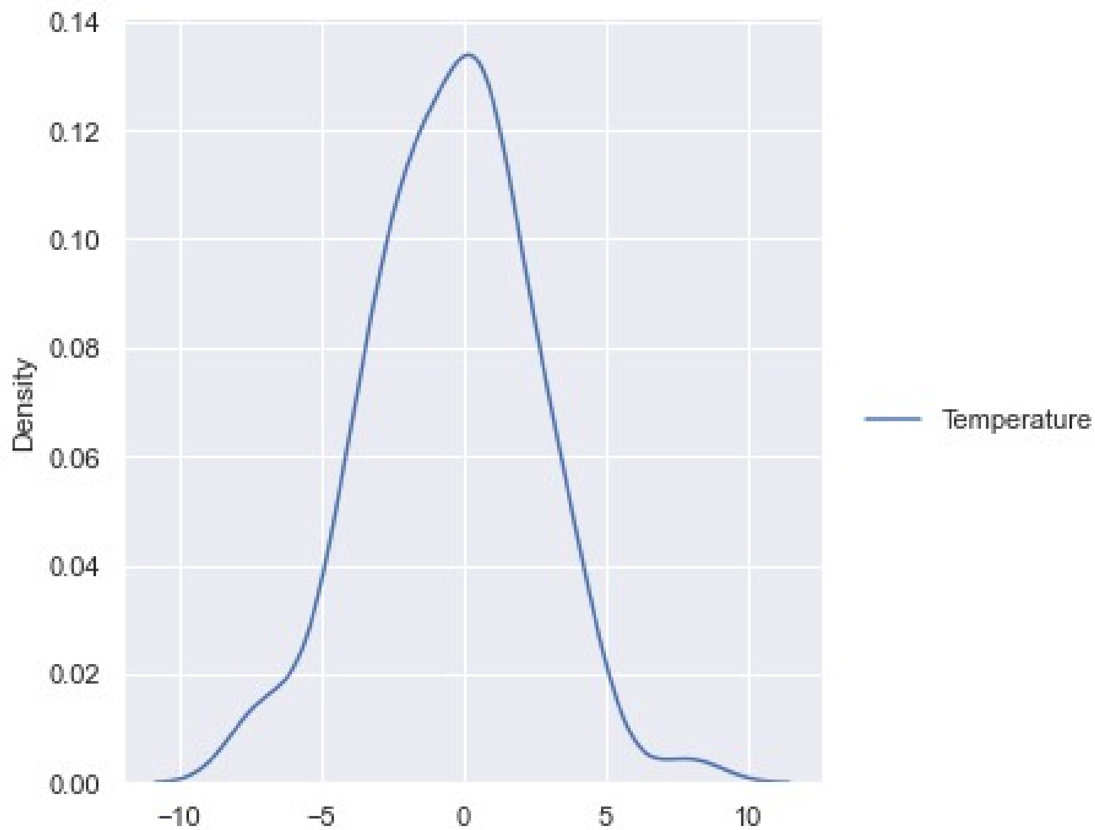
```
array([[29.94108029],
       [31.76255471],
       [30.791717  ],
       [34.31117955],
       [32.51369767],
       [27.85978095],
       [30.92874271],
       [31.54199374],
       [36.8475654 ],
       [30.9651734 ],
       [29.68408041],
       [33.9444298 ],
       [29.87478071],
       [32.25061527],
```

```
[37.33647968],
[31.32112604],
[32.74772623],
[34.00198035],
[28.90136328],
[35.19416916],
[33.11412788],
[33.2749328 ],
[30.866686  ],
[29.69885022],
[32.77245372],
[30.52404157],
[34.30414978],
[33.49450705],
[33.45962492],
[33.1015368 ],
[31.45657782],
[35.09356205],
[33.91118053],
[31.70899562],
[33.95358272],
[29.37486482],
[29.4504296 ],
[29.15672039],
[34.23462908],
[30.43735638],
[34.69034431],
[34.76008915],
[35.6641976 ],
[31.98328466],
[30.0385219 ],
[33.49385722],
[33.89720157],
[36.30116232],
[33.45239605],
[33.03855546],
[37.35575866],
[30.88948065],
[31.14791114],
[29.15552288],
[28.67600904],
[32.3272601 ],
[32.67806969],
[32.45353756],
[34.9534923 ],
[36.5090568 ],
[30.03208462],
[26.64592561],
[35.8183171 ],
[30.987609  ],
```

```
       [31.21645196],
       [31.0356334 ],
       [36.16525824],
       [32.75137565],
       [30.42365119],
       [34.68228166],
       [31.90231119],
       [32.53703286],
       [32.66282222],
       [30.25262092],
       [33.07540182],
       [33.89602943],
       [30.93970635],
       [34.10848385],
       [34.27794362],
       [34.36227647],
       [29.82794407]])
```

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

```
<seaborn.axisgrid.FacetGrid at 0x213f68c98b0>
```




```
score=r2_score(ridge_pred, y_test)
```

```
score
```

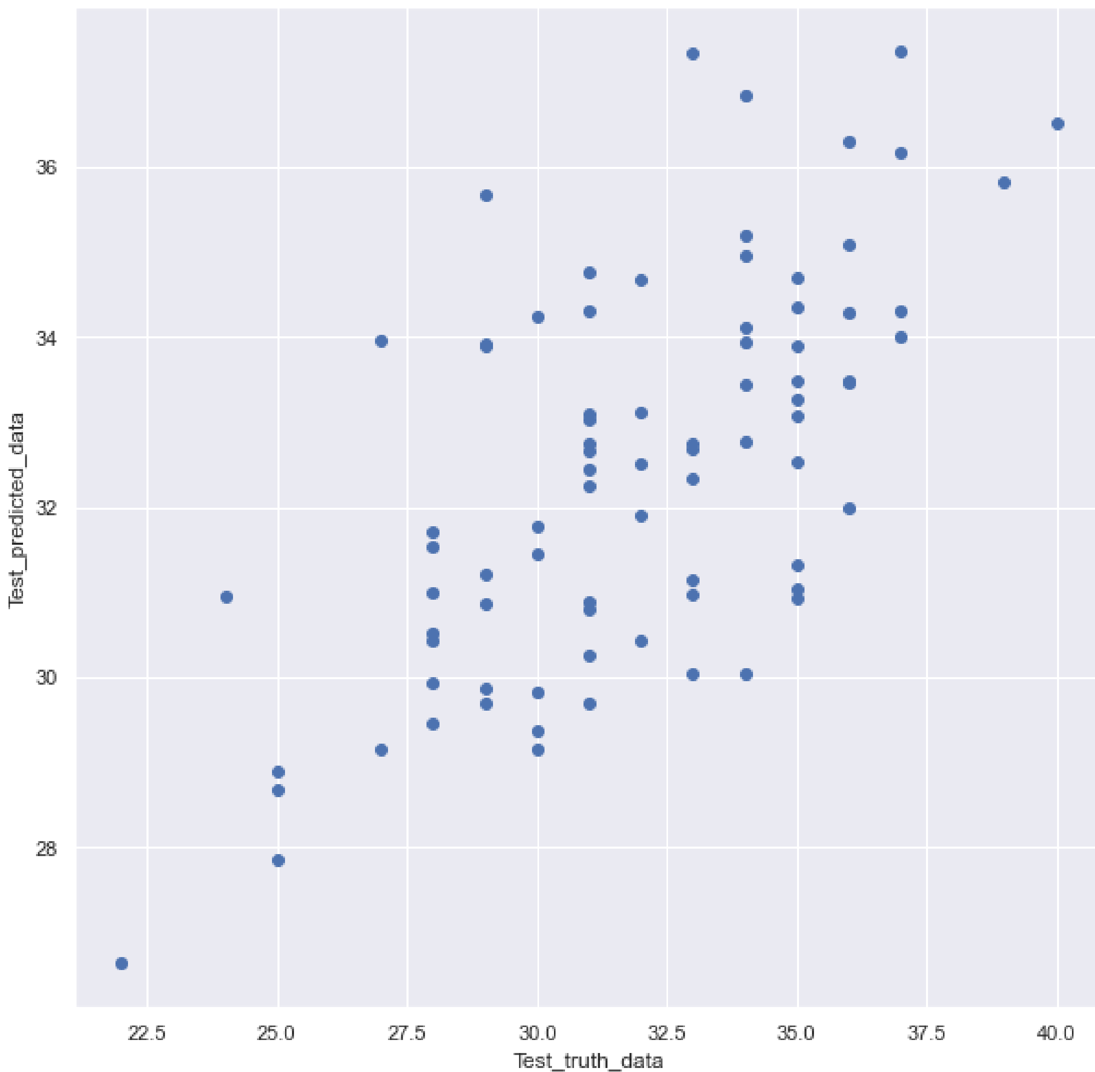
```
-0.40957217455615624
```

Assumptions of ridge regression

y_test and ridge_pred should be linearlly increasing

```
#y_test and ridge_pred should be linearlly increasing
plt.scatter(y_test, ridge_pred)
plt.xlabel("Test_truth_data")
plt.ylabel("Test_predicted_data")
```

```
Text(0, 0.5, 'Test_predicted_data')
```



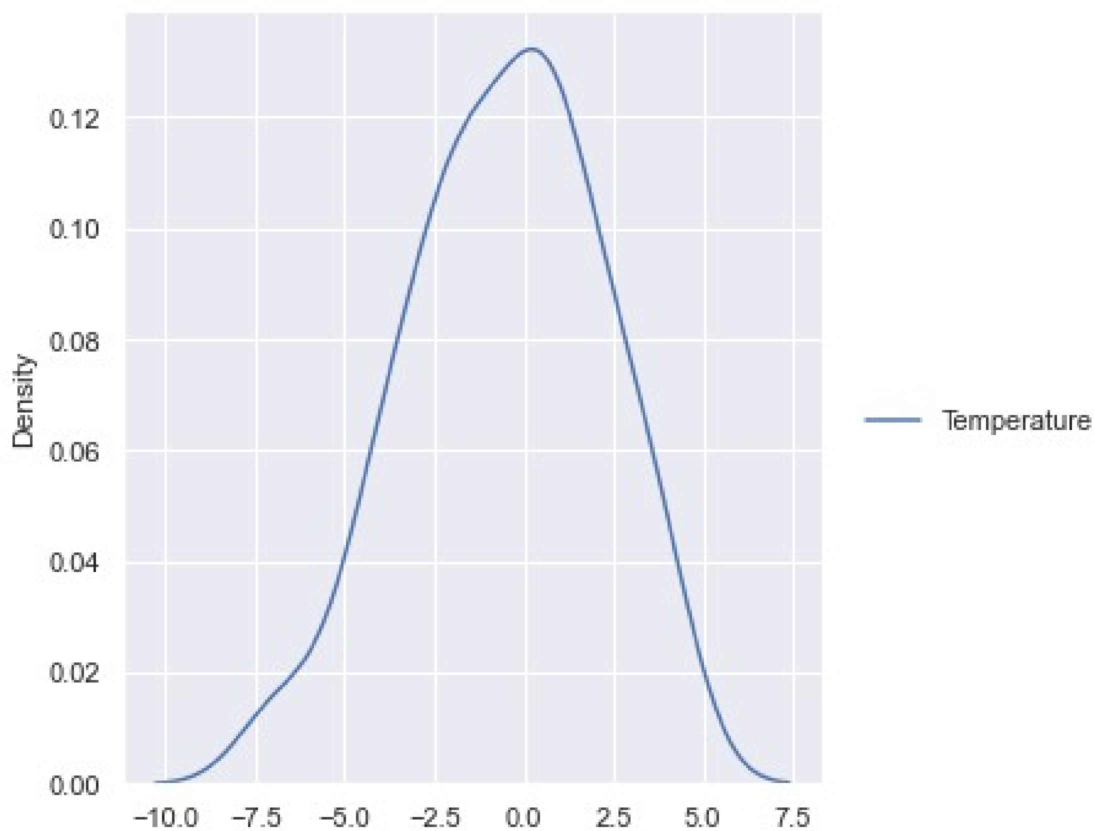

```
#residuel/error
ridge_residual=y_test-ridge_pred
ridge_residual
```

```
    Temperature
51     -1.941080
```

```
224    -1.762555
41      0.208283
192     2.688820
67     -0.513698
..           ...
245    -6.939706
172    -0.108484
183     1.722056
199     0.637724
125     0.172056

[81 rows x 1 columns]
```

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

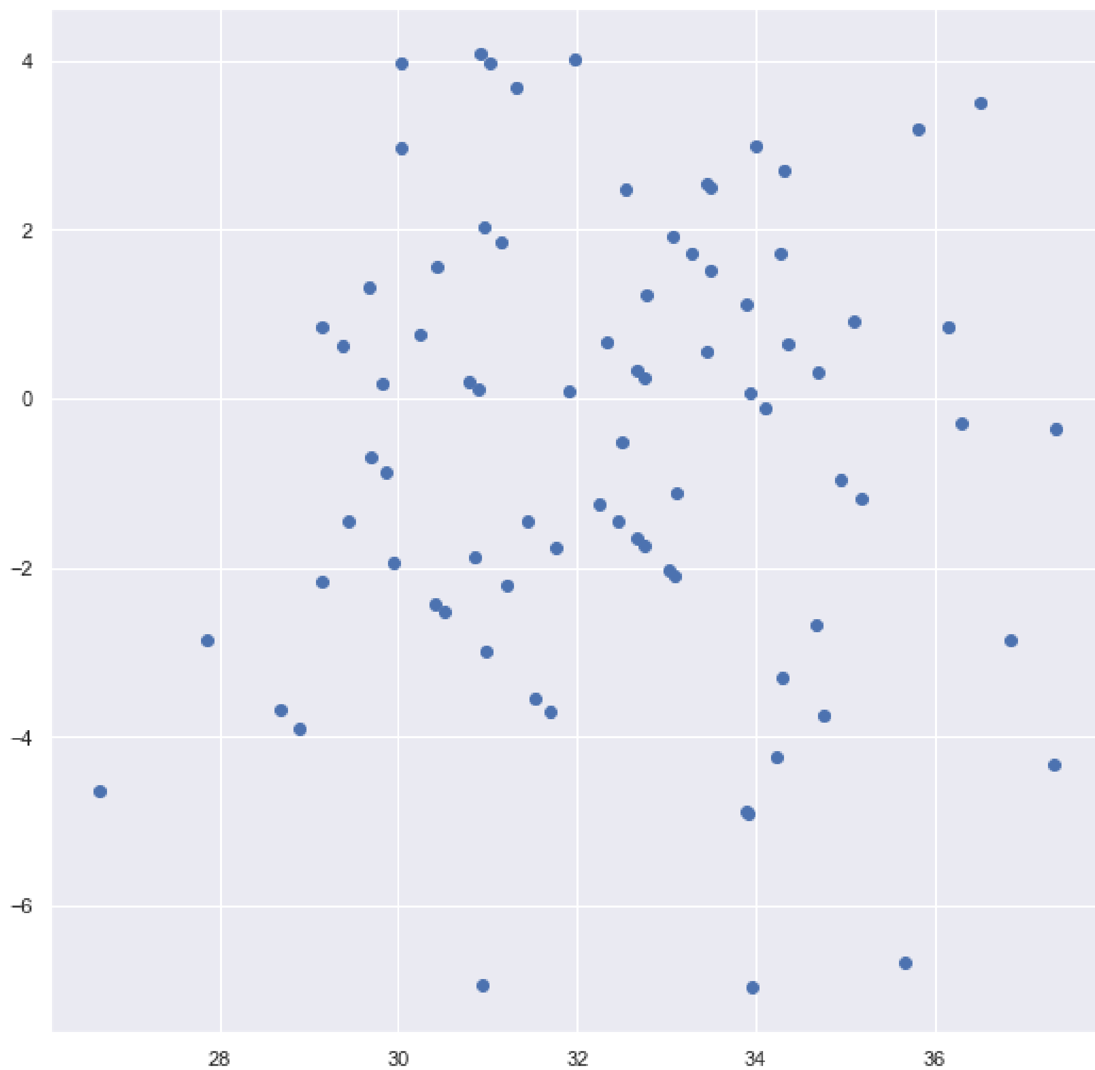
<seaborn.axisgrid.FacetGrid at 0x213f92d7b20>



residual plot is normally distrubuted and little skewed beacause of outliers

```
#scatterplot with ridge_pred and ridge_resideal
plt.scatter(ridge_pred, ridge_residual)
```

<matplotlib.collections.PathCollection at 0x213f93e92e0>



Scatterplot with ridge prediction and ridge residual, it should be uniformally distrubtion (it should not have any shape)

```
#performance metrix
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)))
```

```
7.339619547763014
2.1835739336118043
2.7091732221773888
```

```
## R squard and adjusted R square
from sklearn.metrics import r2_score
score=r2_score(y_test, ridge_pred)
print(score)
```

```
0.41065666561163705
```

```
#Adjusted R squared
1 - (1-score)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
```

```
0.31670338041928936
```

Lasso Regression

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

```
lasso=Lasso()
```

```
lasso
```

```
Lasso()
```

```
parameters={'alpha':[1,2,3,5,6,47,8,9,89,5,2,10]}
lassocv=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error'
, cv=5)
lassocv.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1, 2, 3, 5, 6, 47, 8, 9, 89, 5, 2,
10]},
             scoring='neg_mean_squared_error')
```

```
print(lassocv.best_params_)
```

```
{'alpha': 1}
```

```
print(lassocv.best_score_)
```

```
-7.305375337514984
```

```
lasso_pred=lassocv.predict(X_test)
```

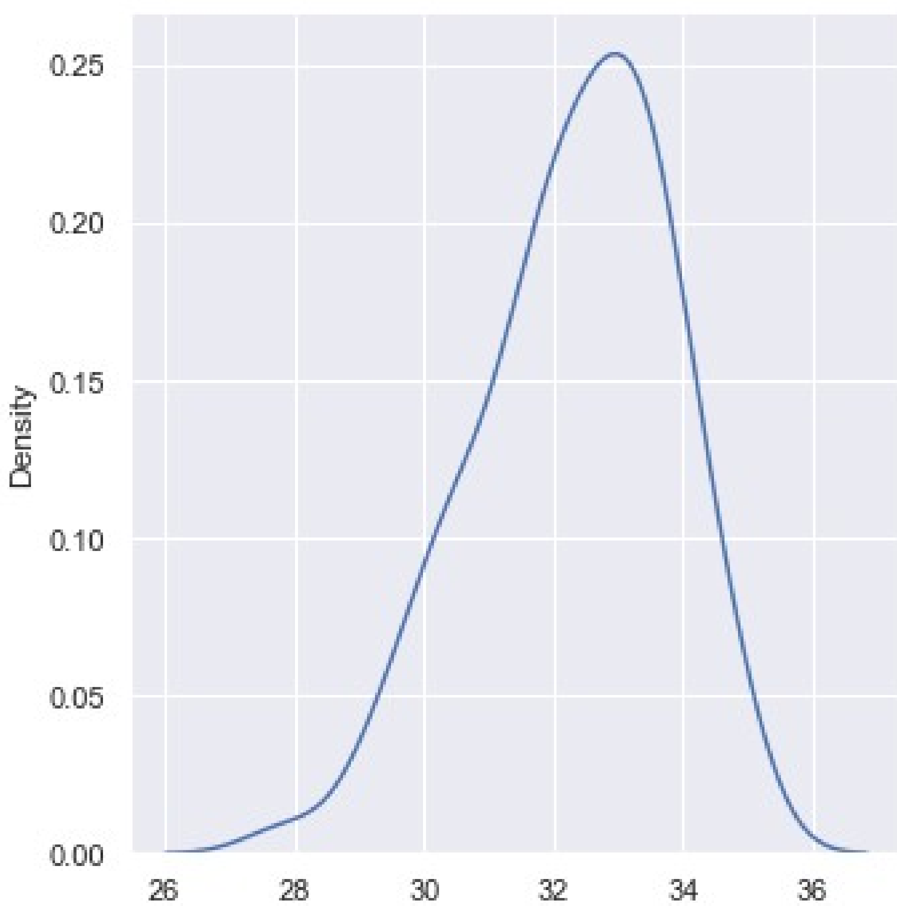
```
lasso_pred
```

```
array([31.06500611, 31.6929418 , 31.37273876, 33.25264037,
32.53495303,
       29.38956497, 31.47868541, 32.36389873, 34.63408271,
31.74292057,
       30.82932091, 33.62246138, 30.38837138, 32.57748256,
34.76114641,
       32.09211617, 33.08122523, 33.70729096, 29.22852024,
33.81016918,
       32.87835676, 33.21395045, 31.3805652 , 30.42986749,
32.54888306,
       31.81695348, 33.33561581, 32.8735165 , 32.38854412,
32.5950227 ,
       31.63372511, 33.84347762, 33.79897898, 31.88544081,
33.34529685,
       30.61093   , 30.05359843, 30.01008413, 33.33766721,
```

```
30.38047906,
       33.22866851, 34.40858702, 33.72502736, 32.45998441,
30.87367185,
       33.02086066, 33.39758918, 33.72371545, 33.26714532,
32.81026343,
       34.97883206, 32.06912862, 31.39808898, 29.72594618,
29.2217436 ,
       32.40359028, 32.17835727, 31.92757681, 34.17831647,
34.44233802,
       30.4629787 , 27.86091037, 34.32290396, 31.56107054,
31.78029808,
       32.29293423, 34.52342663, 32.84467117, 31.49456821,
33.7869353 ,
       32.2814977 , 33.22620675, 32.48953551, 30.53665078,
32.63202256,
       33.7570561 , 31.42102746, 33.25413369, 33.43403145,
33.49063868,
       30.18660173])
```

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

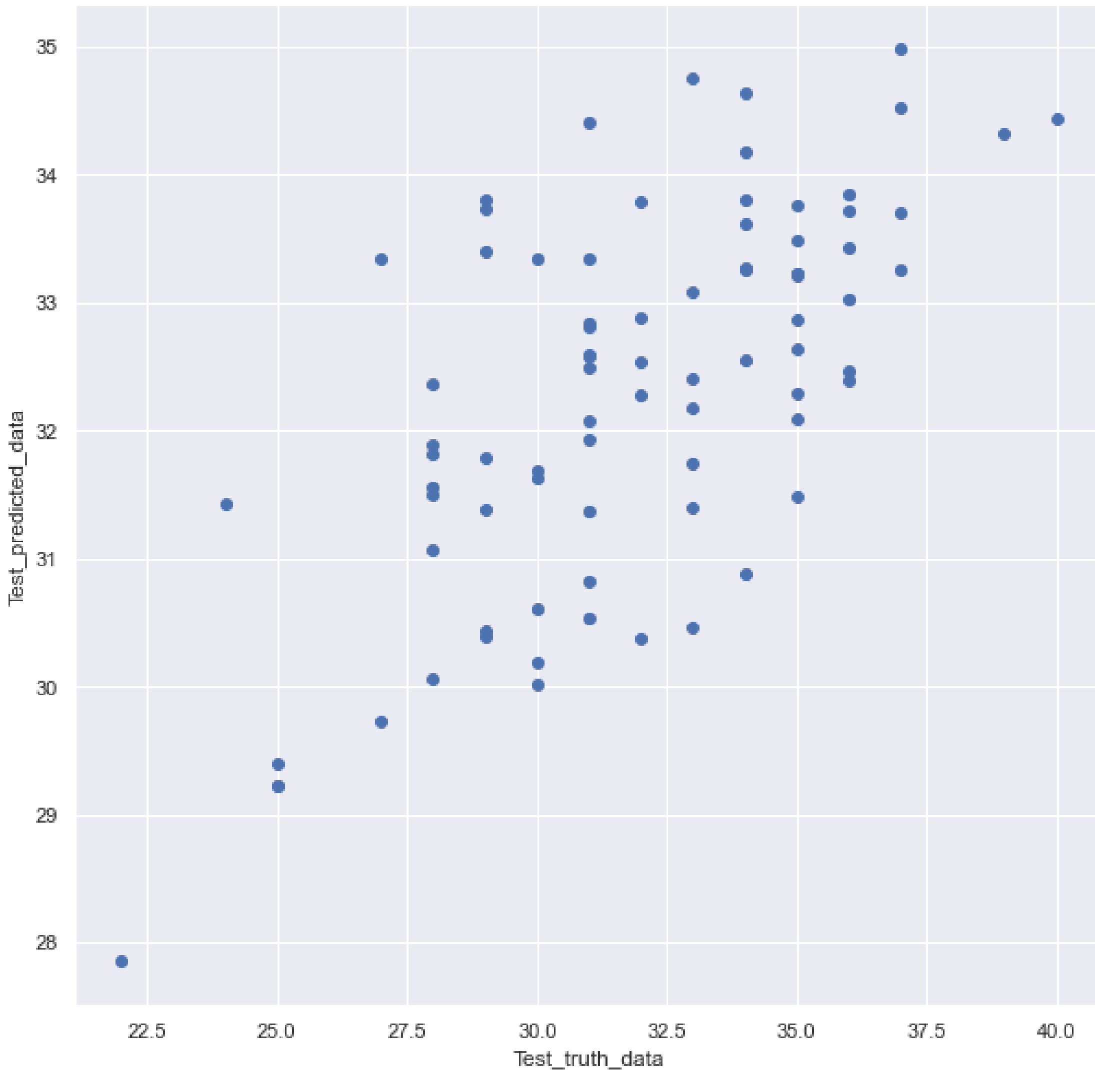
```
<seaborn.axisgrid.FacetGrid at 0x213f9418760>
```



Assumption of Lasso regression

```
#y_test and lasso_pred should be linearlly increasing
plt.scatter(y_test, lasso_pred)
plt.xlabel("Test_truth_data")
plt.ylabel("Test_predicted_data")
```

```
Text(0, 0.5, 'Test_predicted_data')
```



y_test and lasso_pred should be linearlly increasing

```
lasso_pred=lasso_pred.reshape(81,1)
```

```
lasso_residuals=y_test-lasso_pred
```

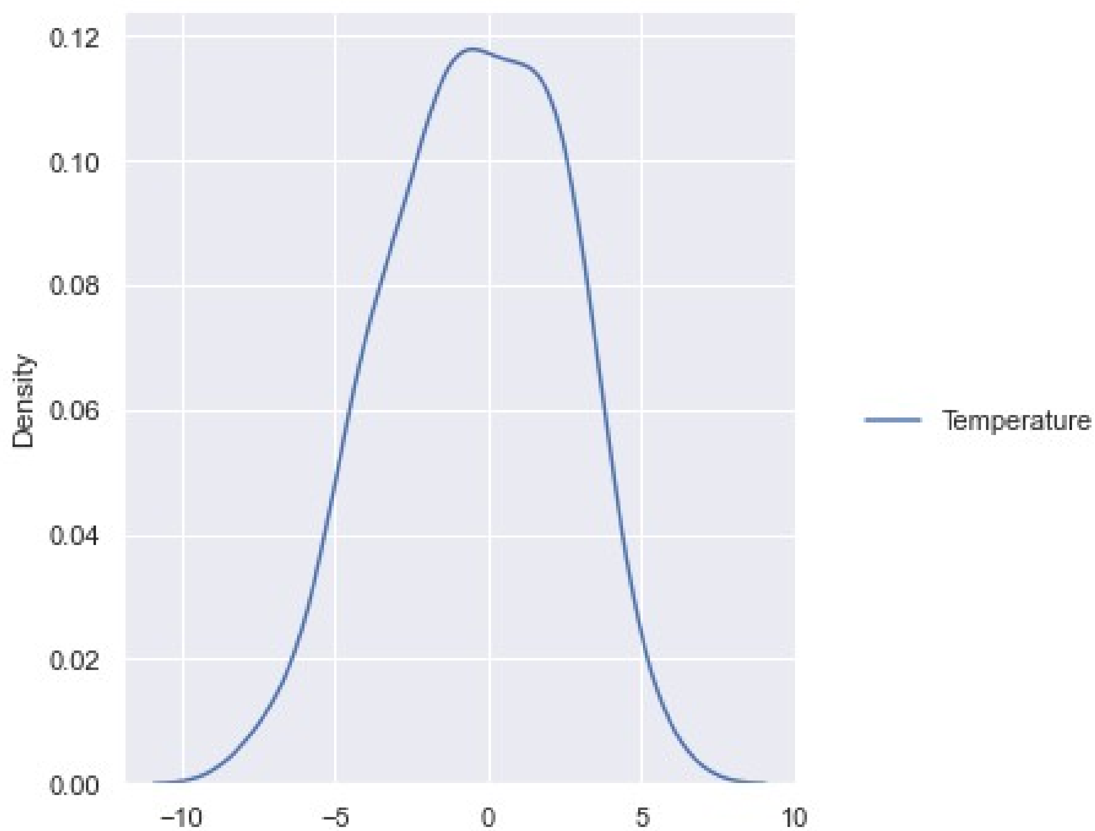
```
lasso_residuals
```

```
     Temperature
51     -3.065006
```

```
224    -1.692942
41     -0.372739
192     3.747360
67     -0.534953
..           ...
245    -7.421027
172     0.745866
183     2.565969
199     1.509361
125    -0.186602

[81 rows x 1 columns]
```

```
sns.displot(lasso_residuals, kind="kde")
```

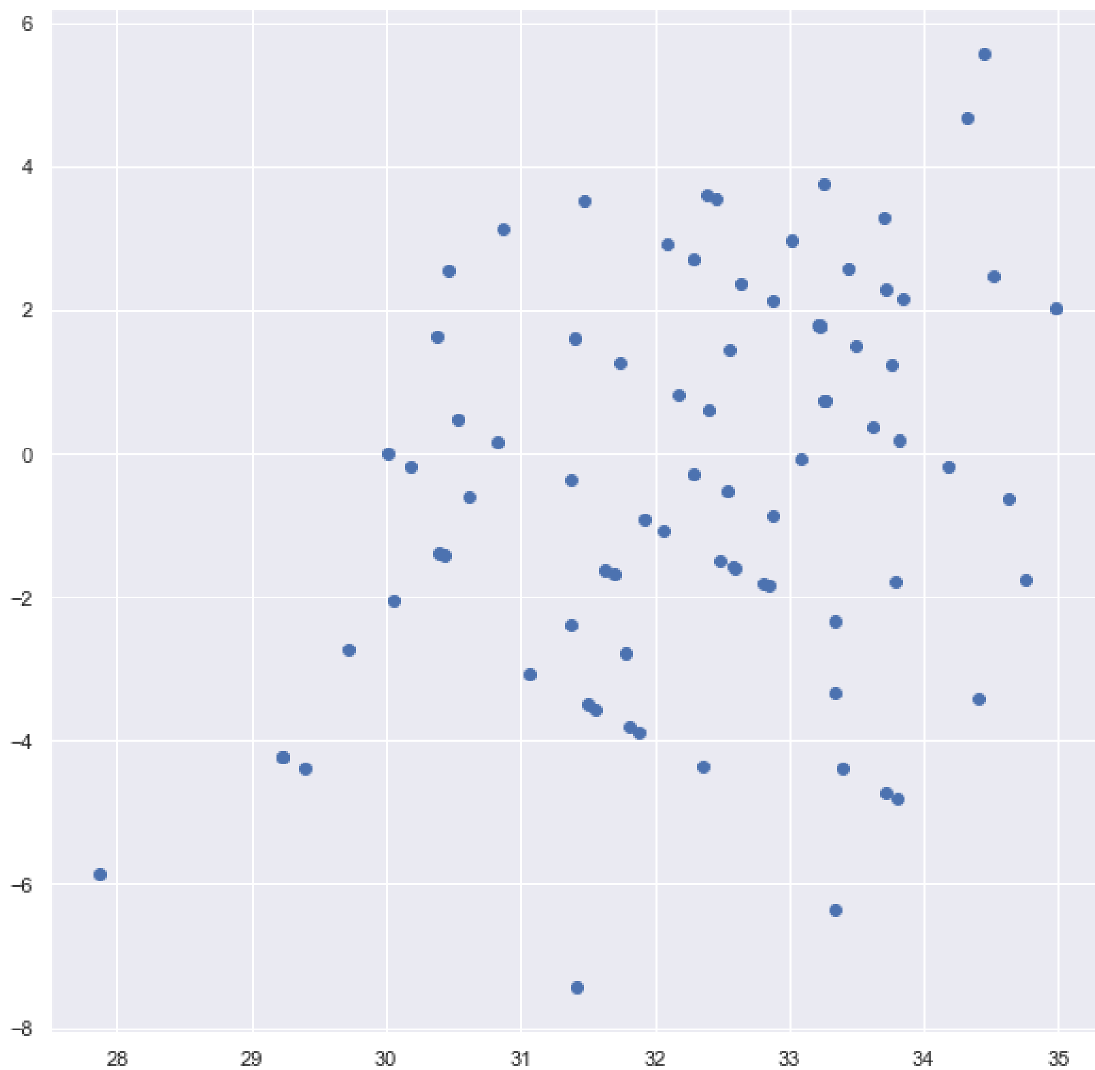
```
<seaborn.axisgrid.FacetGrid at 0x213f947adc0>
```



residuels are normally distrubuted

```
plt.scatter(lasso_pred, lasso_residuals)
```

```
<matplotlib.collections.PathCollection at 0x213fa6f7340>
```



performance metrix

```
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)))
```

```
7.8696515835539005
2.317044907898472
2.8052899286087882
```

R-squared

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

```
0.368097123489204
```

Adjusted R-squared

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

```
0.26735898375559874
```

Elastic Net Regression

```
from sklearn.linear_model import ElasticNet
```

```
elasticnet=ElasticNet()
elasticnet
```

```
ElasticNet()
```

```
elasticnet.fit(X_train, y_train)
```

```
ElasticNet()
```

```
elasticnet_pred=elasticnet.predict(X_test)
```

```
elasticnet_pred
```

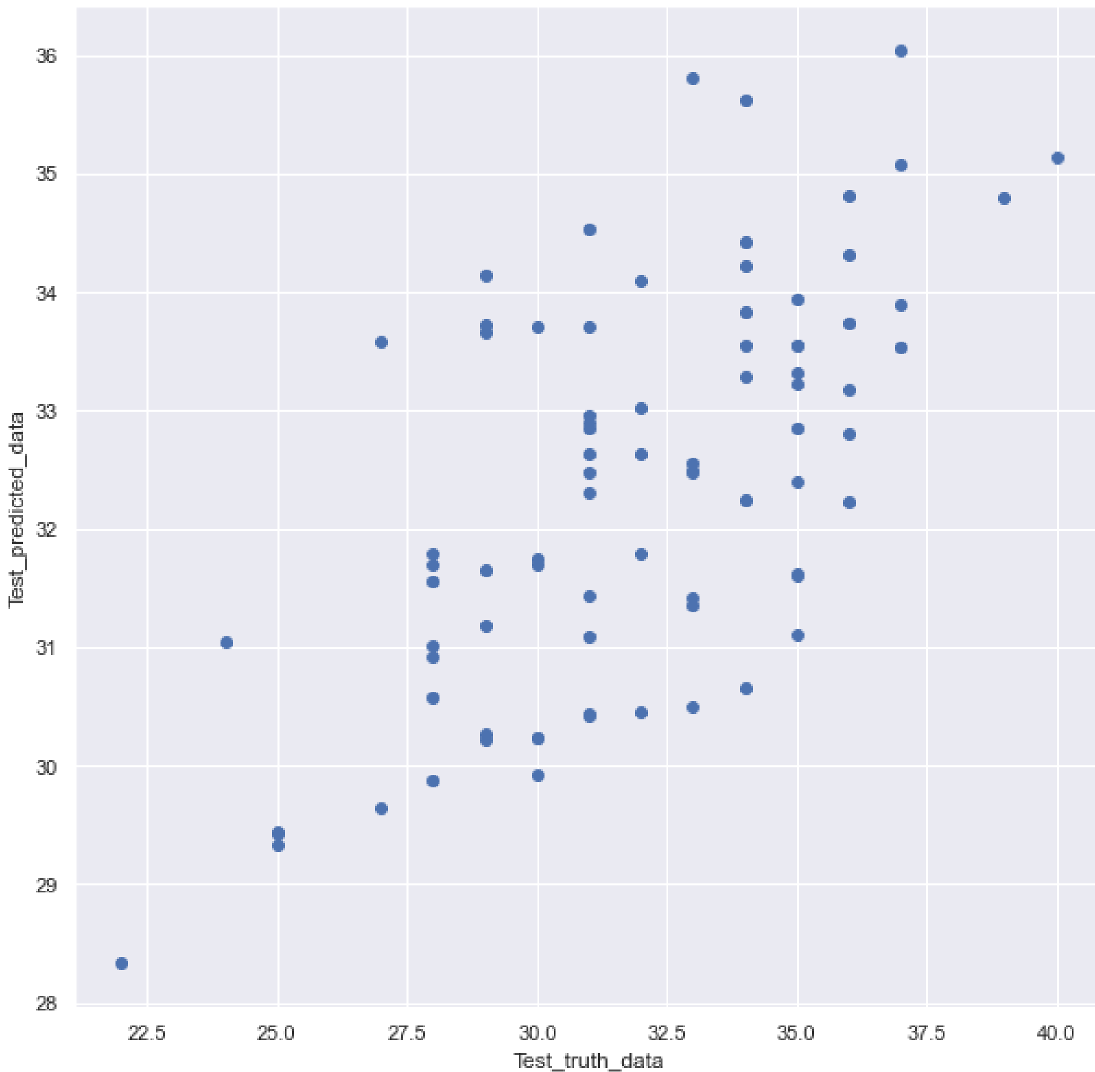
```
array([30.58233822, 31.74586174, 31.09445454, 33.53022515,
32.63419427,
       29.32869664, 31.11079064, 31.70167753, 35.6154579 ,
31.35850018,
       30.43450915, 33.83885723, 30.2240347 , 32.475088  ,
35.81421841,
       31.62292661, 32.49882967, 33.89578669, 29.44833766,
34.21716181,
       33.02648884, 33.31803187, 31.18767492, 30.27232433,
32.23581132,
       31.01871744, 33.70732584, 33.22936002, 32.80188594,
32.84543675,
       31.6976918 , 34.3135809 , 33.72355759, 31.78543796,
33.58215229,
       30.23996414, 29.87392299, 29.91891528, 33.70900631,
30.45500239,
       33.9437523 , 34.53407031, 34.14547336, 32.22330939,
30.66281449,
       33.17190122, 33.6626742 , 34.8057206 , 33.28279936,
32.965061  ,
       36.03624783, 31.42861309, 31.414447  , 29.64474524,
29.42816641,
       32.48118908, 32.54893688, 32.29866047, 34.42618513,
35.14097454,
       30.49554117, 28.34188711, 34.7989069 , 31.55191832,
31.65671625,
       31.59752962, 35.07922341, 32.90263715, 30.92382263,
34.10327908,
       31.79145966, 32.85482119, 32.63669002, 30.41606874,
32.39651287,
       33.54428462, 31.04246852, 33.55023753, 33.73096224,
```

```
33.55844474,
       30.22928558])
```

Assumption of Elastic-Net regression

```
plt.scatter(y_test, elasticnet_pred)
plt.xlabel("Test_truth_data")
plt.ylabel("Test_predicted_data")
```

```
Text(0, 0.5, 'Test_predicted_data')
```



y_test and elasticnet_pred should be linearlly increasing

```
elasticnet_pred=elasticnet_pred.reshape(81,1)
```

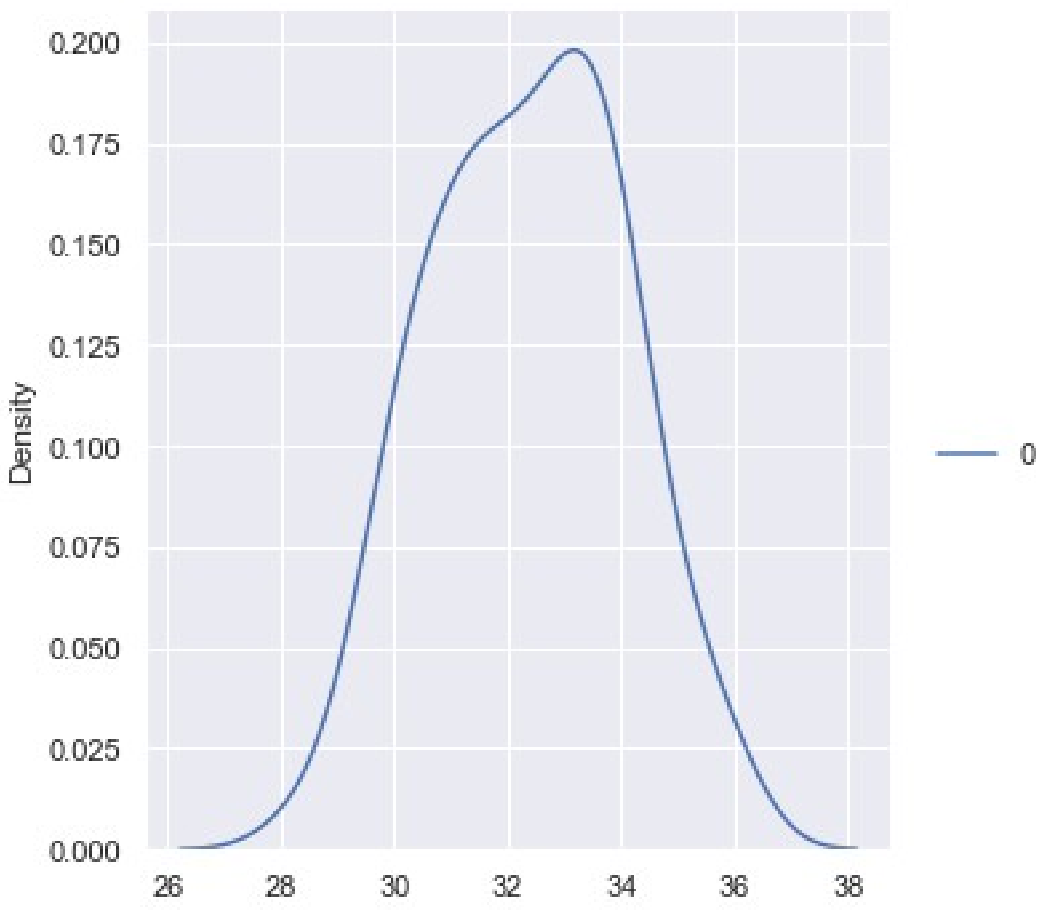
```
elasticnet_residuals=y_test-elasticnet_pred
```

```
elasticnet_residuals
```

```
     Temperature
51     -2.582338
224    -1.745862
41     -0.094455
192     3.469775
67     -0.634194
..           ...
245    -7.042469
172     0.449762
183     2.269038
199     1.441555
125    -0.229286

[81 rows x 1 columns]
```

```python
## residual plot
sns.displot(elasticnet_pred, kind="kde")
```

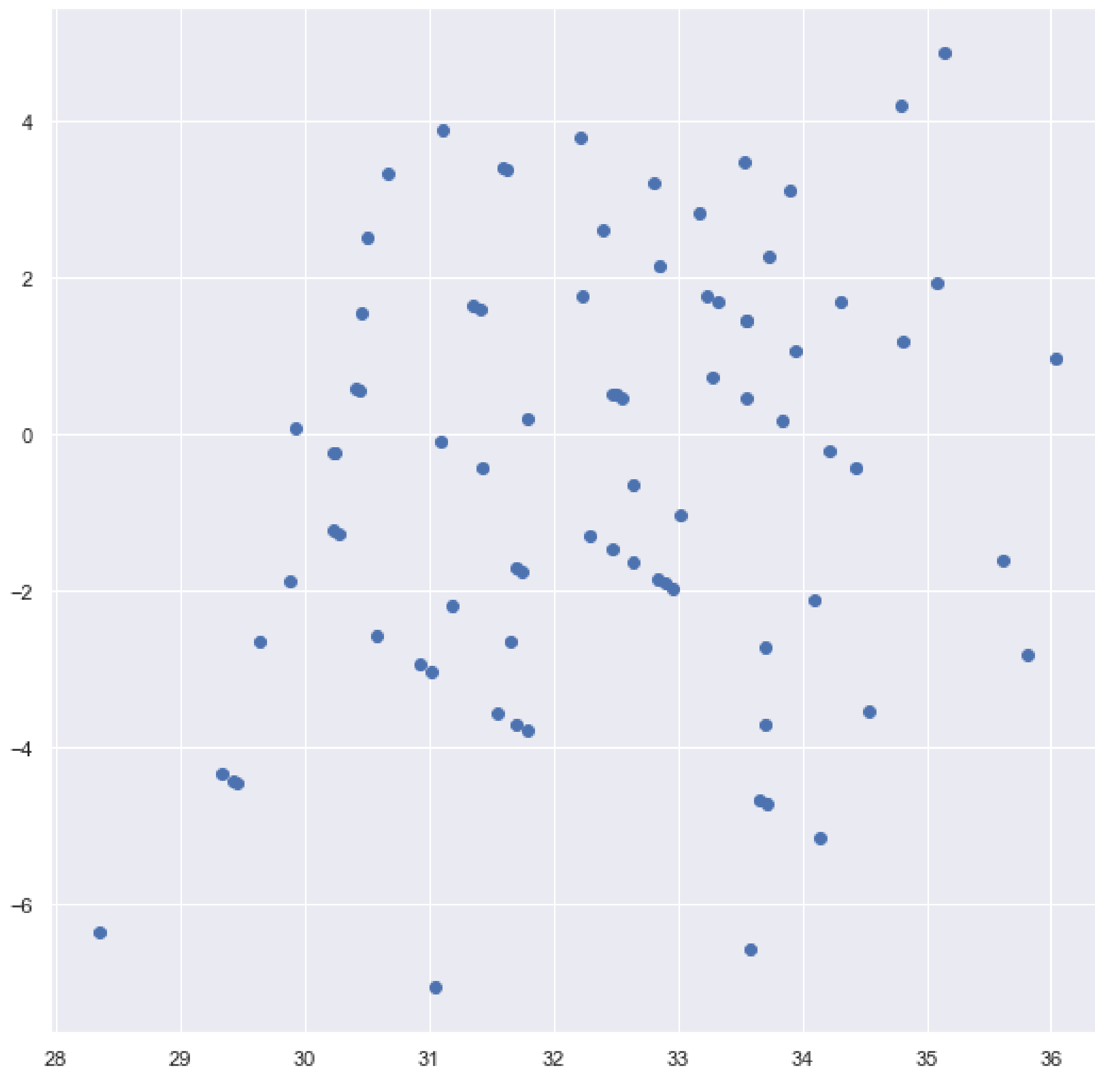
```
<seaborn.axisgrid.FacetGrid at 0x213fa714370>
```



Residual plot is normally distubuted

```python
#scatterplot with elasticnet_pred and EN_residuel
plt.scatter(elasticnet_pred, elasticnet_residuals)
```

```
<matplotlib.collections.PathCollection at 0x213fa847d30>
```



Scatterplot of elasticnet_pred and elasticnet_residuals is uniformally distrubuted

Performance metrics

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

```
7.736339628192311
2.2886972607562712
2.781427624115413
```

R-squared

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

```
0.3788015628372323
```

Adjusted R-squared

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

```
0.2797699279272259
```