Linear Regression on Algerian Forest Fire Dataset

Submitted by : Nilutpal Das

Life Cycle of Machine Learning Project

- Understanding the problem
- Data Collection
- Data Cleaning
- Exploratory Data Analysis
- Data Pre-Processing
- Model Training
- Model Evaluation

1) Problem Statement

- Implement Linear Regression on Algerian Forest Fire Dataset.

2) Data Collection

- The dataset is downloaded from UCI website.
- The dataset includes 244 instances that regroup a data of two regions of Algeria,
- 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 fire (138 classes) and not fire (106 classes) classes.

- This dataset comprises weather data attributes.

- Can be used to predict the forest fire weather type.
- Prediction can help to monitor on fire and make proper prevention.

Importing the libraries

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

```
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Loading the dataset

In [2]:
```
df = pd.read_csv('D:\Data Science\Algerian dataset\Algerian_forest_fires_dataset.csv',
```

Showing top 5 records

In [3]:
```
df.head()
```

Out[3]:

	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

3) Data Cleaning

3.1) Removing the unwanted rows from the dataset

In [4]:
```
df.drop(index=[122,123], inplace=True)
df.reset_index(inplace=True)
df.drop('index', axis=1, inplace=True)
```

3.2) Adding new column name 'Region' in dataset

In [5]:
```
for i in range(len(df)):
    if i<=121:
        df['Region'] = 'Bejaia'
    else:
        df['Region'][i] = 'Sidi Bel-abbes'
```

In [6]:
```
df
```

Out[6]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	R
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
239	26	09	2012	30	65	14	0	85.4	16	44.5	4.5	16.9	6.5	fire	
240	27	09	2012	28	87	15	4.4	41.1	6.5	8	0.1	6.2	0	not fire	
241	28	09	2012	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	not fire	
242	29	09	2012	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	not fire	
243	30	09	2012	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	not fire	

244 rows × 15 columns

3.3) Renaming the names of the columns

In [7]:
```
df.rename(columns={' RH': 'RH', ' Ws': 'Ws', 'Rain ': 'Rain', 'Classes  ':'Classes'},i
```

3.4) Stripping the classes features

In [8]:
```
df.Classes = df.Classes.str.strip()
df['Classes'].unique()
```

Out[8]:
```
array(['not fire', 'fire', nan], dtype=object)
```

In [9]:
```
df.head()
```

Out[9]:

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

4) Exploratory Data Analysis

4.1) Data Profiling

4.1.1) Shape of the dataset

In [10]:
```
df.shape
```

Out[10]:
```
(244, 15)
```

Observation

- This dataset comprises of 244 Rows and 15 Columns.

4.1.2) Columns of the dataset

In [11]:
```
df.columns
```

Out[11]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'Region'],
      dtype='object')
```

4.1.3) Checking missing values

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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
Region         0
dtype: int64
```

Observation

- One NuLL value in 'Classes' feature.

4.1.4) Fixing particular row from the dataset which is having the NuLL value

In [13]:
```
df.iloc[165]
```

Out[13]:
```
day                       14
month                     07
year                    2012
Temperature               37
RH                        37
Ws                        18
Rain                     0.2
FFMC                    88.9
DMC                     12.9
DC                    14.6 9
ISI                     12.5
BUI                     10.4
FWI                   fire  
Classes                  NaN
Region        Sidi Bel-abbes
Name: 165, dtype: object
```

Observation

- Some miss match positions.

4.1.5) Resetting missmatched position

In [14]:
```
df.at[165, 'DC'] = 14.6
df.at[165, 'ISI'] = 9
df.at[165, 'BUI'] = 12.5
df.at[165, 'FWI'] = 10.4
df.at[165, 'Classes'] = 'fire'
```

4.1.6) Again checking the info of that particular Row

In [15]:
```
df.iloc[165]
```

Out[15]:
```
day                       14
month                     07
year                    2012
Temperature               37
RH                        37
Ws                        18
Rain                     0.2
FFMC                    88.9
DMC                     12.9
DC                      14.6
ISI                        9
BUI                     12.5
FWI                     10.4
Classes                 fire
Region        Sidi Bel-abbes
Name: 165, dtype: object
```

4.1.7) Checking again missing values

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
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
Region         0
dtype: int64
```

Observation

- We have zero NuLL values.

4.1.8) Changing the datatypes of all columns

In [17]:
```
df['day']=df['day'].astype(int)
df['month']=df['month'].astype(int)
df['year']=df['year'].astype(int)
df['Temperature']=df['Temperature'].astype(int)
df['RH']=df['RH'].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['FWI']=df['FWI'].astype(float)
df['BUI']=df['BUI'].astype(float)
df['ISI']=df['ISI'].astype(float)
df['Ws']=df['Ws'].astype(float)
df['Region']=df['Region'].astype(object)
```

4.1.9) Adding new feature name 'Date' by replacing unecessary feature like 'day', 'month', 'year'

In [18]:
```
df['Date'] = pd.to_datetime(df[['day', 'month', 'year']])
df.drop(['day', 'month', 'year'],axis=1,inplace = True)
```

In [19]:
```
df.head()
```

Out[19]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region	Date
0	29	57	18.0	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	Bejaia	2012-06-01
1	29	61	13.0	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire	Bejaia	2012-06-02
2	26	82	22.0	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	Bejaia	2012-06-03
3	25	89	13.0	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire	Bejaia	2012-06-04
4	27	77	16.0	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	Bejaia	2012-06-05

4.1.10) Checking unique values

In [20]:
```
df['Classes'].unique()
```

Out[20]:
```
array(['not fire', 'fire'], dtype=object)
```

4.1.11) Check datatypes of the dataset

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temperature  244 non-null    int32
 1   RH           244 non-null    int32
 2   Ws           244 non-null    float64
 3   Rain         244 non-null    float64
 4   FFMC         244 non-null    float64
 5   DMC          244 non-null    float64
 6   DC           244 non-null    float64
 7   ISI          244 non-null    float64
 8   BUI          244 non-null    float64
 9   FWI          244 non-null    float64
 10  Classes      244 non-null    object
 11  Region       244 non-null    object
 12  Date         244 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(8), int32(2), object(2)
memory usage: 23.0+ KB
```

Observation

- 244 Rows and 15 Columns.
- No NuLL values.
- 4 datatypes int32, float64, object and datetime64.
- Datatypes included float64 = 8 Columns, int32 = 2 Columns, object = 2 Columns, datetime64 = 1 Columns.
- Total memory usage is 23.0+ KB

4.1.12) Checking memory usage by the dataset

In [22]:
```
df.memory_usage()
```

Out[22]:
```
Index           128
Temperature     976
RH              976
Ws             1952
Rain           1952
FFMC           1952
DMC            1952
DC             1952
ISI            1952
BUI            1952
FWI            1952
Classes        1952
Region         1952
Date           1952
dtype: int64
```

Observation

- Temperature and RH uses 976 Kb memory.
- Ws, Rain, FFMC, DMC, DC, ISI, BUI, FWI, Classes, Region and date uses 1952 Kb memory.
- Datatype is int64.

4.1.13) Numerical and Categorical features

(a) Numerical Data

In [23]:
```
# define numerical features
numerical_features = [feature for feature in df.columns if df[feature].dtype != 'O']

# print numerical features
print('We have {} numerical features : {}'.format(len(numerical_features), numerical_f
```

```
We have 11 numerical features : ['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'D
C', 'ISI', 'BUI', 'FWI', 'Date']
```

(b) Categorical Data

In [24]:
```
# define categorical features
categorical_features = [feature for feature in df.columns if df[feature].dtype == 'O']

# print categorical features
print('\n We have {} categorical features : {}'.format(len(categorical_features), cate
```

```
 We have 2 categorical features : ['Classes', 'Region']
```

4.2) Feature Information

(a) Weather data Components

- Temperature : 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

(b) Date Components

- day, month and year is merged into date.
- date displayed in (DD/MM/YYYY) format.

(c) FWI Components

- FFMC : Fine Fuel Moisture Code index the FWI system: 28.6 to 92.5
- DMC : Duff Moisture Code index from FWI system: 1.1 to 65.9
- ISI : Initial Spread Index from FWI system: 0 to 18.5
- BUI : Buildup Index from FWI index: 1.1 to 68
- DC : Drought Code index from FWI system: 7 to 220.4
- FWI : Fire Weather Index: 0 to 31.1
- Classes : Fire and not Fire
- Region : 1 for Bejaia region and 0 for Sidi Bel-abbes region

4.3) Univariate Analysis

- The term univariate analysis refers to the analysis of one variable prefix “uni” means “one.”
- The purpose of univariate analysis is to understand the distribution of values for a single variable.

In [25]:

```
df.var()
```

Out[25]:

```
Temperature      13.204817
RH              221.539415
Ws                7.897102
Rain              3.997623
FFMC            205.565939
DMC             152.968382
DC             2267.632245
ISI              17.260932
BUI             201.686818
FWI              55.180617
dtype: float64
```

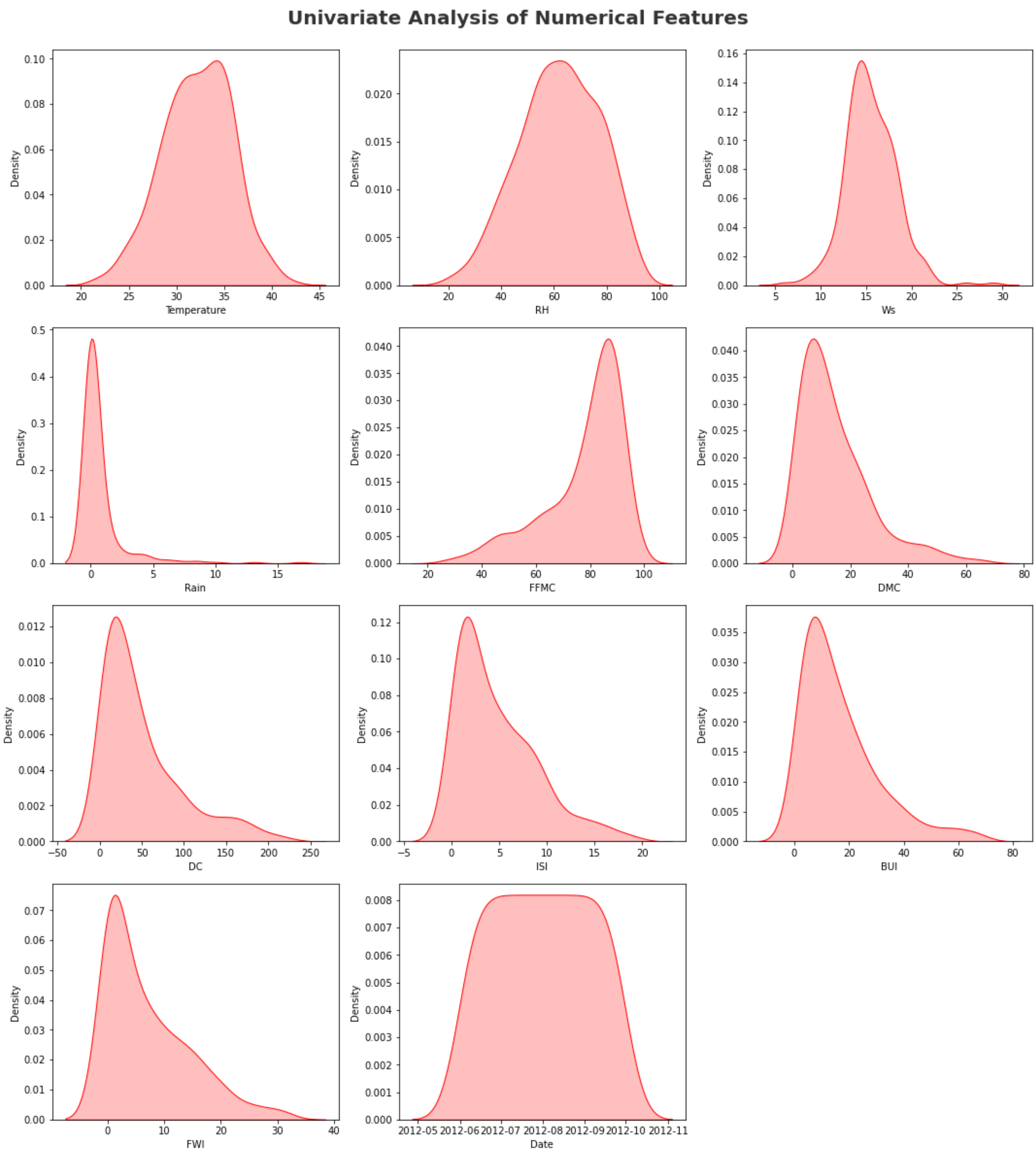
4.3.1) Numerical Features Analysis

kde plot

In [26]:

```
plt.figure(figsize=(15, 20))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20, fontweight='bol

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

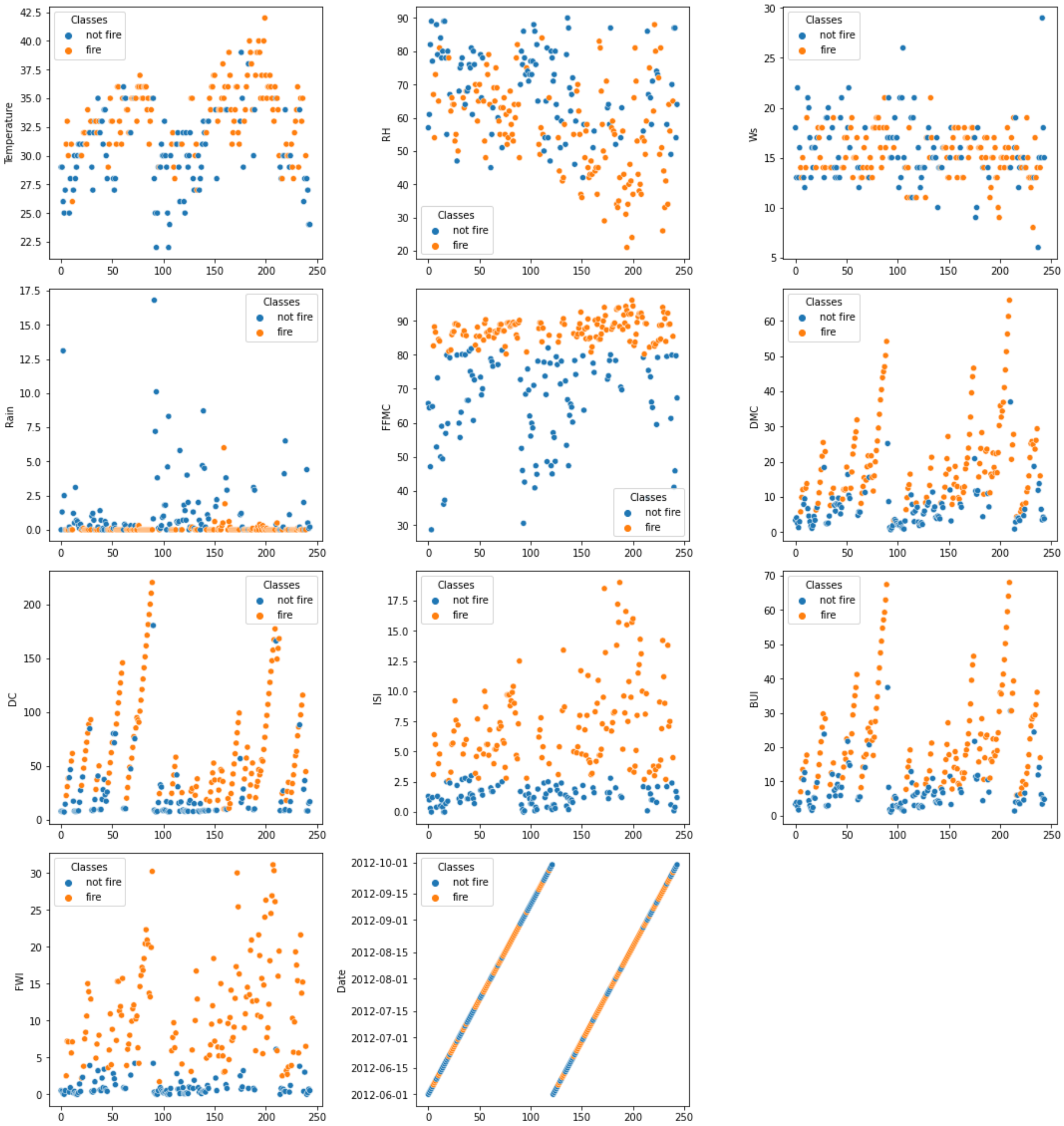


Scatter plot

In [27]:
```
plt.figure(figsize=(15, 20))
plt.suptitle('scatter plot with each numerical feature to explore feature', fontsize=2

for i in range(0, len(numerical_features)):
    plt.subplot(5, 3, i+1)
    sns.scatterplot(y=numerical_features[i], x=df.index, data=df,hue='Classes')
    plt.tight_layout()
```

scatter plot with each numerical feature to explore feature



Observation

- Rain,ISI,BUI,DMC are right skewed and postively skewed.
- FFMC is a Left skewed and Negetively skewed.
- Outliers in Rain, ISI, BUI, DMC and FFMC.

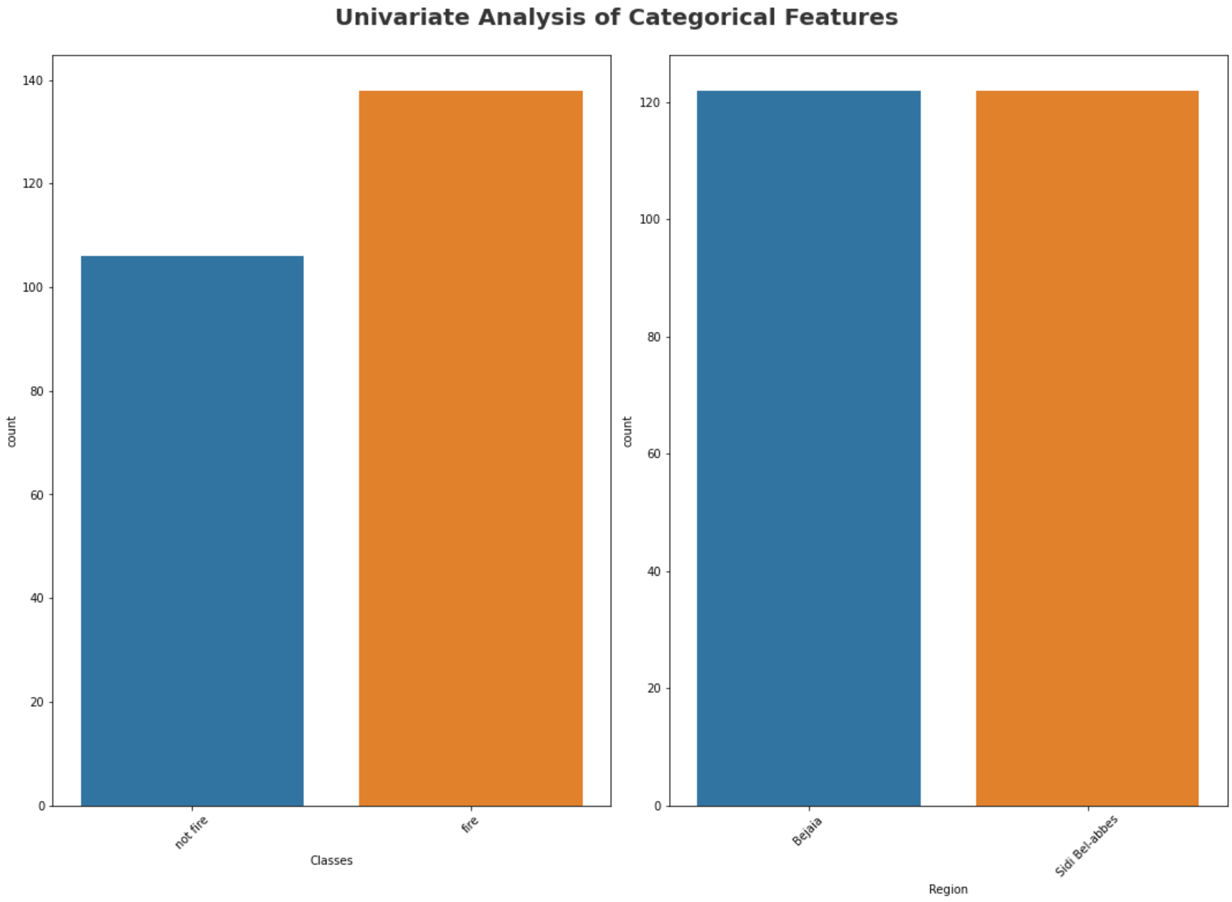
4.3.2) Categorical Features

Count plot

In [28]:

```
plt.figure(figsize=(15,20))
plt.suptitle('Univariate Analysis of Categorical Features', fontsize=20, fontweight='b
cat1 = ['Classes', 'Region']
for i in range(0, len(cat1)):
    plt.subplot(2, 2, i+1)
```

```
    sns.countplot(x=df[cat1[i]])
    plt.xlabel(cat1[i])
    plt.xticks(rotation=45)
    plt.tight_layout()
```



Observation

- Region Sidi Bel Abbes experiences more fire cases than Bejaia.
- Region Bejaia experiences less non fire cases than Sidi Bel Abbes.

Statistical Analysis

In [33]:
```
df.describe().T
```

Out[33]:

	count	mean	std	min	25%	50%	75%	max
Temperature	244.0	32.172131	3.633843	22.0	30.000	32.00	35.000	42.0
RH	244.0	61.938525	14.884200	21.0	52.000	63.00	73.250	90.0
Ws	244.0	15.504098	2.810178	6.0	14.000	15.00	17.000	29.0
Rain	244.0	0.760656	1.999406	0.0	0.000	0.00	0.500	16.8
FFMC	244.0	77.887705	14.337571	28.6	72.075	83.50	88.300	96.0
DMC	244.0	14.673361	12.368039	0.7	5.800	11.30	20.750	65.9
DC	244.0	49.288115	47.619662	6.9	13.275	33.10	68.150	220.4
ISI	244.0	4.759836	4.154628	0.0	1.400	3.50	7.300	19.0
BUI	244.0	16.673361	14.201648	1.1	6.000	12.45	22.525	68.0
FWI	244.0	7.049180	7.428366	0.0	0.700	4.45	11.375	31.1

Since DMC and BUI are highly coorelated, so dropping anyone won't harm the dataset.

In [34]:
```
df_new=df.drop(columns=('DMC'),axis=1)
```

In [35]:
```
df_new
```

Out[35]:

	Temperature	RH	Ws	Rain	FFMC	DC	ISI	BUI	FWI	Classes	Region	Date
0	29	57	18.0	0.0	65.7	7.6	1.3	3.4	0.5	not fire	Bejaia	2012-06-01
1	29	61	13.0	1.3	64.4	7.6	1.0	3.9	0.4	not fire	Bejaia	2012-06-02
2	26	82	22.0	13.1	47.1	7.1	0.3	2.7	0.1	not fire	Bejaia	2012-06-03
3	25	89	13.0	2.5	28.6	6.9	0.0	1.7	0.0	not fire	Bejaia	2012-06-04
4	27	77	16.0	0.0	64.8	14.2	1.2	3.9	0.5	not fire	Bejaia	2012-06-05
...	...	...	...	...	...	...	...	...	...	...	...	...
239	30	65	14.0	0.0	85.4	44.5	4.5	16.9	6.5	fire	Sidi Bel-abbes	2012-09-26
240	28	87	15.0	4.4	41.1	8.0	0.1	6.2	0.0	not fire	Sidi Bel-abbes	2012-09-27
241	27	87	29.0	0.5	45.9	7.9	0.4	3.4	0.2	not fire	Sidi Bel-abbes	2012-09-28
242	24	54	18.0	0.1	79.7	15.2	1.7	5.1	0.7	not fire	Sidi Bel-abbes	2012-09-29
243	24	64	15.0	0.2	67.3	16.5	1.2	4.8	0.5	not fire	Sidi Bel-abbes	2012-09-30

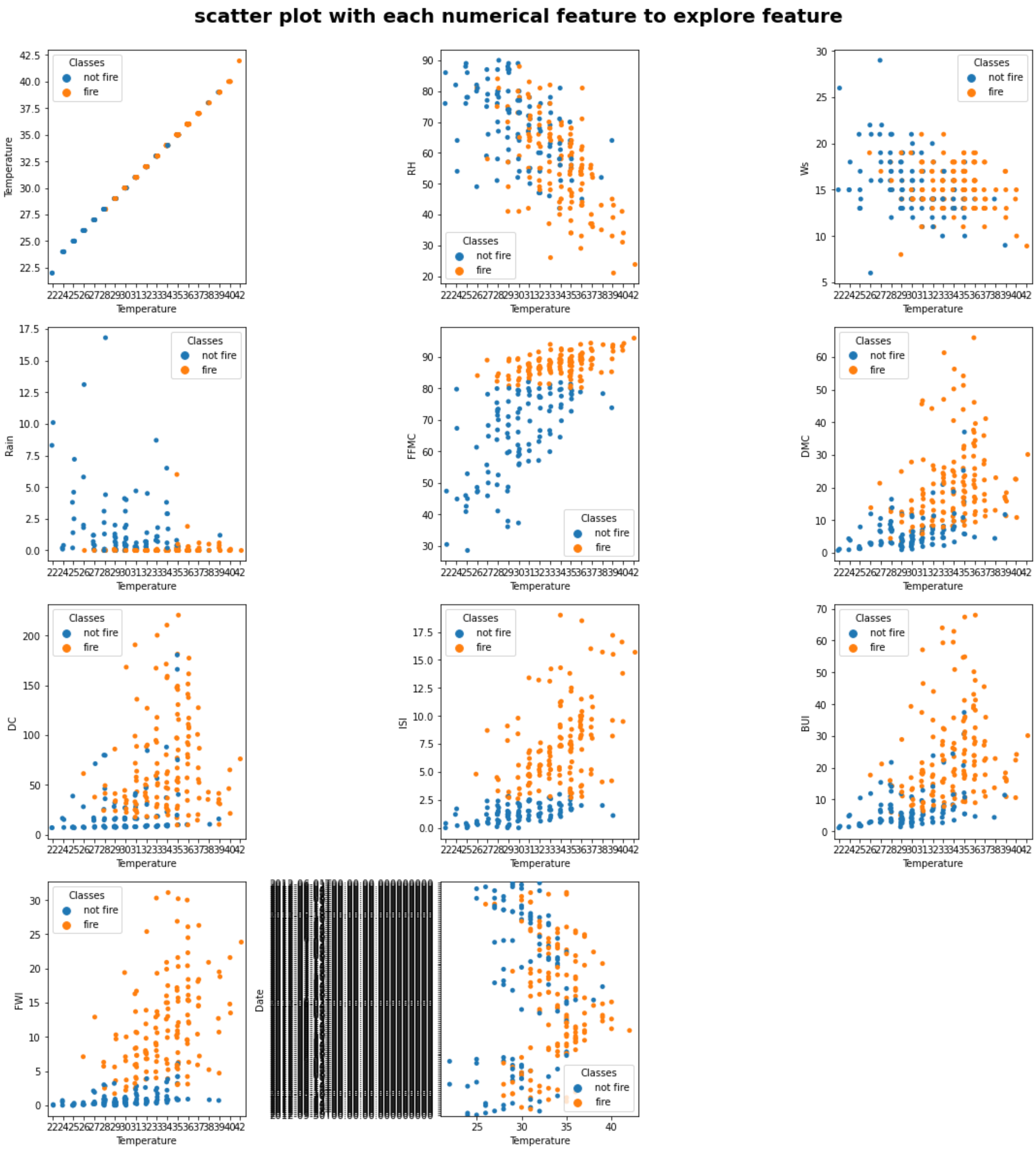
244 rows × 12 columns

Target feature is Temperature

So, analyzing each feature with respect to Temperature

In [36]:
```
plt.figure(figsize=(15, 20))
plt.suptitle('scatter plot with each numerical feature to explore feature', fontsize=2

for i in range(0, len(numerical_features)):
    plt.subplot(5, 3, i+1)
    sns.stripplot(y=numerical_features[i], x='Temperature', data=df,hue='Classes')
    plt.tight_layout()
```

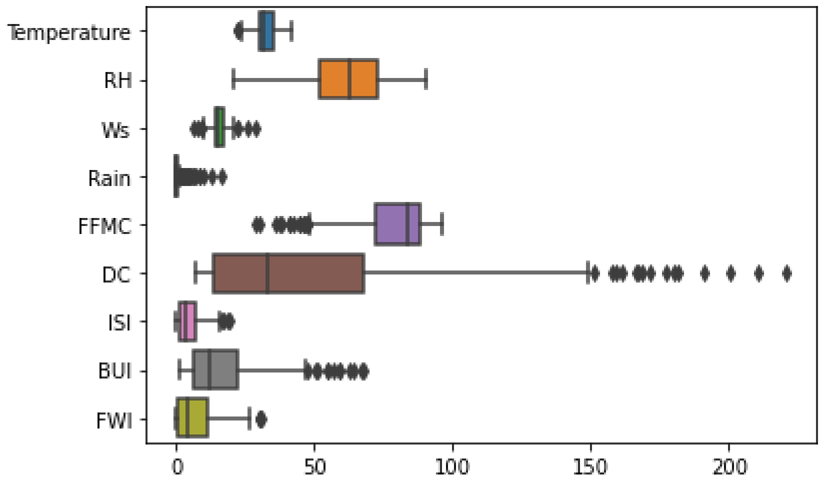


Box plot

For finding outliers

In [37]:
```
sns.boxplot(data=df_new,orient='h')
```

Out[37]: <AxesSubplot:>



Observation

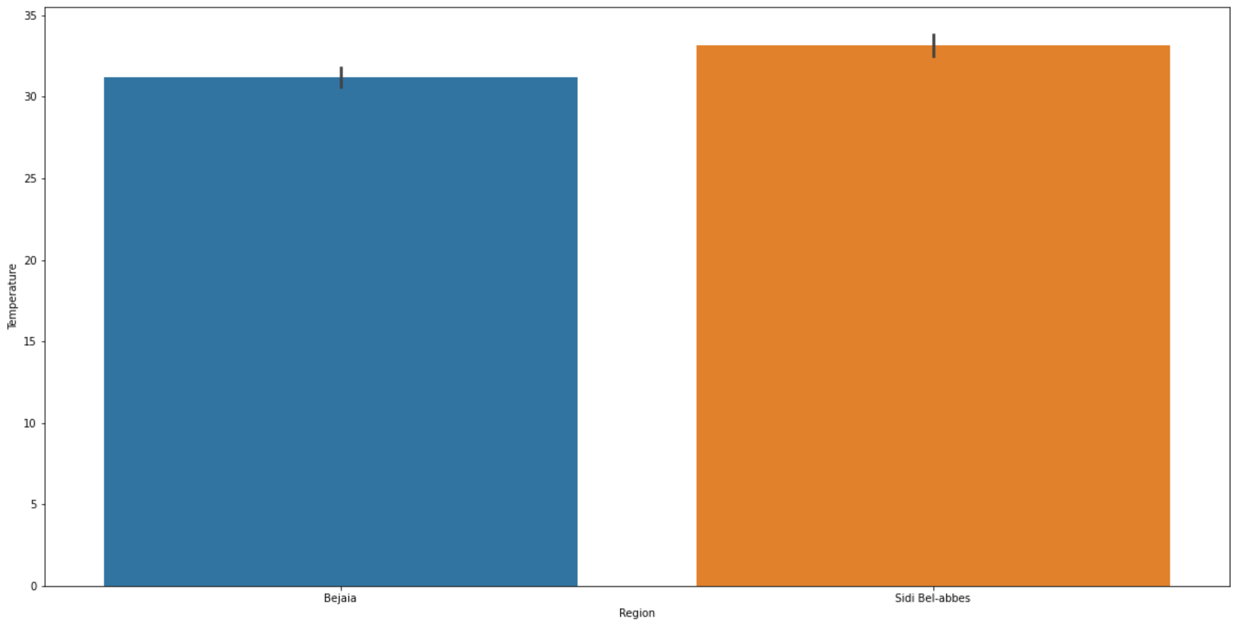
- All features except RH have outliers.

Graphical Analysis

Temperature w.r.t Region

In [38]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)
sns.barplot(x='Region', y='Temperature', data=df)
```

Out[38]: <AxesSubplot:xlabel='Region', ylabel='Temperature'>

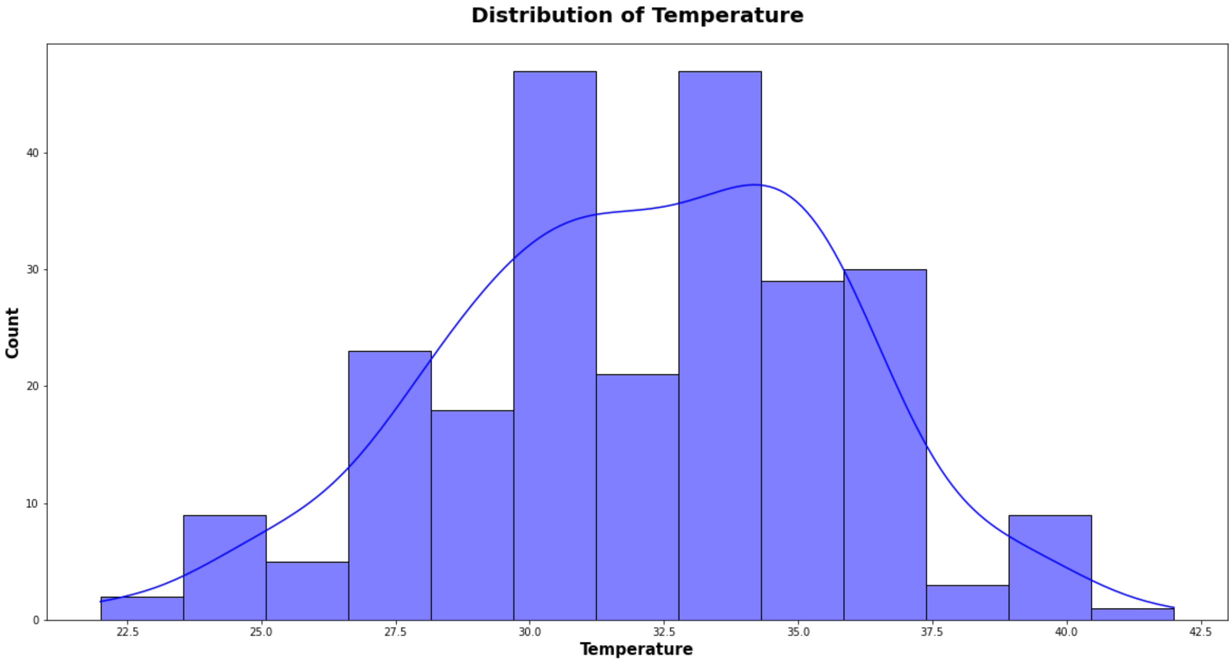


Observation

- Sidi bel Abbas region experiences most of the time High Temperature.

Temperature w.r.t Temperature range

In [39]:
```
plt.subplots(figsize=(20,10))
sns.histplot("Distribution of Temperature",x=df.Temperature,color='b',kde=True)
plt.title("Distribution of Temperature",weight='bold',fontsize=20,pad=20)
plt.xlabel("Temperature",weight='bold',fontsize=15)
plt.ylabel("Count",weight='bold',fontsize=15)
plt.show()
```



Observation

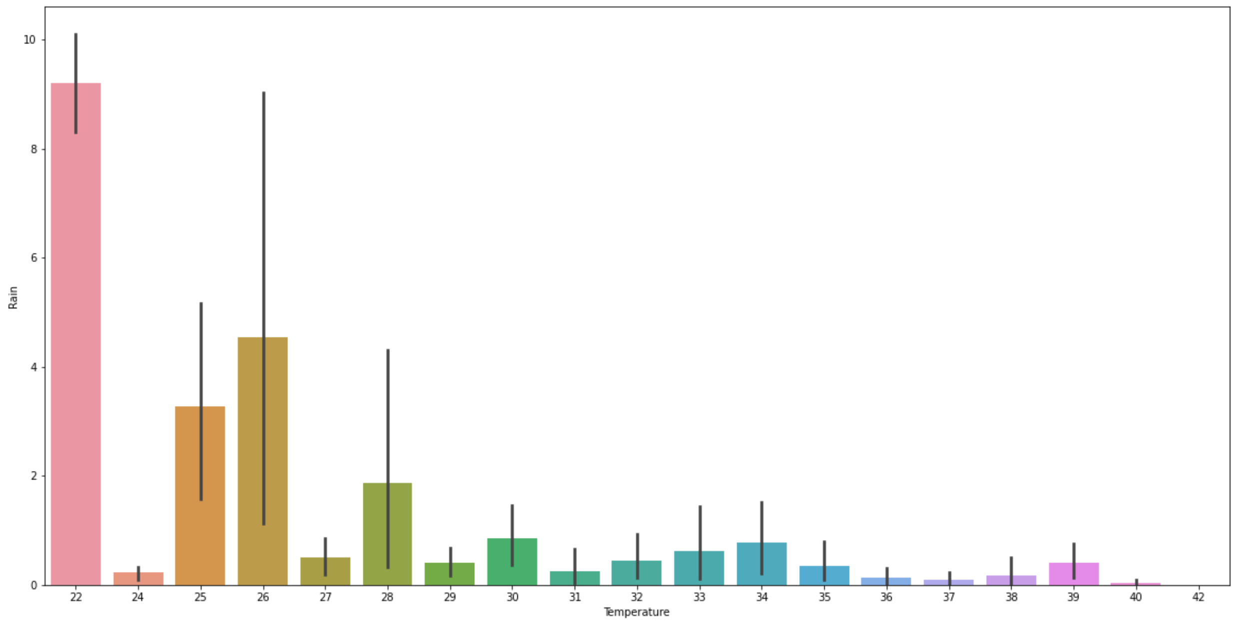
- Temperature range is 30 - 35 degree celcius.

Temperature w.r.t Rain

In [44]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

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

Out[44]: <AxesSubplot:xlabel='Temperature', ylabel='Rain'>



Observation

- When Rain is high, Temperature is low and vice-versa.

4.4) Multivariate Analysis

Multicollinearity in Numerical features

In [51]:
```
df.cov()
```

Out[51]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC
Temperature	13.204817	-35.396782	-2.840215	-2.374270	35.297598	21.712423	64.111931
RH	-35.396782	221.539415	9.874739	6.635431	-137.785533	-74.580245	-156.165754
Ws	-2.840215	9.874739	7.897102	0.956129	-6.577727	-0.043306	10.203135
Rain	-2.374270	6.635431	0.956129	3.997623	-15.595918	-7.135415	-28.258988
FFMC	35.297598	-137.785533	-6.577727	-15.595918	205.565939	106.820535	344.044709
DMC	21.712423	-74.580245	-0.043306	-7.135415	106.820535	152.968382	515.552604
DC	64.111931	-156.165754	10.203135	-28.258988	344.044709	515.552604	2267.632245
ISI	9.148506	-42.561327	0.142964	-2.889611	44.124525	34.856991	99.696270
BUI	23.553987	-73.916459	1.209369	-8.501670	120.185186	172.521016	636.831657
FWI	15.300965	-64.178446	0.708851	-4.823366	73.640607	80.407612	260.718118

In [50]:
```
df.corr()
```

Out[50]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
Temperature	1.000000	-0.654443	-0.278132	-0.326786	0.677491	0.483105	0.370498	0.605971
RH	-0.654443	1.000000	0.236084	0.222968	-0.645658	-0.405133	-0.220330	-0.688268
Ws	-0.278132	0.236084	1.000000	0.170169	-0.163255	-0.001246	0.076245	0.012245
Rain	-0.326786	0.222968	0.170169	1.000000	-0.544045	-0.288548	-0.296804	-0.347862
FFMC	0.677491	-0.645658	-0.163255	-0.544045	1.000000	0.602391	0.503910	0.740751
DMC	0.483105	-0.405133	-0.001246	-0.288548	0.602391	1.000000	0.875358	0.678355
DC	0.370498	-0.220330	0.076245	-0.296804	0.503910	0.875358	1.000000	0.503919
ISI	0.605971	-0.688268	0.012245	-0.347862	0.740751	0.678355	0.503919	1.000000
BUI	0.456415	-0.349685	0.030303	-0.299409	0.590251	0.982206	0.941672	0.641351
FWI	0.566839	-0.580457	0.033957	-0.324755	0.691430	0.875191	0.737041	0.922422

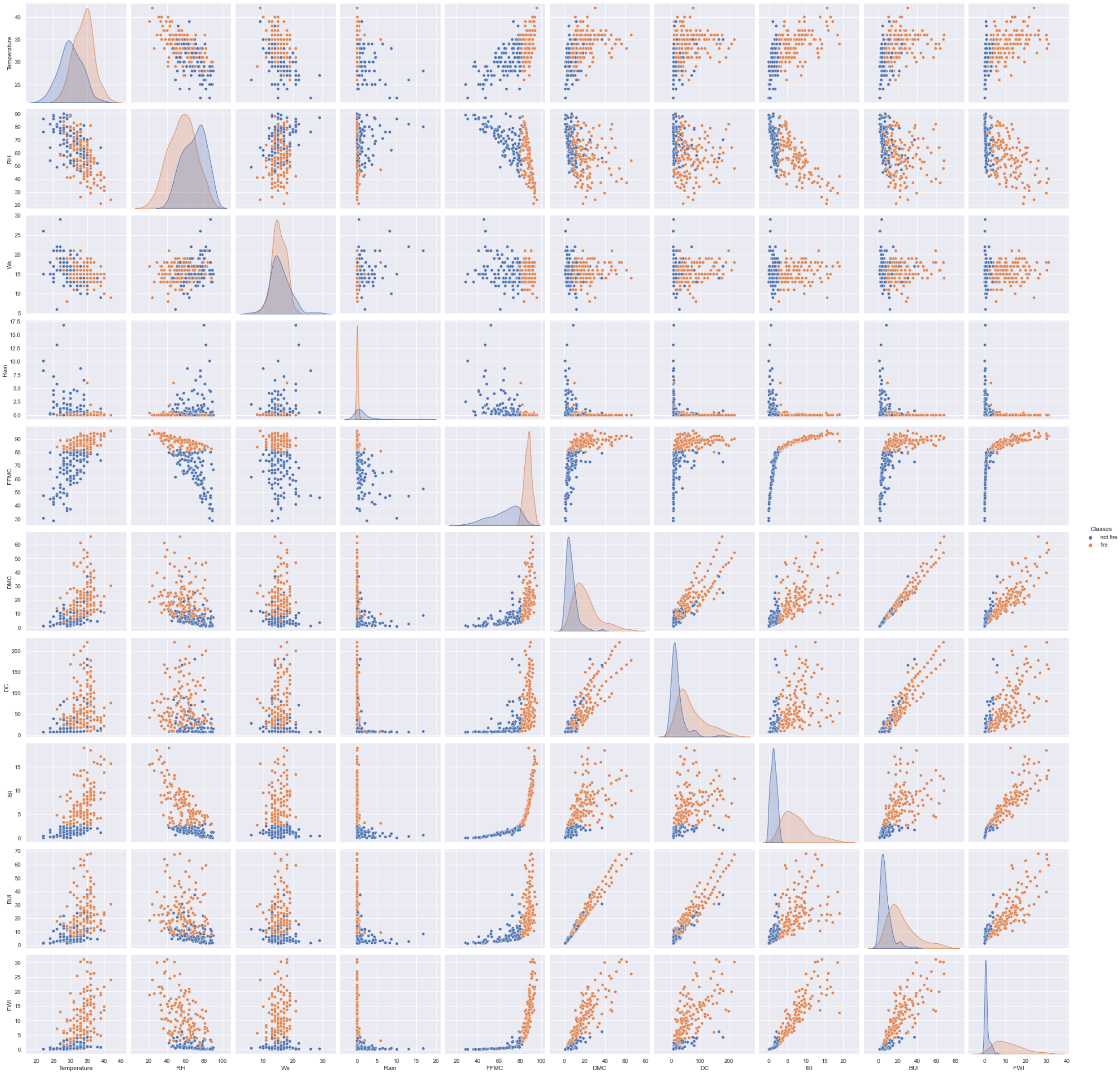
Pair plot

In [48]:

```
sns.pairplot(df,hue='Classes',size=3)
```

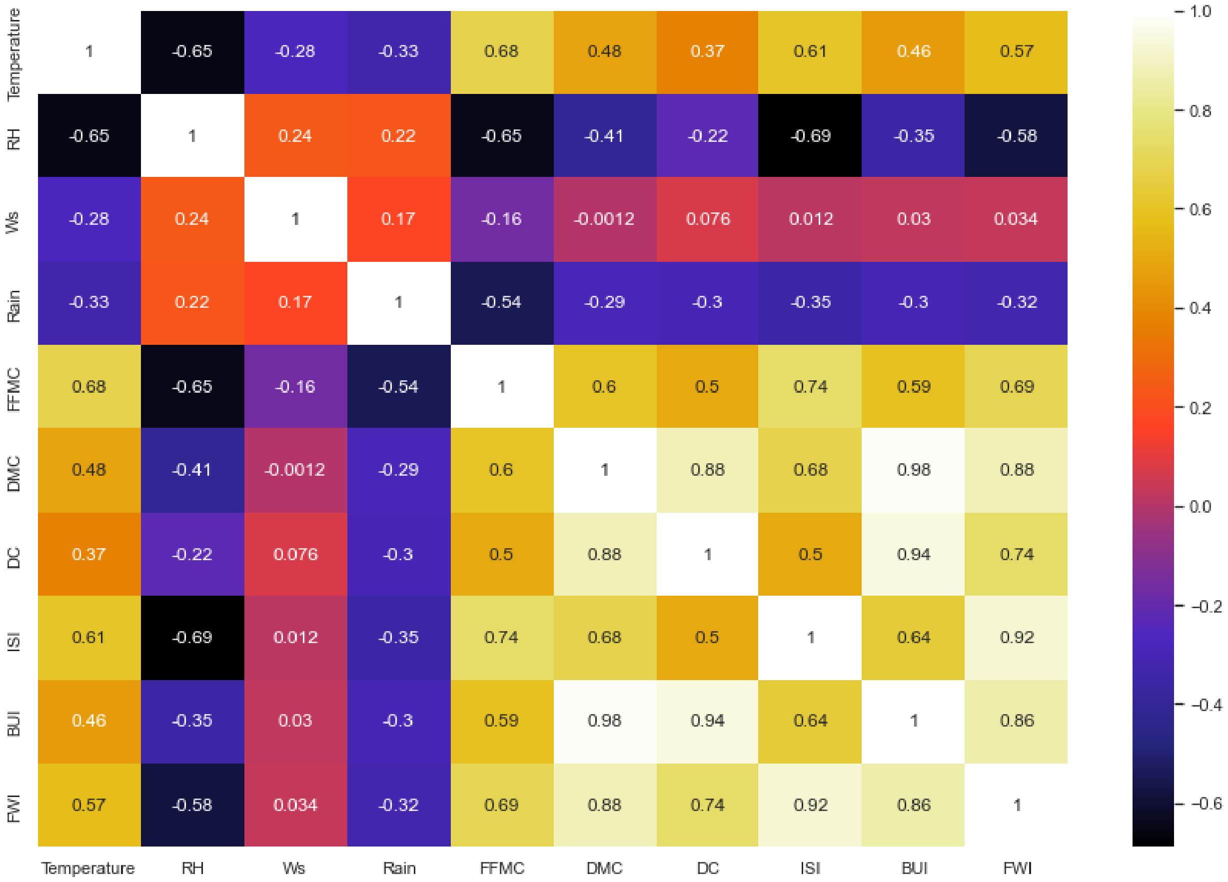
Out[48]:

```
<seaborn.axisgrid.PairGrid at 0x28c35a87850>
```



Heat map

In [49]:
```
plt.figure(figsize = (15,10))
sns.heatmap(df.corr(), cmap="CMRmap", annot=True)
plt.show()
```

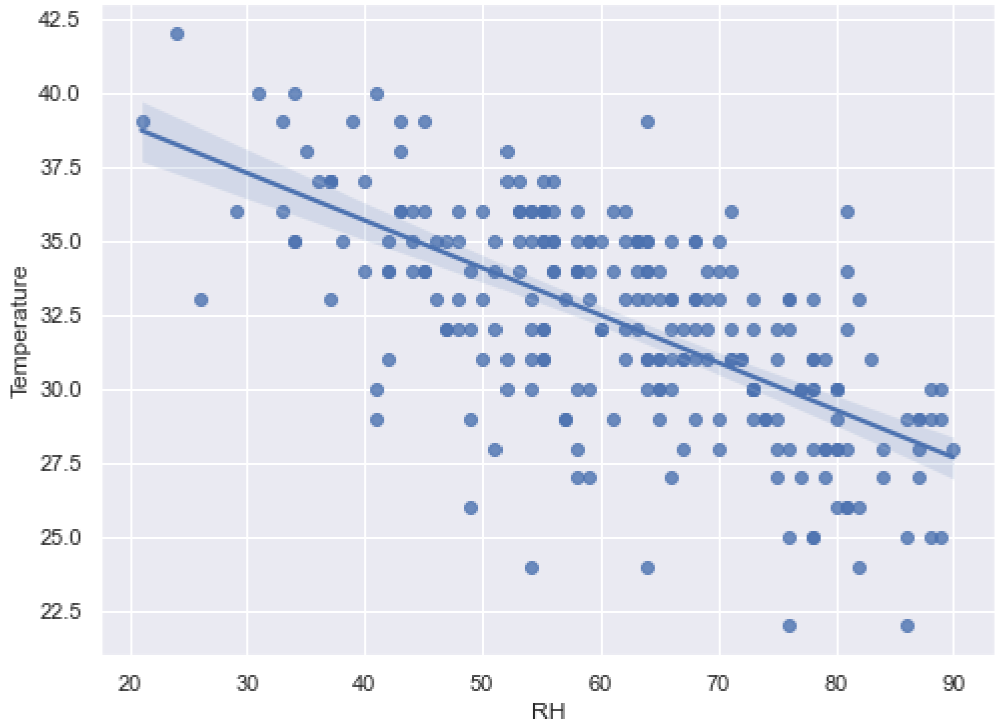


observation

- Highly +ve correlated features are DMC and BUI - 'White color'
- Highly -ve correlated features are RH and Temp, RH and FFMC, RH and ISI - 'Black color'

In [45]:
```
sns.set(rc={'figure.figsize':(8,6)})
sns.regplot(x = "RH", y = "Temperature", data = df)
```

Out[45]: <AxesSubplot:xlabel='RH', ylabel='Temperature'>



Independent and Dependent Features

In [41]:
```
x=df_new.iloc[:,1:-3]
y=df_new.iloc[:,0]
```

In [42]:
```
x
```

Out[42]:

	RH	Ws	Rain	FFMC	DC	ISI	BUI	FWI
0	57	18.0	0.0	65.7	7.6	1.3	3.4	0.5
1	61	13.0	1.3	64.4	7.6	1.0	3.9	0.4
2	82	22.0	13.1	47.1	7.1	0.3	2.7	0.1
3	89	13.0	2.5	28.6	6.9	0.0	1.7	0.0
4	77	16.0	0.0	64.8	14.2	1.2	3.9	0.5
...	...	...	...	...	...	...	...	...
239	65	14.0	0.0	85.4	44.5	4.5	16.9	6.5
240	87	15.0	4.4	41.1	8.0	0.1	6.2	0.0
241	87	29.0	0.5	45.9	7.9	0.4	3.4	0.2
242	54	18.0	0.1	79.7	15.2	1.7	5.1	0.7
243	64	15.0	0.2	67.3	16.5	1.2	4.8	0.5

244 rows × 8 columns

In [43]:
```
y
```

Out[43]:
```
0      29
1      29
2      26
3      25
4      27
       ..
239    30
240    28
241    27
242    24
243    24
Name: Temperature, Length: 244, dtype: int32
```

Splitting training and testing data

In [52]:
```
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
```

independent training data

In [53]:
```
x_train
```

Out[53]:

	RH	Ws	Rain	FFMC	DC	ISI	BUI	FWI
237	49	6.0	2.0	61.3	28.1	0.6	11.9	0.4
78	54	18.0	0.0	89.4	110.9	9.7	27.5	16.1
25	64	18.0	0.0	86.8	71.8	6.7	21.6	10.6
124	80	14.0	2.0	48.7	7.6	0.3	2.6	0.1
176	64	9.0	1.2	73.8	15.9	1.1	11.4	0.7
...	...	...	...	...	...	...	...	...
64	69	13.0	0.0	85.0	19.8	4.0	8.2	3.9
15	89	13.0	0.7	36.1	7.6	0.0	2.2	0.0
228	51	13.0	0.0	88.7	50.2	6.9	17.8	9.8
125	64	14.0	0.0	79.4	15.4	2.2	5.6	1.0
9	79	12.0	0.0	73.2	46.3	1.3	12.6	0.9

163 rows × 8 columns

independent testing data

In [54]:
```
x_test
```

Out[54]:

	RH	Ws	Rain	FFMC	DC	ISI	BUI	FWI
162	56	15.0	2.9	74.8	9.5	1.6	6.8	0.8
60	64	17.0	0.0	87.2	145.7	6.8	41.2	15.7
61	45	14.0	0.0	78.8	10.2	2.0	4.7	0.9
63	63	14.0	0.3	76.6	10.0	1.7	5.5	0.8
69	59	17.0	0.0	87.4	57.0	6.9	17.9	9.9
...	...	...	...	...	...	...	...	...
169	68	15.0	0.0	86.1	51.6	5.2	23.9	9.1
232	41	8.0	0.1	83.9	86.0	2.7	28.9	5.6
144	59	16.0	0.8	74.2	8.3	1.6	6.7	0.8
208	37	16.0	0.0	92.2	167.2	13.1	64.0	30.3
105	76	26.0	8.3	47.4	7.0	0.4	1.6	0.1

81 rows × 8 columns

dependent training data

In [55]:

```
y_train
```

Out[55]:

```
237    26
78     36
25     31
124    29
176    39
       ..
64     34
15     29
228    32
125    30
9      28
Name: Temperature, Length: 163, dtype: int32
```

dependent testing data

In [57]:

```
y_test
```

Out[57]:

```
162    34
60     35
61     36
63     35
69     35
       ..
169    33
232    29
144    33
208    33
105    22
Name: Temperature, Length: 81, dtype: int32
```

In [58]:

```
x_train.shape
```

Out[58]:

```
(163, 8)
```

In [59]:
```
x_test.shape
```
Out[59]:
```
(81, 8)
```
In [60]:
```
y_train.shape
```
Out[60]:
```
(163,)
```
In [61]:
```
y_test.shape
```
Out[61]:
```
(81,)
```

Standardization and Feature Scaling the dataset

In [62]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```
In [63]:
```
scaler
```
Out[63]:
```
▾ StandardScaler
StandardScaler()
```
In [65]:
```
x_train=scaler.fit_transform(x_train)
```
In [66]:
```
x_test=scaler.transform(x_test)
```
In [67]:
```
x_train
```
Out[67]:
```
array([[-0.85631108, -3.36419461,  0.88853946, ..., -0.97156746,
        -0.32636097, -0.86597829],
       [-0.52508491,  0.99944243, -0.441414  , ...,  1.19293541,
         0.76499972,  1.21371864],
       [ 0.13736742,  0.99944243, -0.441414  , ...,  0.47936304,
         0.35224151,  0.48516239],
       ...,
       [-0.72382061, -0.81873967, -0.441414  , ...,  0.52693453,
         0.08639724,  0.37919057],
       [ 0.13736742, -0.45510325, -0.441414  , ..., -0.59099552,
        -0.76710278, -0.78649943],
       [ 1.13104591, -1.18237609, -0.441414  , ..., -0.80506723,
        -0.27738965, -0.7997459 ]])
```
In [68]:
```
x_test
```

Out[68]:
```
array([[-3.92594448e-01, -9.14668296e-02,  1.48701853e+00,
        -1.82411230e-01, -8.21325921e-01, -7.33709998e-01,
        -6.83151962e-01, -8.12992382e-01],
       [ 1.37367416e-01,  6.35806011e-01, -4.41414004e-01,
         6.64566895e-01,  2.03374779e+00,  5.03148781e-01,
         1.72343828e+00,  1.16073273e+00],
       [-1.12129201e+00, -4.55103250e-01, -4.41414004e-01,
         9.08075201e-02, -8.06652268e-01, -6.38567015e-01,
        -8.30065901e-01, -7.99745905e-01],
       [ 7.11221826e-02, -4.55103250e-01, -2.41920984e-01,
        -5.94627923e-02, -8.10844740e-01, -7.09924252e-01,
        -7.74098686e-01, -8.12992382e-01],
       [-1.93858749e-01,  6.35806011e-01, -4.41414004e-01,
         6.78227832e-01,  1.74386276e-01,  5.26934527e-01,
         9.33931438e-02,  3.92437047e-01],
       [-5.91330147e-01,  2.72169591e-01, -4.41414004e-01,
         8.21667676e-01,  2.36495311e+00,  1.35943563e+00,
         2.16418009e+00,  2.03500022e+00],
       [-5.91330147e-01,  1.36307885e+00, -4.41414004e-01,
         8.01176270e-01,  1.04642054e+00,  1.26429264e+00,
         5.13147255e-01,  1.10774682e+00],
       [-5.91330147e-01,  6.35806011e-01, -1.08925637e-01,
         1.86434082e-01,  2.10711606e+00, -4.72066794e-01,
         9.81872679e-01, -1.37422043e-01],
       [ 5.34838813e-01, -4.55103250e-01, -4.41414004e-01,
         3.64026270e-01, -3.05651815e-01, -3.53138066e-01,
        -3.82328182e-01, -4.42091020e-01],
       [ 7.11221826e-02, -4.55103250e-01, -4.41414004e-01,
         6.50905957e-01, -2.44860965e-01,  2.17719832e-01,
        -2.84385556e-01, -1.82037479e-02],
       [-3.92594448e-01,  2.72169591e-01, -4.41414004e-01,
         7.80684863e-01,  1.76482512e-01,  8.36149222e-01,
         5.06151353e-01,  8.29570795e-01],
       [ 9.98555444e-01,  2.09035169e+00,  7.55544118e-01,
        -1.29577764e+00, -8.44384520e-01, -8.52638726e-01,
        -9.90971643e-01, -8.79224768e-01],
       [-1.84998957e+00,  2.72169591e-01, -3.08418657e-01,
         7.39702051e-01, -7.50658327e-02,  6.69649001e-01,
         6.54095363e-02,  4.71915910e-01],
       [ 6.01084046e-01, -4.55103250e-01,  3.88093477e+00,
        -8.85949510e-01, -8.29710866e-01, -8.76424472e-01,
        -9.14016723e-01, -8.65978291e-01],
       [-9.88801544e-01,  9.99442431e-01,  3.54844640e+00,
         2.27416895e-01, -8.17133449e-01, -3.76923811e-01,
        -5.01258514e-01, -5.21569883e-01],
       [ 5.34838813e-01,  2.72169591e-01, -4.41414004e-01,
         5.82601270e-01, -4.83831892e-01,  1.70148341e-01,
        -2.70393753e-01, -3.14502251e-02],
       [-5.25084914e-01, -1.54601251e+00, -3.74916331e-01,
         4.25500489e-01, -4.69158239e-01, -3.76923811e-01,
        -5.08254416e-01, -5.08323406e-01],
       [ 1.06480068e+00, -4.55103250e-01,  4.89553424e-01,
        -2.21789092e+00, -8.63250645e-01, -1.06671044e+00,
        -9.90971643e-01, -9.05717723e-01],
       [-1.78374434e+00, -9.14668296e-02, -4.41414004e-01,
         1.11537783e+00, -1.25375502e-01,  2.62008015e+00,
         4.43188236e-01,  1.84954954e+00],
       [-7.90065846e-01,  2.72169591e-01, -4.41414004e-01,
         8.48989551e-01,  4.67859344e-01,  1.14536392e+00,
         1.12179072e+00,  1.37267636e+00],
       [ 2.03612648e-01, -4.55103250e-01, -4.41414004e-01,
         4.80144239e-01,  1.17787898e-01, -1.62852100e-01,
        -5.35207952e-02, -1.77161475e-01],
       [-1.93858749e-01,  1.36307885e+00, -4.41414004e-01,
```

```
         7.94345801e-01,  2.50540093e+00,  1.21672115e+00,
         1.59051614e+00,  1.65085238e+00],
       [-5.91330147e-01,  9.99442431e-01, -4.41414004e-01,
         7.87515332e-01,  6.62809311e-01,  1.07400668e+00,
         6.46069390e-01,  1.06800739e+00],
       [ 4.87694966e-03,  2.72169591e-01, -4.41414004e-01,
         7.05549707e-01, -2.97266870e-01,  5.50720273e-01,
        -1.15453840e-02,  3.39451138e-01],
       [-1.91623481e+00,  6.35806011e-01, -4.41414004e-01,
         1.10854736e+00, -3.47576539e-01,  2.97686634e+00,
         2.34341252e-02,  1.66409886e+00],
       [ 1.85974347e+00, -9.14668296e-02, -4.41414004e-01,
        -7.28848729e-01, -7.12321639e-01, -8.28852981e-01,
        -6.62164256e-01, -8.39485337e-01],
       [-6.13682833e-02,  9.99442431e-01, -2.41920984e-01,
         1.86434082e-01,  8.74529168e-01, -4.48281049e-01,
         7.24054382e-02, -3.62612156e-01],
       [-9.88801544e-01, -4.55103250e-01,  2.40697100e-02,
         2.01142638e-03, -8.35999575e-01, -6.86138506e-01,
        -6.83151962e-01, -7.99745905e-01],
       [ 1.19729114e+00,  2.72169591e-01,  7.55544118e-01,
        -2.05395967e+00, -8.59058173e-01, -1.04292469e+00,
        -9.48996232e-01, -9.05717723e-01],
       [ 1.13104591e+00, -9.14668296e-02, -4.41414004e-01,
         5.41618457e-01,  1.83041288e+00,  3.64812051e-03,
         1.45759401e+00,  4.98408864e-01],
       [ 1.26353638e+00,  2.09035169e+00,  3.41545106e+00,
        -1.97199404e+00, -8.59058173e-01, -1.01913895e+00,
        -9.48996232e-01, -9.05717723e-01],
       [ 1.59476254e+00,  2.72169591e-01, -4.41414004e-01,
        -2.70285420e+00, -8.48576992e-01, -1.09049618e+00,
        -1.06093066e+00, -9.18964200e-01],
       [-5.25084914e-01, -8.18739670e-01, -4.41414004e-01,
         7.32871582e-01, -3.81116318e-01,  4.08005798e-01,
        -3.96319986e-01,  3.47821611e-02],
       [-7.23820613e-01,  2.72169591e-01,  2.08549759e+00,
         2.01142638e-03, -8.21325921e-01, -6.38567015e-01,
        -6.20188845e-01, -7.46759996e-01],
       [ 1.37367416e-01, -9.14668296e-02, -4.41414004e-01,
         6.30414551e-01,  3.16930338e-01,  2.41505578e-01,
         1.21376751e-01,  1.93739888e-01],
       [ 3.36103114e-01,  1.36307885e+00, -4.41414004e-01,
        -1.41428417e-01, -6.78781859e-01, -6.38567015e-01,
        -8.79037214e-01, -8.12992382e-01],
       [ 1.19729114e+00,  2.72169591e-01, -1.75423310e-01,
        -1.20698154e+00, -4.52388349e-01, -9.00210218e-01,
        -8.02082293e-01, -8.65978291e-01],
       [ 4.87694966e-03, -9.14668296e-02, -4.41414004e-01,
         3.98178614e-01, -6.49434553e-01, -2.10423591e-01,
        -7.11135569e-01, -4.95076928e-01],
       [ 1.32978161e+00, -9.14668296e-02, -1.75423310e-01,
        -2.22472138e+00, -8.67443118e-01, -1.06671044e+00,
        -1.06093066e+00, -9.18964200e-01],
       [-9.22556312e-01, -8.18739670e-01, -4.41414004e-01,
         8.76311426e-01,  1.25394792e+00,  9.55077950e-01,
         8.97921857e-01,  1.10774682e+00],
       [ 4.02348347e-01,  1.36307885e+00, -4.41414004e-01,
         7.39702051e-01,  1.71721612e+00,  9.78863696e-01,
         1.26870466e+00,  1.30644398e+00],
       [-4.58839681e-01, -9.14668296e-02, -4.41414004e-01,
         8.08006738e-01,  3.88202368e-01,  8.59934967e-01,
         8.20966936e-01,  1.00177500e+00],
       [-1.32002771e+00,  2.09035169e+00, -4.41414004e-01,
         8.96802832e-01, -3.81116318e-01,  2.07300800e+00,
```

```
         1.00389046e-01,  1.29319750e+00],
       [ 8.66064978e-01,  2.72169591e-01, -4.41414004e-01,
         3.23043457e-01, -5.11082963e-01, -3.29352320e-01,
        -7.39119177e-01, -5.87802269e-01],
       [-1.38627294e+00, -1.90964893e+00, -3.74916331e-01,
         9.92429394e-01,  3.44181408e-01,  1.14536392e+00,
         5.34134960e-01,  1.04151443e+00],
       [-4.58839681e-01, -4.55103250e-01, -4.41414004e-01,
         7.80684863e-01,  3.84009896e-01,  6.45863256e-01,
         3.73229218e-01,  6.17627160e-01],
       [ 1.19729114e+00,  2.09035169e+00,  1.07301951e+01,
        -1.70560576e+00, -8.38095811e-01, -9.71567455e-01,
        -5.78213434e-01, -8.79224768e-01],
       [-1.18753724e+00, -8.18739670e-01, -4.41414004e-01,
         8.62650488e-01, -6.13798537e-01,  8.59934967e-01,
        -2.84385556e-01,  3.52697615e-01],
       [-4.58839681e-01, -9.14668296e-02, -4.41414004e-01,
         7.94345801e-01, -1.12798084e-01,  7.88577730e-01,
         2.96274297e-01,  6.70613069e-01],
       [ 1.72725301e+00, -8.18739670e-01, -4.41414004e-01,
        -4.41969042e-01, -6.72493151e-01, -8.28852981e-01,
        -9.00024919e-01, -8.52731814e-01],
       [ 1.06480068e+00, -4.55103250e-01, -4.41414004e-01,
         2.41077832e-01, -3.58057720e-01, -4.95852540e-01,
        -5.71217532e-01, -6.27541701e-01],
       [ 4.02348347e-01, -4.55103250e-01, -4.41414004e-01,
         5.41618457e-01, -1.16990557e-01, -2.01376252e-02,
        -1.65455225e-01, -1.24175566e-01],
       [-1.25378248e+00,  2.72169591e-01, -4.41414004e-01,
         9.37785644e-01,  1.86604889e+00,  1.62107883e+00,
         2.35306944e+00,  2.32642272e+00],
       [ 3.36103114e-01, -4.55103250e-01, -4.41414004e-01,
         3.50365332e-01, -5.55103923e-01, -3.76923811e-01,
        -6.69160158e-01, -5.87802269e-01],
       [ 1.46227207e+00,  9.99442431e-01, -4.41414004e-01,
         4.32330957e-01,  1.29760879e-02, -4.39233710e-02,
        -3.95289915e-02, -8.44361341e-02],
       [ 6.01084046e-01,  6.35806011e-01, -2.41920984e-01,
        -5.37595604e-01, -3.89501263e-01, -7.57495743e-01,
        -8.02082293e-01, -8.39485337e-01],
       [ 2.69857881e-01, -9.14668296e-02, -3.74916331e-01,
         3.57195801e-01,  1.00868828e+00, -3.29352320e-01,
         1.32467187e+00,  1.01014547e-01],
       [-1.65125387e+00, -8.18739670e-01, -4.41414004e-01,
         1.02658174e+00,  7.79594102e-02,  1.66865032e+00,
         7.37016114e-01,  1.51838761e+00],
       [ 7.11221826e-02,  6.35806011e-01,  2.90060404e-01,
        -3.19020605e-01,  1.66001331e-01, -7.33709998e-01,
         3.59237414e-01, -5.87802269e-01],
       [-8.56311079e-01,  1.36307885e+00, -4.41414004e-01,
         7.60193457e-01, -3.20325469e-01,  1.05022093e+00,
        -2.91381458e-01,  4.45422956e-01],
       [ 2.03612648e-01, -9.14668296e-02, -3.74916331e-01,
         2.68399707e-01,  2.81294322e-01, -4.48281049e-01,
        -4.54948219e-03, -3.89105111e-01],
       [ 7.33574512e-01, -9.14668296e-02, -4.41414004e-01,
         6.23584082e-01,  1.64175162e+00,  2.17719832e-01,
         1.28969236e+00,  6.57366591e-01],
       [-6.13682833e-02, -8.18739670e-01,  4.23055751e-01,
        -8.92779979e-01, -8.61154409e-01, -8.76424472e-01,
        -8.86033116e-01, -8.65978291e-01],
       [ 1.32978161e+00,  2.45398811e+00,  8.26978122e+00,
        -2.07445107e+00, -8.71635590e-01, -1.04292469e+00,
        -9.69983938e-01, -9.05717723e-01],
```

```
       [-1.27613516e-01,  9.99442431e-01, -2.41920984e-01,
        -2.53104486e-02, -3.52373448e-02, -5.90995523e-01,
        -1.72451127e-01, -5.74555792e-01],
       [-1.84998957e+00, -4.55103250e-01, -4.41414004e-01,
         1.08122549e+00, -5.71873813e-01,  2.16815098e+00,
        -4.17307692e-01,  8.69310227e-01],
       [ 1.66100777e+00,  9.99442431e-01, -4.41414004e-01,
         1.72773145e-01, -4.27233515e-01, -4.48281049e-01,
        -3.33356869e-01, -4.95076928e-01],
       [ 2.69857881e-01,  2.45398811e+00, -1.75423310e-01,
        -6.33222167e-01,  4.74148053e-01, -6.86138506e-01,
        -8.15044026e-02, -6.40788178e-01],
       [-5.91330147e-01, -4.55103250e-01, -4.41414004e-01,
         8.21667676e-01,  1.65223280e+00,  7.88577730e-01,
         2.02426206e+00,  1.47864818e+00],
       [-4.58839681e-01, -4.55103250e-01, -4.41414004e-01,
         5.96262207e-01, -6.34760899e-01,  7.50053577e-02,
        -5.85209336e-01, -2.69886815e-01],
       [ 9.32310211e-01,  1.72671527e+00,  2.40697100e-02,
        -9.81576073e-01, -8.27614630e-01, -8.05067235e-01,
        -9.48996232e-01, -8.52731814e-01],
       [-5.25084914e-01,  9.99442431e-01, -3.74916331e-01,
         1.52281739e-01, -7.01840458e-01, -7.09924252e-01,
        -8.02082293e-01, -8.26238859e-01],
       [ 7.33574512e-01, -9.14668296e-02, -4.41414004e-01,
         6.23584082e-01, -2.17609895e-01,  2.17719832e-01,
        -2.14426538e-01,  2.15356839e-02],
       [ 1.37367416e-01, -9.14668296e-02, -3.08418657e-01,
        -6.94696386e-01, -6.74589387e-01, -8.28852981e-01,
        -8.23069999e-01, -8.52731814e-01],
       [ 4.68593580e-01, -8.18739670e-01,  2.40697100e-02,
        -7.42509667e-01, -8.25518394e-01, -8.52638726e-01,
        -7.53110980e-01, -8.52731814e-01],
       [ 1.32978161e+00,  2.09035169e+00, -4.41414004e-01,
         5.07466114e-01,  3.17619652e+00, -6.77091167e-02,
         2.98969651e+00,  8.29570795e-01],
       [ 4.02348347e-01, -9.14668296e-02, -4.41414004e-01,
         5.89431738e-01,  6.11895206e-02,  1.22576849e-01,
         5.13147255e-01,  2.86465229e-01],
       [-1.38627294e+00, -2.63692177e+00, -3.74916331e-01,
         4.39161426e-01,  7.82294775e-01, -4.72066794e-01,
         8.62942348e-01, -1.77161475e-01],
       [-1.93858749e-01,  2.72169591e-01,  9.05673835e-02,
        -2.23394042e-01, -8.46480756e-01, -7.33709998e-01,
        -6.90147864e-01, -8.12992382e-01],
       [-1.65125387e+00,  2.72169591e-01, -4.41414004e-01,
         1.00609033e+00,  2.48443857e+00,  2.00165076e+00,
         3.31850390e+00,  3.09471840e+00],
       [ 9.32310211e-01,  3.90853379e+00,  5.07789289e+00,
        -2.05395967e+00, -8.73731826e-01, -1.01913895e+00,
        -1.04693886e+00, -9.05717723e-01]])
```

Linear Regression

Model Training

In [69]:
```python
from sklearn.linear_model import LinearRegression
regression=LinearRegression()
```

In [70]:
```python
regression
```

Out[70]:

```
▾ LinearRegression
LinearRegression()
```

In [71]:

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

Out[71]:

```
▾ LinearRegression
LinearRegression()
```

Coefficients and Intercept

In [72]:

```
print(regression.coef_)
```

```
[-1.26727905 -0.54101797 -0.21943186  1.05414753  0.53223083  0.02362972
 -0.25283375  0.23646389]
```

In [73]:

```
print(regression.intercept_)
```

```
32.17791411042945
```

In [74]:

```
reg_pred=regression.predict(x_test)
```

In [75]:

```
reg_pred
```

Out[75]:

```
array([31.73234662, 33.39030119, 33.61405111, 31.87954716, 33.06586657,
       34.96791538, 33.85027588, 33.63331414, 32.04804585, 33.05943254,
       33.62988831, 27.82290771, 35.29347506, 29.44088401, 31.91096644,
       31.87132156, 33.9602693 , 28.18105257, 36.08101602, 34.34075554,
       32.79958404, 33.97078827, 33.7809388 , 32.80308456, 35.80069454,
       28.76933598, 32.31539014, 33.19651414, 27.726304  , 32.1854697 ,
       26.1619812 , 26.83083981, 34.07093816, 32.02044265, 33.00423786,
       30.61595053, 29.01549448, 32.44992806, 27.79955503, 35.53550024,
       32.73243283, 34.01374459, 33.88861804, 31.13859887, 36.41778054,
       34.19675343, 24.84645618, 34.98096926, 33.78536207, 29.7113244 ,
       31.21947136, 32.53177937, 35.69163645, 32.19023599, 30.33258848,
       30.33765202, 32.56225677, 36.14611876, 31.18504901, 33.45719224,
       32.38282845, 32.76026682, 31.20488789, 24.7061469 , 31.70034426,
       36.06315152, 29.54046257, 29.98448783, 34.87238095, 33.47909491,
       28.60101772, 32.16252428, 32.00059826, 31.01651762, 30.7682894 ,
       31.12273348, 32.40918113, 36.0516737 , 31.53536376, 36.54304548,
       25.16382485])
```

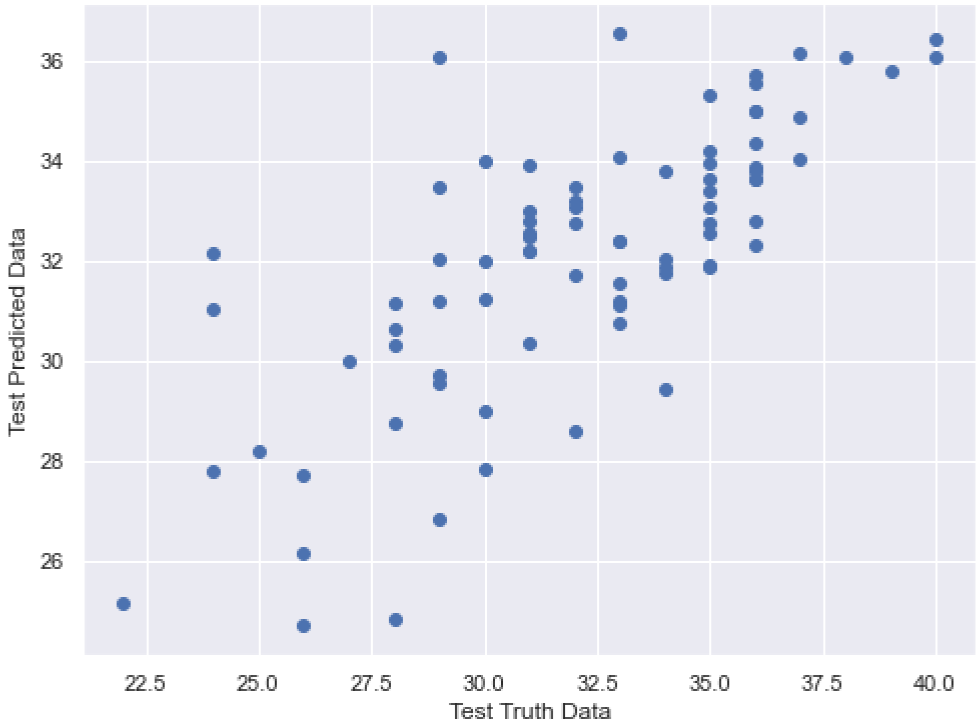
Assumption of Linear Regression

In [77]:

```
# Relationship between real data and predicted data
plt.scatter(y_test,reg_pred)
plt.xlabel('Test Truth Data')
plt.ylabel('Test Predicted Data')
```

Out[77]:

```
Text(0, 0.5, 'Test Predicted Data')
```



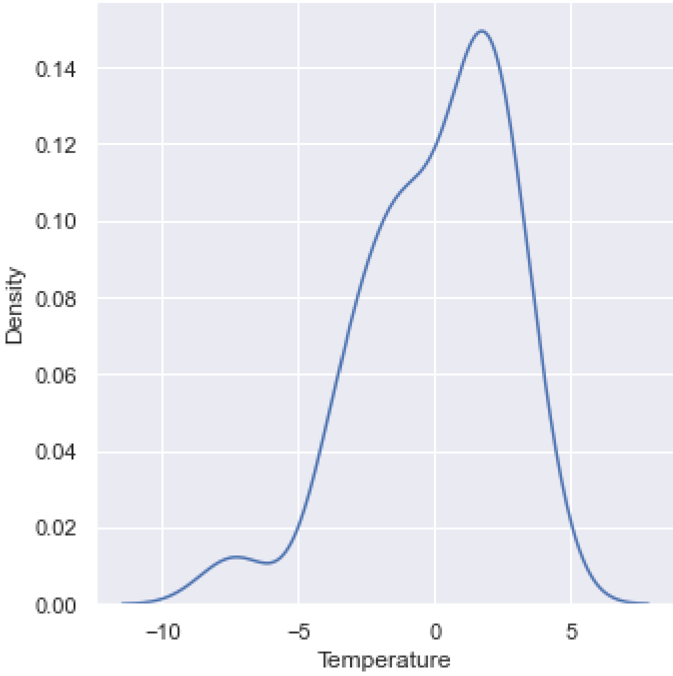
In [78]:
```
# Calculating residual
residuals=y_test - reg_pred
```

In [79]:
```
residuals
```

Out[79]:
```
162    2.267653
60     1.609699
61     2.385949
63     3.120453
69     1.934133
         ...
169    0.590819
232   -7.051674
144    1.464636
208   -3.543045
105   -3.163825
Name: Temperature, Length: 81, dtype: float64
```

In [80]:
```
# Distribution of residual are approximately Normal Distribution
sns.displot(residuals,kind="kde")
```

Out[80]:
```
<seaborn.axisgrid.FacetGrid at 0x28c3a96fca0>
```

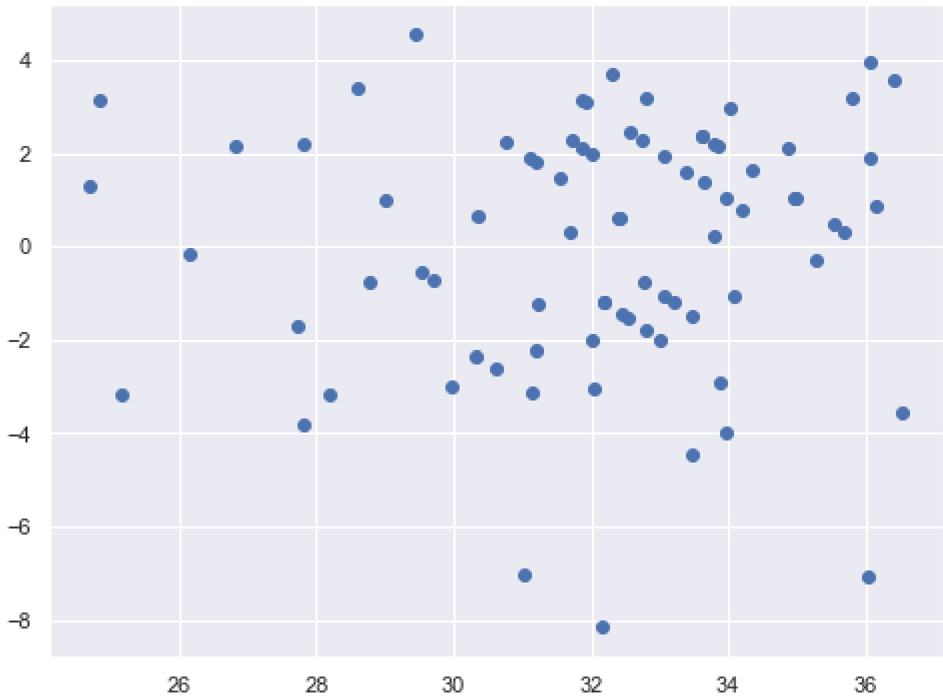


Observation

- Distribution is left skewed.

In [81]:
```
# Scatterplot with prediction and residual
# Uniform Distribution
plt.scatter(reg_pred,residuals)
```

Out[81]: <matplotlib.collections.PathCollection at 0x28c3aa0af40>



Performance Metrics

In [82]:
```
# 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)))
```

```
6.945272606141771
2.177260793883495
2.635388511423272
```

R Squared and Adjusted R Squared

In [83]:
```
# R Squared
from sklearn.metrics import r2_score
score=r2_score(y_test,reg_pred)
print(score)
```

```
0.5407291663922256
```

In [85]:
```
# Adjusted R Squared
1 - (1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[85]:
```
0.48969907376913957
```

Ridge regression

In [86]:
```
## Ridge
from sklearn.linear_model import Ridge
ridge=Ridge()
```

In [87]:
```
ridge.fit(x_train,y_train)
```

Out[87]:
```
▾ Ridge
Ridge()
```

In [88]:
```
print(ridge.coef_)
```

```
[-1.25556305 -0.54050633 -0.22268482  1.03678561  0.48209609  0.07521787
 -0.16554943  0.16717726]
```

In [89]:
```
print(ridge.intercept_)
```

```
32.17791411042945
```

In [90]:
```
ridge_pred=ridge.predict(x_test)
```

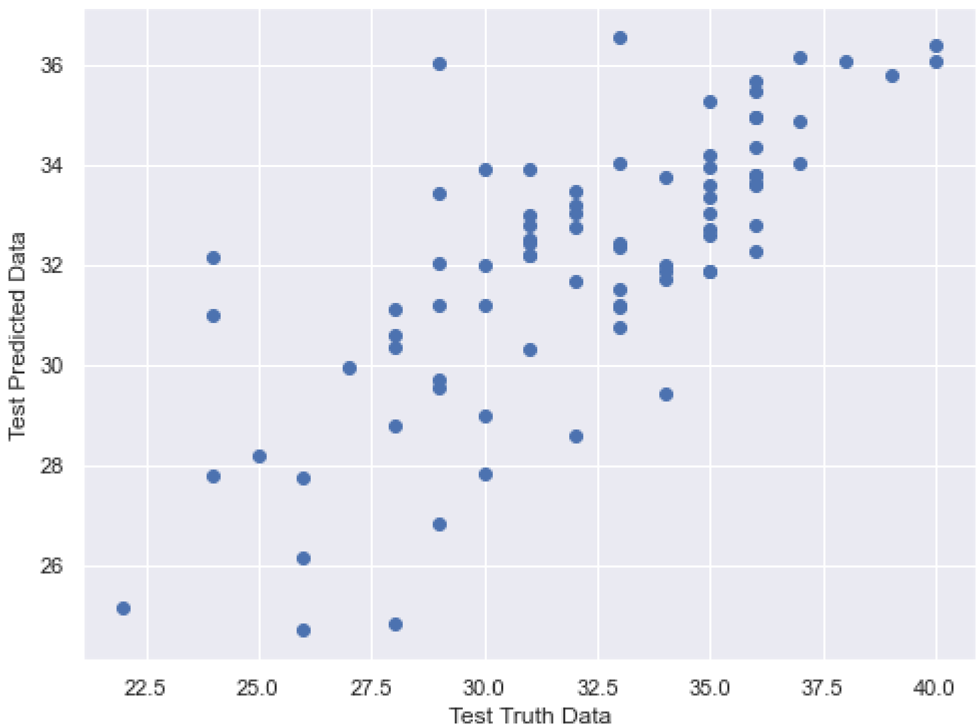
In [91]:
```
ridge_pred
```

Out[91]:
```
array([31.72605725, 33.37613461, 33.59099929, 31.87475756, 33.05298308,
       34.94776191, 33.81236937, 33.58905972, 32.04356065, 33.05011456,
       33.63429804, 27.82848492, 35.2714216 , 29.42705632, 31.89830943,
       31.88066019, 33.94209218, 28.19471634, 36.09412677, 34.35677201,
       32.78813328, 33.91846485, 33.7668546 , 32.81125404, 35.81850096,
       28.79848967, 32.27720636, 33.18688252, 27.74286139, 32.19181954,
       26.17139638, 26.85538177, 34.05623218, 32.01112485, 32.99003146,
       30.6051685 , 29.01741906, 32.43839626, 27.81381278, 35.49852092,
       32.71106191, 34.02287733, 33.90529872, 31.12955785, 36.40084135,
       34.18287694, 24.85860673, 34.97897127, 33.79331599, 29.71173246,
       31.21495697, 32.52728129, 35.69651118, 32.17999865, 30.34364518,
       30.32374592, 32.60145273, 36.15126634, 31.2167123 , 33.45003811,
       32.3710609 , 32.75537003, 31.19849874, 24.70008377, 31.6966219 ,
       36.0677736 , 29.56236916, 29.97858574, 34.88447303, 33.46788285,
       28.60599862, 32.14126044, 32.00168867, 31.00944793, 30.77692287,
       31.17275271, 32.43324777, 36.05170092, 31.53749347, 36.5617424 ,
       25.15849168])
```

Assumption of Ridge regression

In [92]:
```
# Relationship between real data and predicted data
plt.scatter(y_test,ridge_pred)
plt.xlabel('Test Truth Data')
plt.ylabel('Test Predicted Data')
```

Out[92]:
```
Text(0, 0.5, 'Test Predicted Data')
```



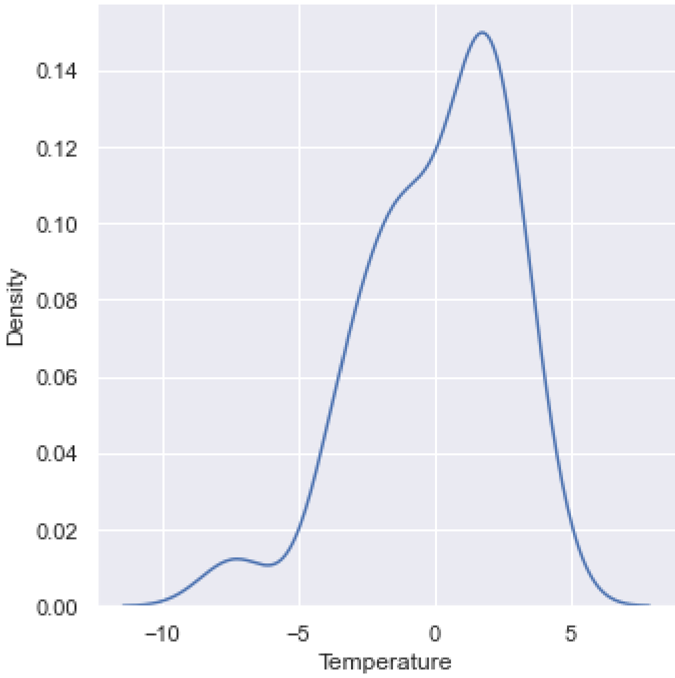
In [93]:
```
# Calculating residual
residuals=y_test - ridge_pred
```

In [94]:
```
residuals
```

Out[94]:
```
162    2.273943
60     1.623865
61     2.409001
63     3.125242
69     1.947017
         ...   
169    0.566752
232   -7.051701
144    1.462507
208   -3.561742
105   -3.158492
Name: Temperature, Length: 81, dtype: float64
```

In [95]:
```
# Distribution of residual are approximately Normal Distribution
sns.displot(residuals,kind="kde")
```

Out[95]:
```
<seaborn.axisgrid.FacetGrid at 0x28c3aa8bdc0>
```

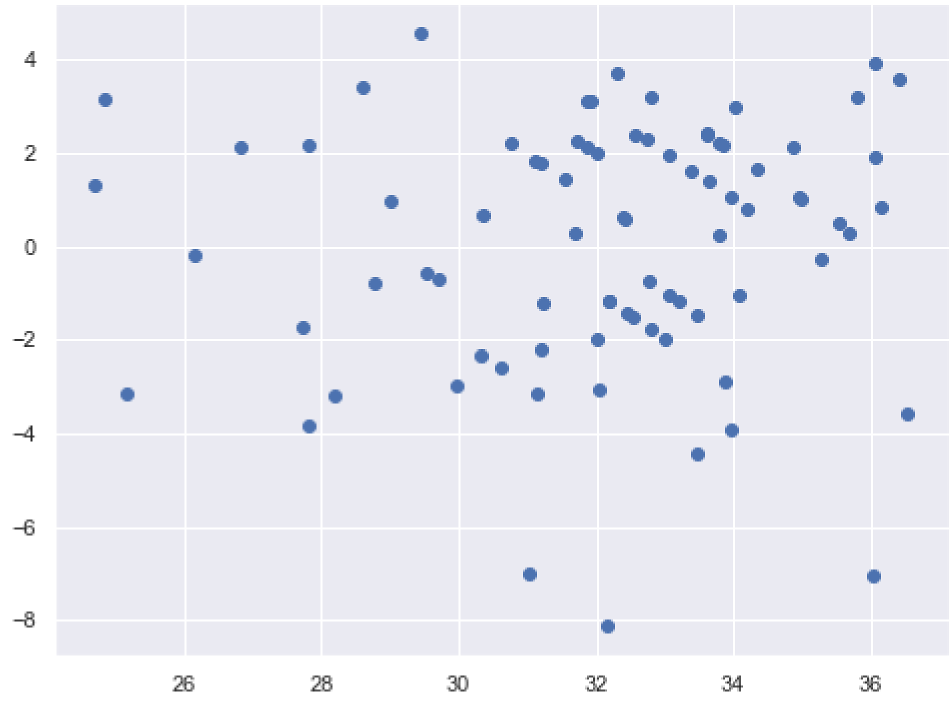


Observation

- Distribution is left skewed.

In [96]:
```
# Scatterplot with prediction and residual
# Uniform Distribution
plt.scatter(reg_pred,residuals)
```

Out[96]:
```
<matplotlib.collections.PathCollection at 0x28c3ab50220>
```



In [97]:
```
## Performance Metrics
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)))
```

```
6.937034660929738
2.177014764095557
2.6338251006719746
```

In [98]:
```
# R Squared
from sklearn.metrics import r2_score
score=r2_score(y_test,ridge_pred)
print(score)
```

```
0.5412739179346487
```

In [99]:
```
# Adjusted R Squared
1 - (1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[99]:
```
0.49030435326072075
```

Lasso regression

In [100…
```
from sklearn.linear_model import Lasso
lasso=Lasso()
```

In [102…
```
lasso
```

Out[102]:
```
▾ Lasso
Lasso()
```

In [103…
```
lasso.fit(x_train, y_train)
```

Out[103]:

```
▾ Lasso
Lasso()
```

In [104…

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

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

In [105…

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

```
32.17791411042945
```

In [106…

```
lasso_pred=lasso.predict(x_test)
```

In [107…

```
lasso_pred
```

Out[107]:

```
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])
```

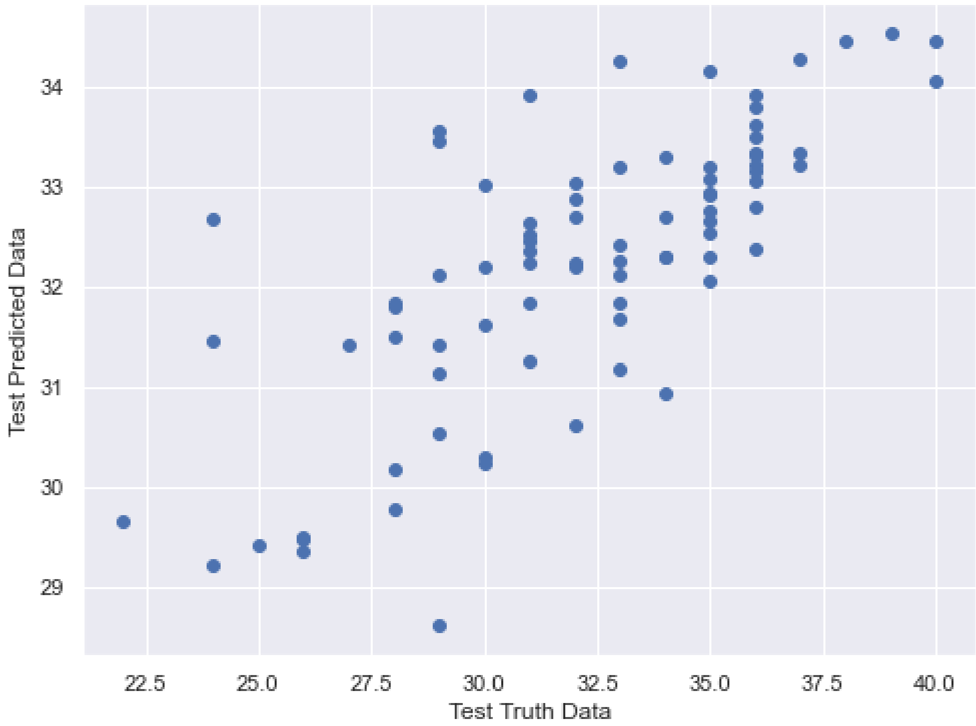
Assumption of Lasso regression

In [108…

```
# Relationship between real data and predicted data
plt.scatter(y_test,lasso_pred)
plt.xlabel('Test Truth Data')
plt.ylabel('Test Predicted Data')
```

Out[108]:

```
Text(0, 0.5, 'Test Predicted Data')
```



In [109…

```
# Calculating residual
residuals=y_test - lasso_pred
```

In [110…

```
residuals
```

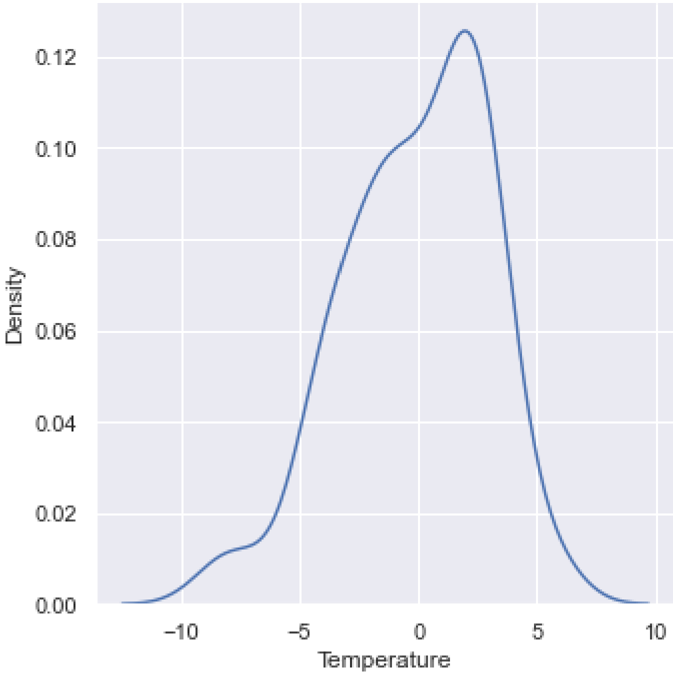
Out[110]:

```
162    1.702999
60     2.325597
61     2.933905
63     2.926530
69     2.075023
         ...
169    0.583574
232   -4.568827
144    0.882714
208   -1.267362
105   -7.667085
Name: Temperature, Length: 81, dtype: float64
```

In [111…

```
# Distribution of residual are approximately Normal Distribution
sns.displot(residuals,kind="kde")
```

Out[111]: <seaborn.axisgrid.FacetGrid at 0x28c3ab66c70>



Observation

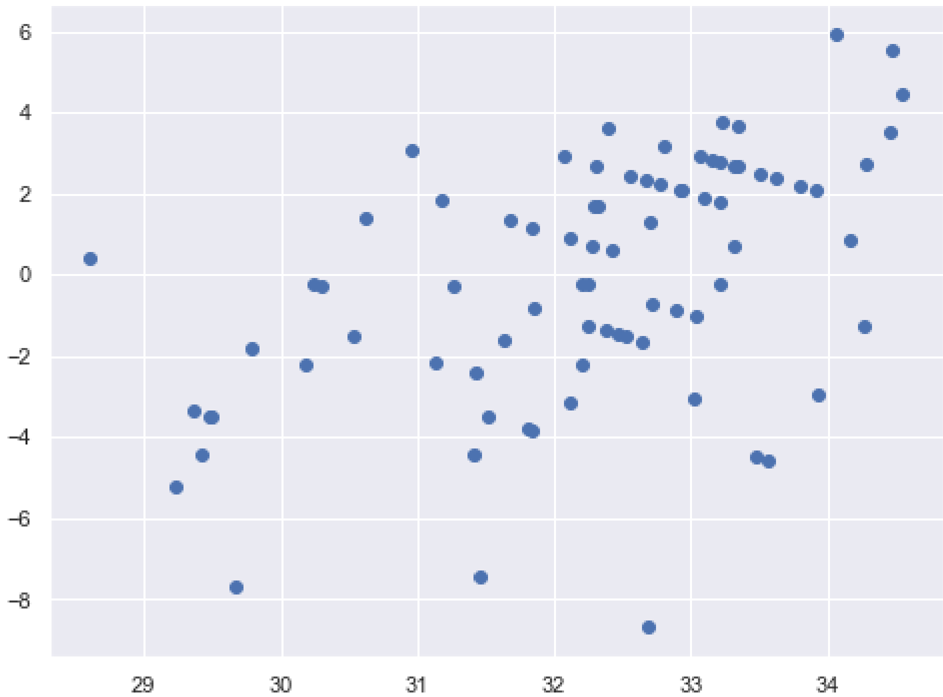
- Distribution is left skewed.

In [112…

```
# Scatterplot with prediction and residual
# Uniform Distribution
plt.scatter(lasso_pred,residuals)
```

Out[112]:

```
<matplotlib.collections.PathCollection at 0x28c3ac5dee0>
```



In [113…

```
## Performance Metrics
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
```

In [114…
```python
# R Squared
from sklearn.metrics import r2_score
score=r2_score(y_test,lasso_pred)
print(score)
```

```
0.39784019626969913
```

In [115…
```python
# Adjusted R Squared
1 - (1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[115]:
```
0.33093355141077685
```

Elastic Net regression

In [116…
```python
from sklearn.linear_model import ElasticNet
elastic=ElasticNet()
```

In [117…
```python
elastic
```

Out[117]:
```
▾ ElasticNet
ElasticNet()
```

In [118…
```python
elastic.fit(x_train,y_train)
```

Out[118]:
```
▾ ElasticNet
ElasticNet()
```

In [119…
```python
print(elastic.coef_)
```

```
[-0.68377791 -0.10873001 -0.02083121  0.70493387  0.          0.22896344
  0.07733533  0.1822098 ]
```

In [120…
```python
print(elastic.intercept_)
```

```
32.17791411042945
```

In [121…
```python
elastic_pred=elastic.predict(x_test)
```

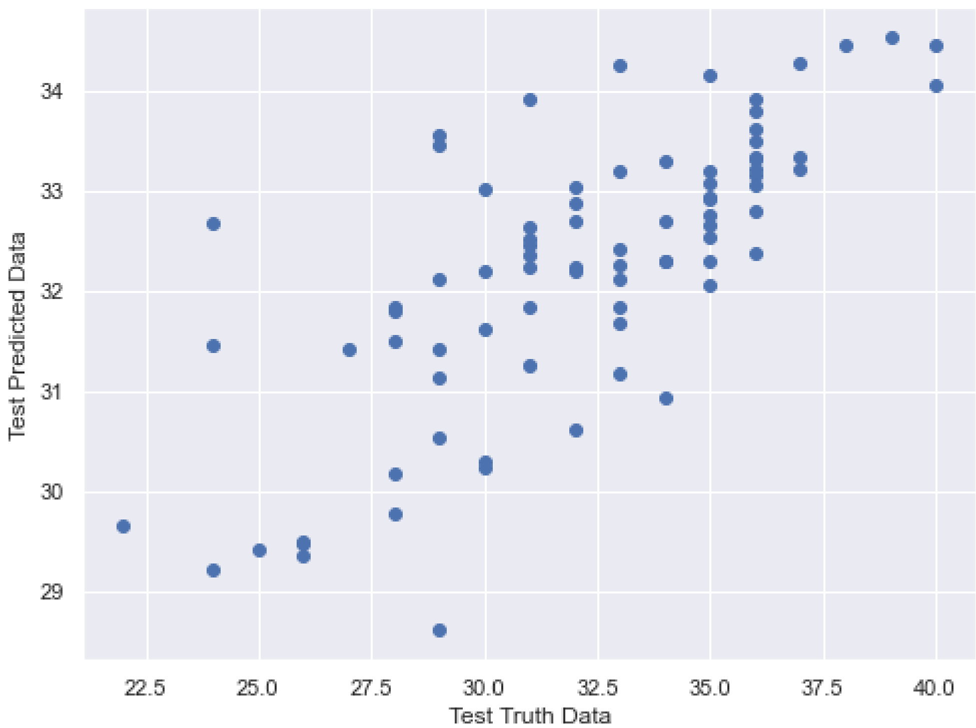
In [122…
```python
elastic_pred
```

Out[122]:
```
array([31.92778274, 32.95250724, 32.71119722, 31.77134083, 32.92801741,
       33.99050179, 33.53902004, 32.58962193, 31.93651966, 32.67134649,
       33.35804182, 29.90659996, 34.18554019, 30.68186444, 32.61165892,
       32.21481692, 32.7945839 , 29.43974043, 35.17418981, 33.89534297,
       32.36213   , 33.43380855, 33.5283983 , 32.8386007 , 35.19633069,
       30.01766464, 32.08455825, 32.54878071, 29.38878202, 32.00984925,
       29.15361138, 28.66254395, 33.22090486, 32.27099015, 32.64751022,
       31.34705984, 30.05651772, 32.28102625, 29.22022742, 34.01465745,
       32.84551243, 33.52330689, 34.21265774, 31.55337147, 34.53418824,
       33.38994892, 29.2787136 , 33.93541974, 33.3964204 , 30.36875762,
       31.40639778, 32.30324864, 34.65294213, 32.00860236, 31.35483638,
       30.93541449, 32.30838596, 34.84461644, 31.58190605, 33.45940298,
       32.0717569 , 32.40440528, 31.24375379, 28.88836001, 31.89035871,
       34.88631868, 30.84584502, 31.00368194, 33.82667918, 32.89340265,
       30.24712959, 32.16831989, 32.17222913, 31.2018347 , 31.01375885,
       31.77514228, 32.45739477, 33.65628852, 31.75201203, 35.27466484,
       29.08242011])
```

Assumption of ElasticNet regression

In [123…
```
# Relationship between real data and predicted data
plt.scatter(y_test,lasso_pred)
plt.xlabel('Test Truth Data')
plt.ylabel('Test Predicted Data')
```

Out[123]:
```
Text(0, 0.5, 'Test Predicted Data')
```



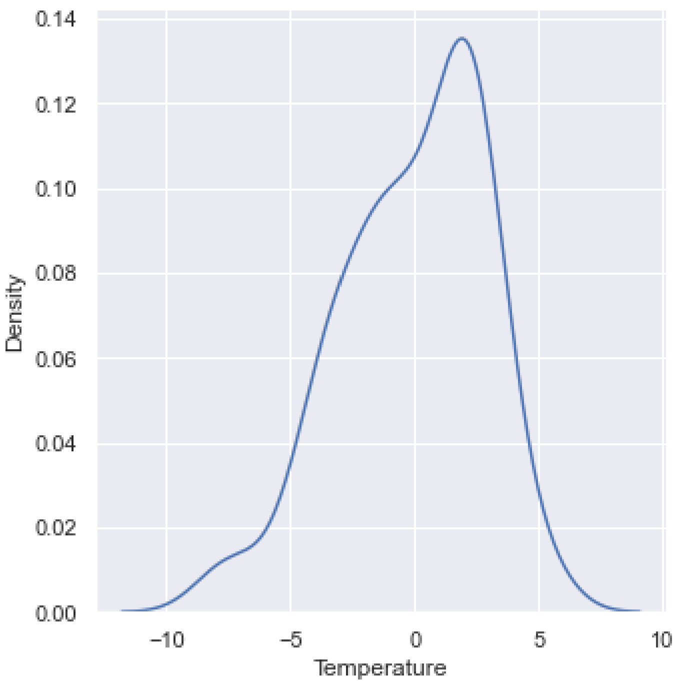
In [124…
```
# Calculating residual
residuals=y_test - elastic_pred
```

In [125…
```
residuals
```

Out[125]:
```
162    2.072217
60     2.047493
61     3.288803
63     3.228659
69     2.071983
         ...
169    0.542605
232   -4.656289
144    1.247988
208   -2.274665
105   -7.082420
Name: Temperature, Length: 81, dtype: float64
```

In [126…
```
# Distribution of residual are approximately Normal Distribution
sns.displot(residuals,kind="kde")
```

Out[126]: <seaborn.axisgrid.FacetGrid at 0x28c3acf5400>

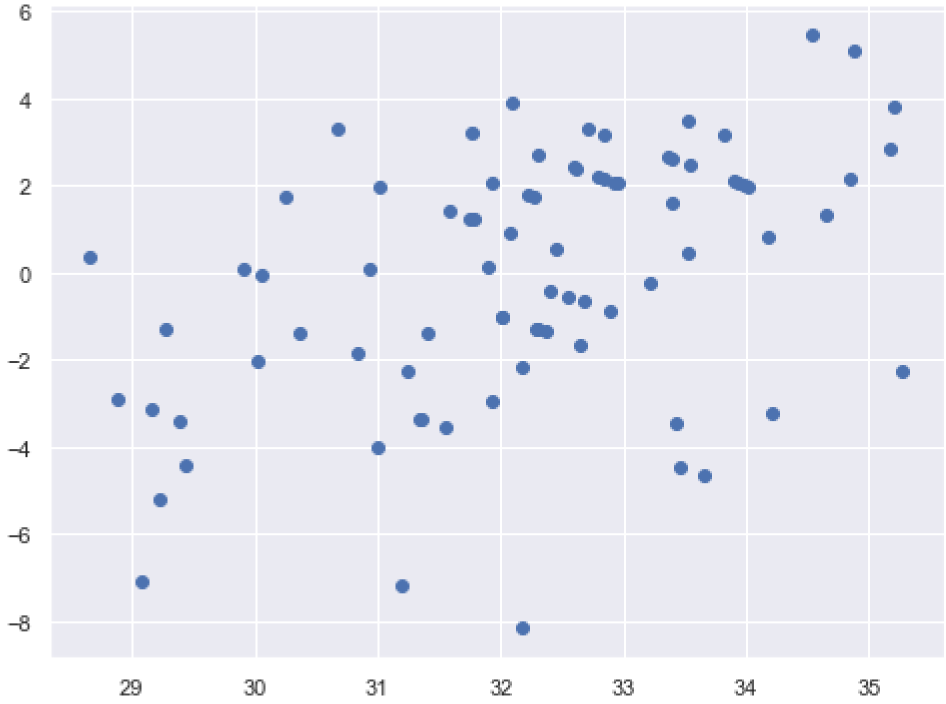


Observation

- Distribution is left skewed.

In [127…
```
# Scatterplot with prediction and residual
# Uniform Distribution
plt.scatter(elastic_pred,residuals)
```

Out[127]: <matplotlib.collections.PathCollection at 0x28c3ada0700>



In [128…

```
## Performance Metrics
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)))
```

```
8.317775387996829
2.3928315256929684
2.8840553718673343
```

In [129…

```
# R Squared
from sklearn.metrics import r2_score
score=r2_score(y_test,elastic_pred)
print(score)
```

```
0.4499695184276321
```

In [130…

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

Out[130]:

```
0.38885502047514686
```

THE END