The code used in our Machine Learning project :

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing

%matplotlib inline
weather_df = pd.read_csv('kanpur.csv', parse_dates=['date_time'], index
_col='date_time')
weather_df.head(5)
weather_df.columns
weather_df.shape
weather_df.describe()
weather_df.isnull().any()
weather_df_num=weather_df.loc[:,['maxtempC','mintempC','cloudcover','hu
midity','tempC', 'sunHour','HeatIndexC', 'precipMM', 'pressure','windsp
eedKmph']]
weather_df_num.head()
weather_df_num.shape
weather_df_num.columns
weather_df_num.plot(subplots=True, figsize=(25,20))
weather_df_num['2019':'2020'].resample('D').fillna(method='pad').plot(s
ubplots=True, figsize=(25,20))
weather_df_num.hist(bins=10,figsize=(15,15))
weth=weather_df_num['2019':'2020']
weth.head()
weather_y=weather_df_num.pop("tempC")
weather_x=weather_df_num
train_X,test_X,train_y,test_y=train_test_split(weather_x,weather_y,test
_size=0.2,random_state=4)
train_X.shape
train_X.head(10)
train_y.shape
train_y.head()
plt.scatter(weth.mintempC, weth.tempC)
plt.xlabel("Minimum Temperature")
plt.ylabel("Temperature")
plt.show()
plt.scatter(weth.HeatIndexC, weth.tempC)
plt.xlabel("Heat Index")
plt.ylabel("Temperature")
```

```
plt.show()
plt.scatter(weth.pressure, weth.tempC)
plt.xlabel("Minimum Temperature")
plt.ylabel("Temperature")
plt.show()
plt.scatter(weth.mintempC, weth.tempC)
plt.xlabel("Minimum Temperature")
plt.ylabel("Temperature")
plt.show()
model=LinearRegression()
model.fit(train_X,train_y)
prediction = model.predict(test_X)
#calculating error
np.mean(np.absolute(prediction-test_y))
print('Variance score: %.2f' % model.score(test_X, test_y))
for i in range(len(prediction)):
  prediction[i]=round(prediction[i],2)
pd.DataFrame({'Actual':test_y,'Prediction':prediction,'diff':(test_y-
prediction)})
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(random_state=0)
regressor.fit(train_X,train_y)
prediction2=regressor.predict(test_X)
np.mean(np.absolute(prediction2-test_y))
print('Variance score: %.2f' % regressor.score(test_X, test_y))
for i in range(len(prediction2)):
  prediction2[i]=round(prediction2[i],2)
pd.DataFrame({'Actual':test_y,'Prediction':prediction2,'diff':(test_y-
prediction2)})
from sklearn.metrics import r2_score
print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - te
st_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - tes
t_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y,prediction))
print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction2 - t
est_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction2 - te
st_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y,prediction2))
```